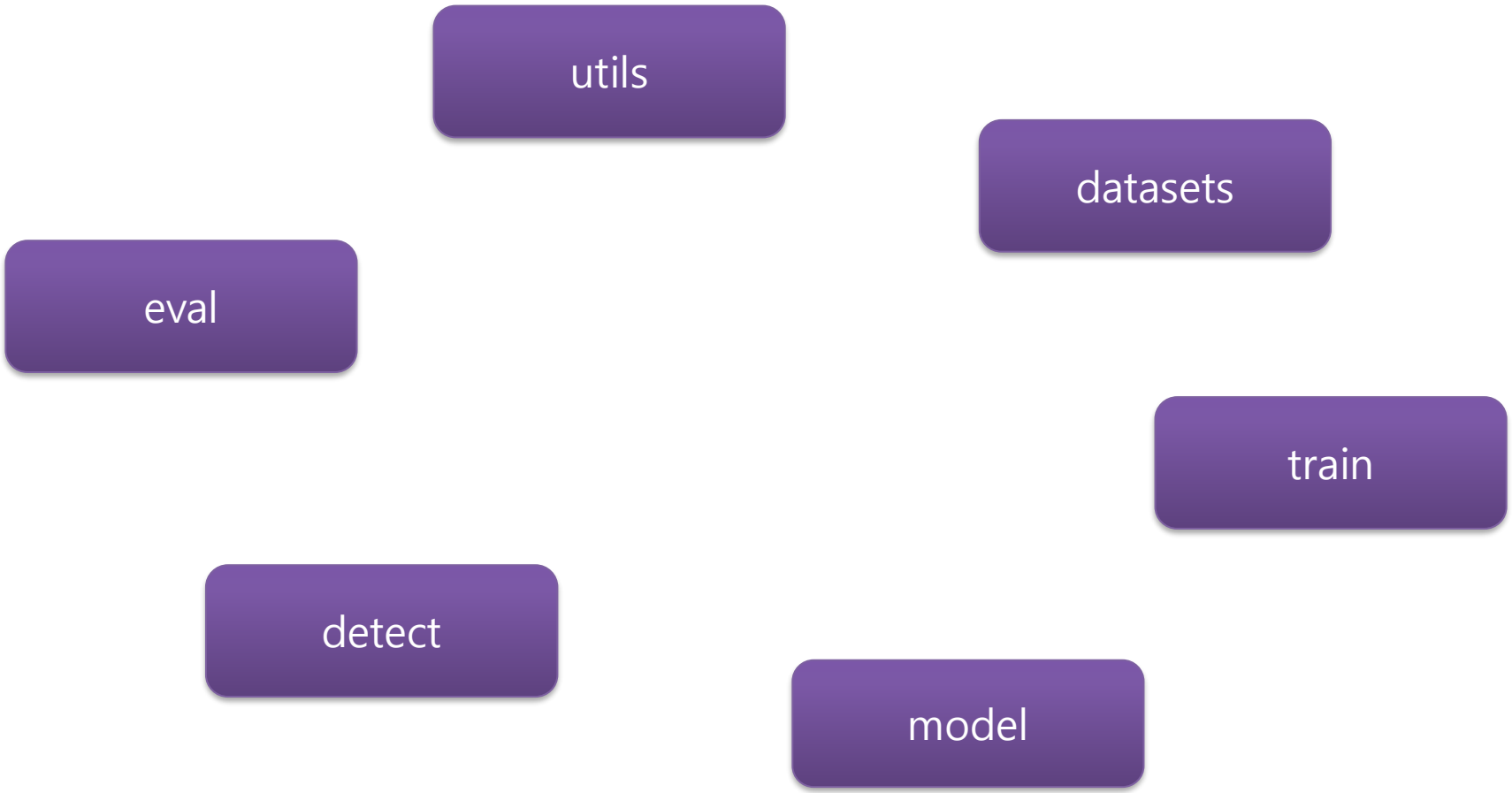
SSD code 흐름

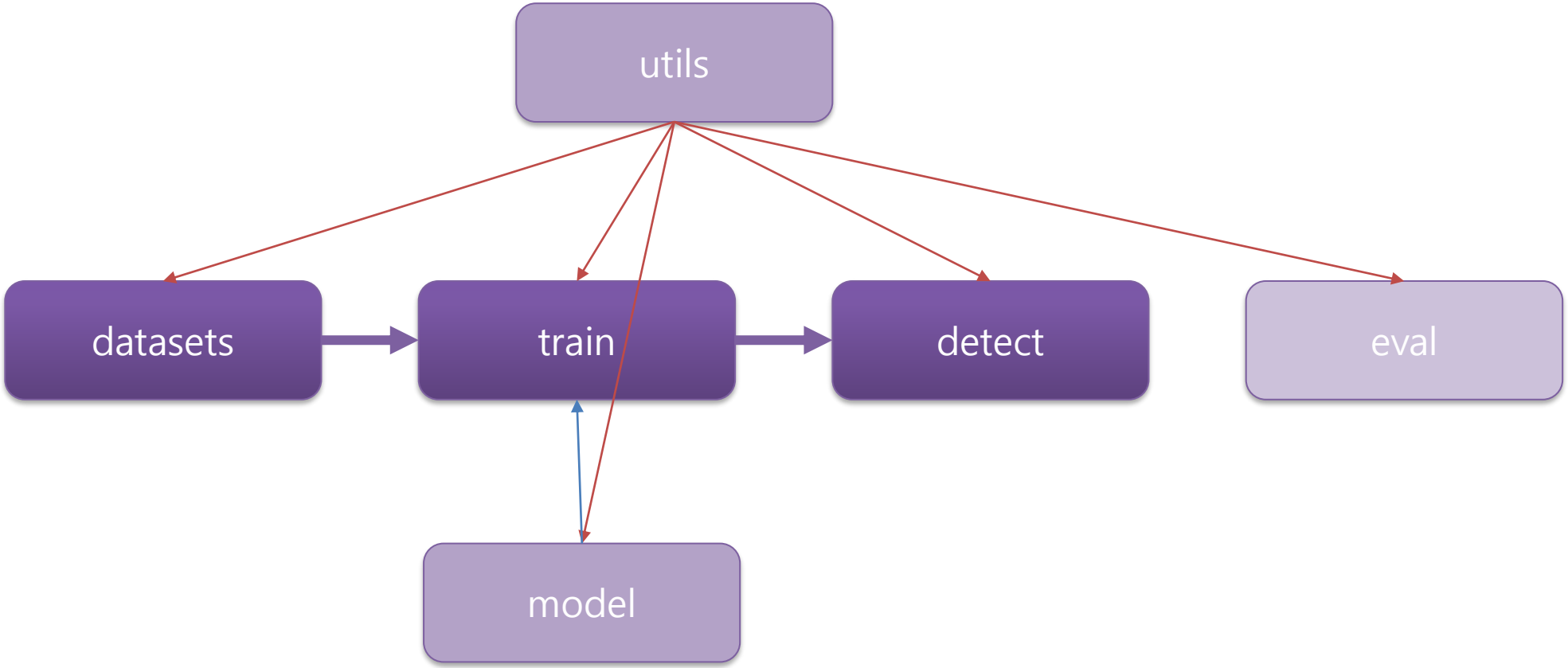
이연희

SSD Tutorial Code

SSD Tutorial Code



SSD Tutorial Code



utils.py

*색깔을 다르게 표시한거는 진한 보라색이 밝은 보라색의 함수들을 호출하기 때문에 관련이 있어서 이렇게 정리를 해둔 것입니다. 예를 들면 마지막 줄의 transform에서 밑의 다섯가지의 함수들을 포함하고 있다고 보시면 됩니다.

create_data_lists(voc07_path, voc12_path, output_folder)

parse_annotation(annotation_path)

calculate_mAP(det_boxes, det_labels, det_scores, true_boxes, true_labels, true_difficulties)

find_jaccard_overlap(set_1, set_2)

find_intersection(set_1, set_2)

xy_to_cxcy(xy)

cxcy_to_xy(cxcy)

cxcy_to_gcxgcy(cxcy, priors_cxcy)

gcxgcy_to_cxcy(gcxgcy, priors_cxcy)

decimate(tensor, m)

accuracy(scores, targets, k)

clip_gradient(optimizer, grad_clip)

adjust_learning_rate(optimizer, scale)

class AverageMeter(object)

transform(image, boxes, labels, difficulties, split)

photometric_distort(image)

expand(image, boxes, filler)

random_crop(image, boxes, labels, difficulties)

flip(image, boxes)

resize(image, boxes, dims=(300, 300), return_percent_coords=True)

save_checkpoint(epoch, model, optimizer)

datasets.py

data

PASCAL_VOC_2007
PASCAL_VOC_2012

20가지 유형의 개체

```
voc_labels = ('aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable',
              'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor')
```

각 이미지는 하나 이상의 ground truth 객체를 포함할 수 있음

각 이미지는 다음이 포함됨

- 절대 경계 좌표의 경계 상자
- 레이블(위에서 언급한 객체 유형 중 하나)
- 감지 어려움

utils.py

transform(image, boxes, labels, difficulties, split)

photometric_distort(image)

expand(image, boxes, filler)

random_crop(image, boxes, labels, difficulties)

flip(image, boxes)

resize(image, boxes, dims=(300, 300),
return_percent_coords=True)

- ✓ 밝기, 대비, 채도 및 색조를 각각 50%의 확률로 임의의 순서로 무작위로 조정
- ✓ Zoom out을 수행 - 크거나 부분적인 물체를 감지하는 학습에 도움이 된다.
- ✓ 이미지의 크기를 300,300픽셀로 조정
- ✓ **Normalize** : VGG기반을 pretrain하는데 사용된 imageNet 데이터의 평균 및 표준 편차로 이미지를 정규화

train.py

- 파라미터 설정

```
# Learning parameters
checkpoint = None  # path to model checkpoint, None if none
batch_size = 8  # batch size
iterations = 120000  # number of iterations to train
workers = 4  # number of workers for loading data in the DataLoader
print_freq = 200  # print training status every __ batches
lr = 1e-3  # learning rate
decay_lr_at = [80000, 100000]  # decay learning rate after these many iterations
decay_lr_to = 0.1  # decay learning rate to this fraction of the existing learning rate
momentum = 0.9  # momentum
weight_decay = 5e-4  # weight decay
grad_clip = None  # clip if gradients are exploding, which may happen at larger batch size

cudnn.benchmark = True
```

- 데이터 불러오기

```
# Custom dataloaders
train_dataset = PascalVOCDataset(data_folder,
                                 split='train',
                                 keep_difficult=keep_difficult)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True,
                                           collate_fn=train_dataset.collate_fn, num_workers=workers,
                                           pin_memory=True)  # note that we're passing the collate function here
```

- **train**

```
# Epochs
for epoch in range(start_epoch, epochs):

    # Decay learning rate at particular epochs
    if epoch in decay_lr_at:
        adjust_learning_rate(optimizer, decay_lr_to)

    # One epoch's training
    train(train_loader=train_loader,
          model=model,
          criterion=criterion,
          optimizer=optimizer,
          epoch=epoch)

    # Save checkpoint
    save_checkpoint(epoch, model, optimizer)
```

model.py

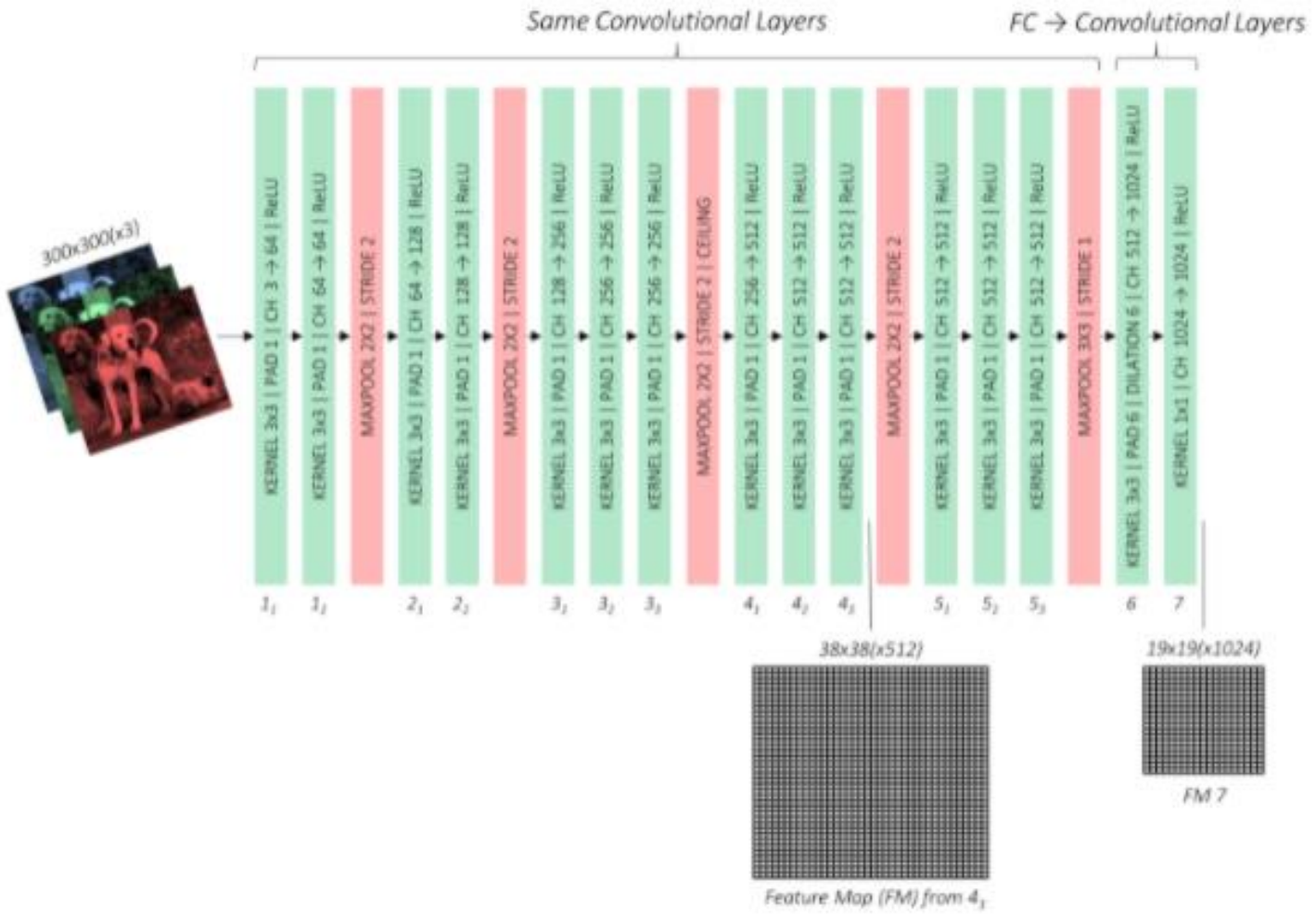
- Base convolutions
  -> lower-level feature maps를 제공하는 기존의 이미지 분류 아키텍처

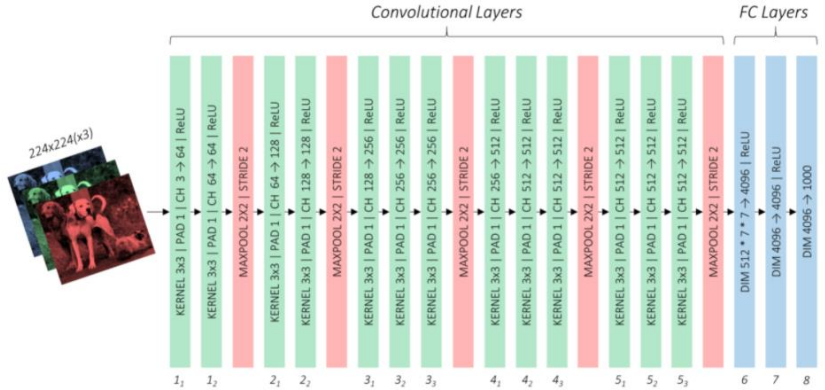
- Auxiliary convolutions
  -> higher-level feature maps를 제공하는 base network 위에 추가된 것

- Prediction convolutions
  ->이러한 feature map에서 위치와 objects를 식별

model.py

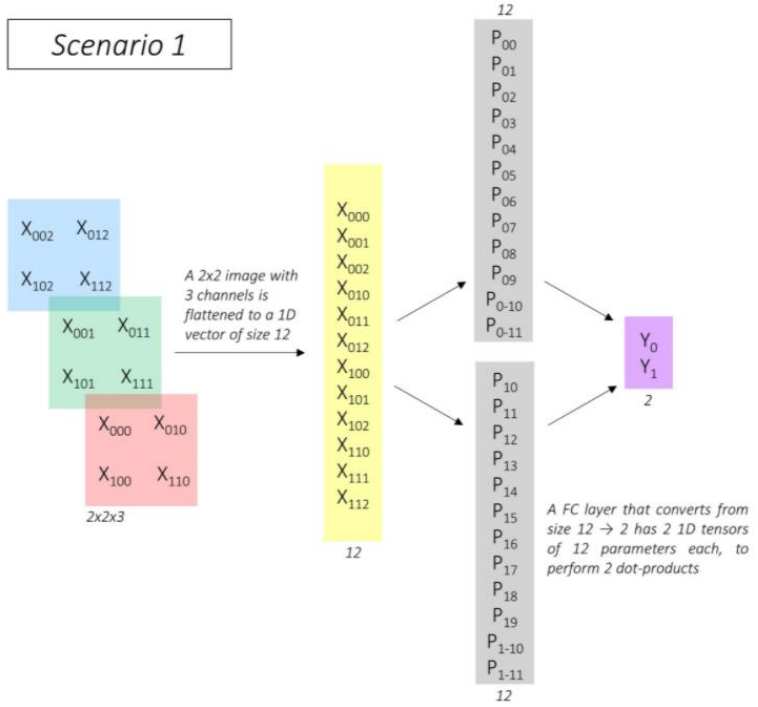
- Base convolutions

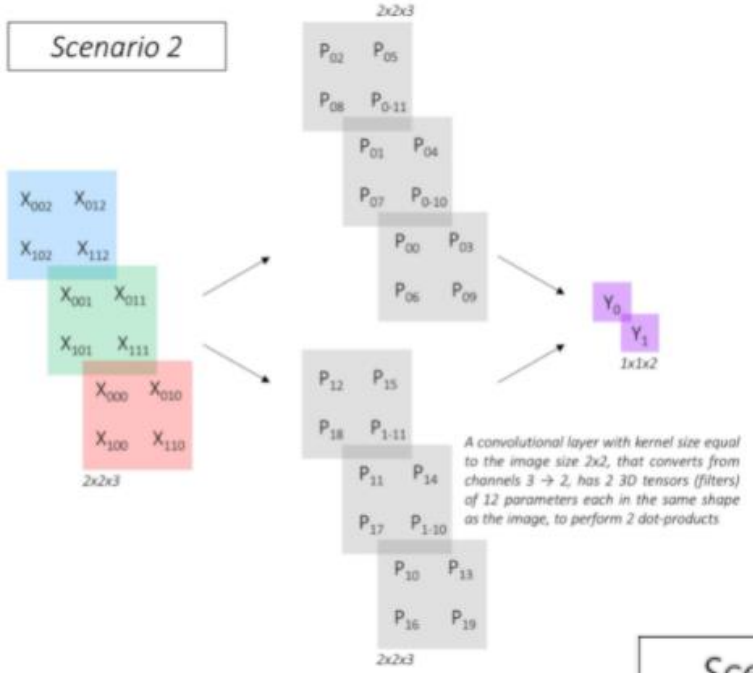




model.py

- Base convolutions





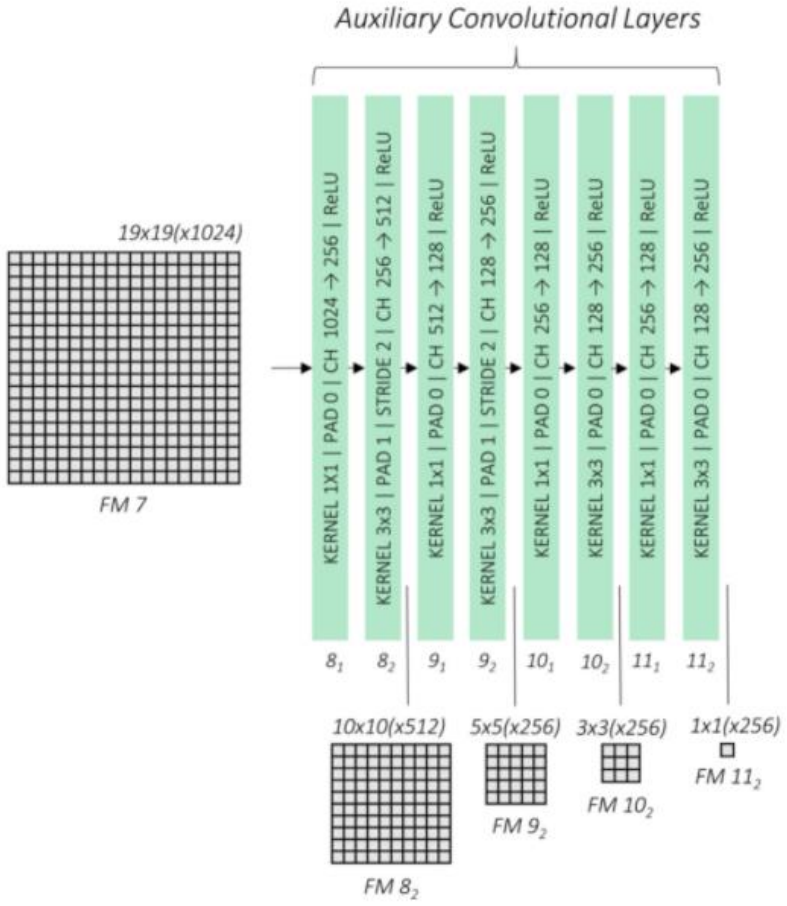
Scenario 1 ≡ Scenario 2

$Y_0 = P_{00} X_{000} + P_{01} X_{001} + P_{02} X_{002} + P_{03} X_{010} + \dots$

$Y_1 = P_{10} X_{000} + P_{11} X_{001} + P_{12} X_{002} + P_{13} X_{010} + \dots$

model.py

- Auxiliary convolutions



2개의 layer로 구성된 4개의 컨볼루션 blocks

Base -> pooling을 통해 크기를 줄임

Auxiliary -> 2개의 layer를 통해 크기를 줄임

- auxiliary structure에서 점차 output size (w, h) 를 작게 만드는 과정을 거칠 때 공간 정보를 최대한 유지하면서 output size를 줄이는 것,
- (1024, 19, 19) → (512, 10, 10) 으로 바로 줄이기 보다는 (256, 19, 19) 과정을 한 번 거쳐 이미지의 전반적인 공간 정보는 최대한 함축 시켜 유지한 채 output size를 축소

detect.py

```
# Transform
image = normalize(to_tensor(resize(original_image)))
```

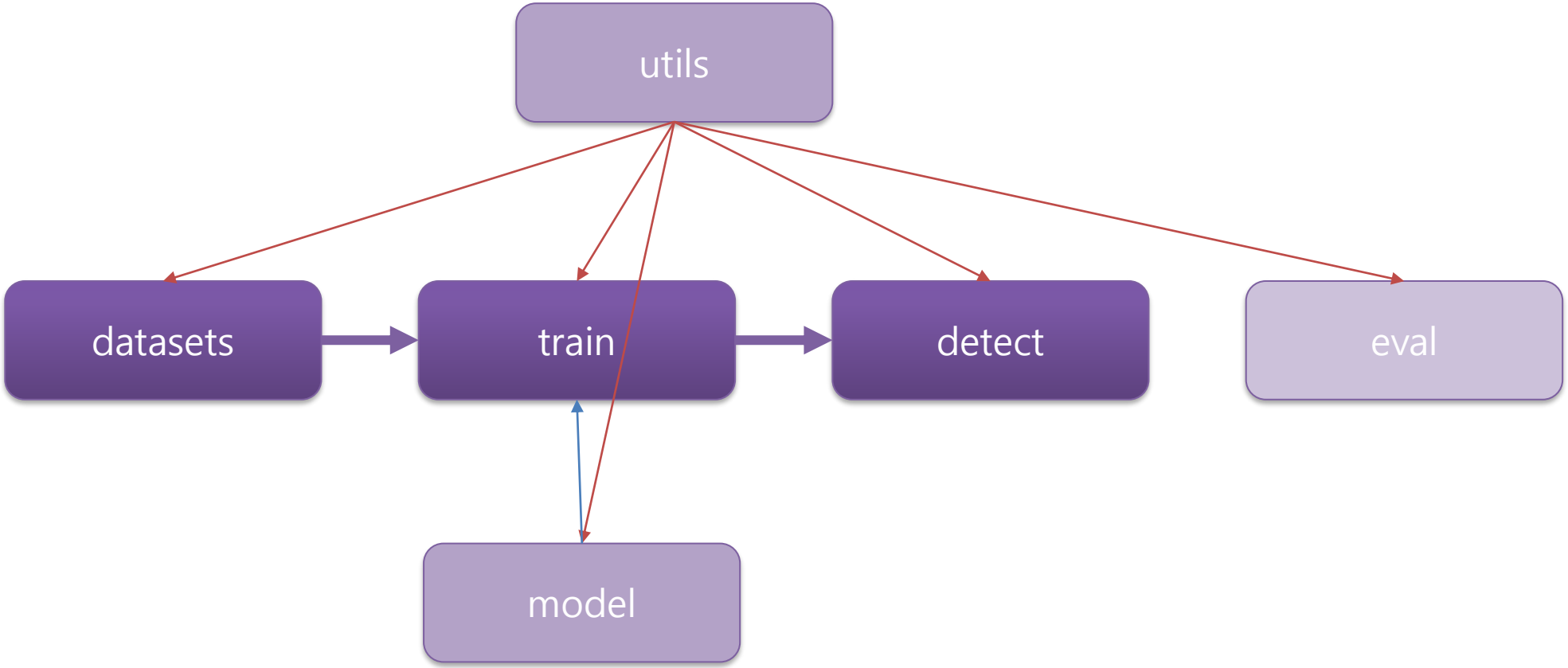
```
# Transforms
resize = transforms.Resize((300, 300))
to_tensor = transforms.ToTensor()
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

RGB 채널의 크기를 조정하고 정규화하여 이미지를 사전 처리

↓

detect

SSD Tutorial Code



Thank you