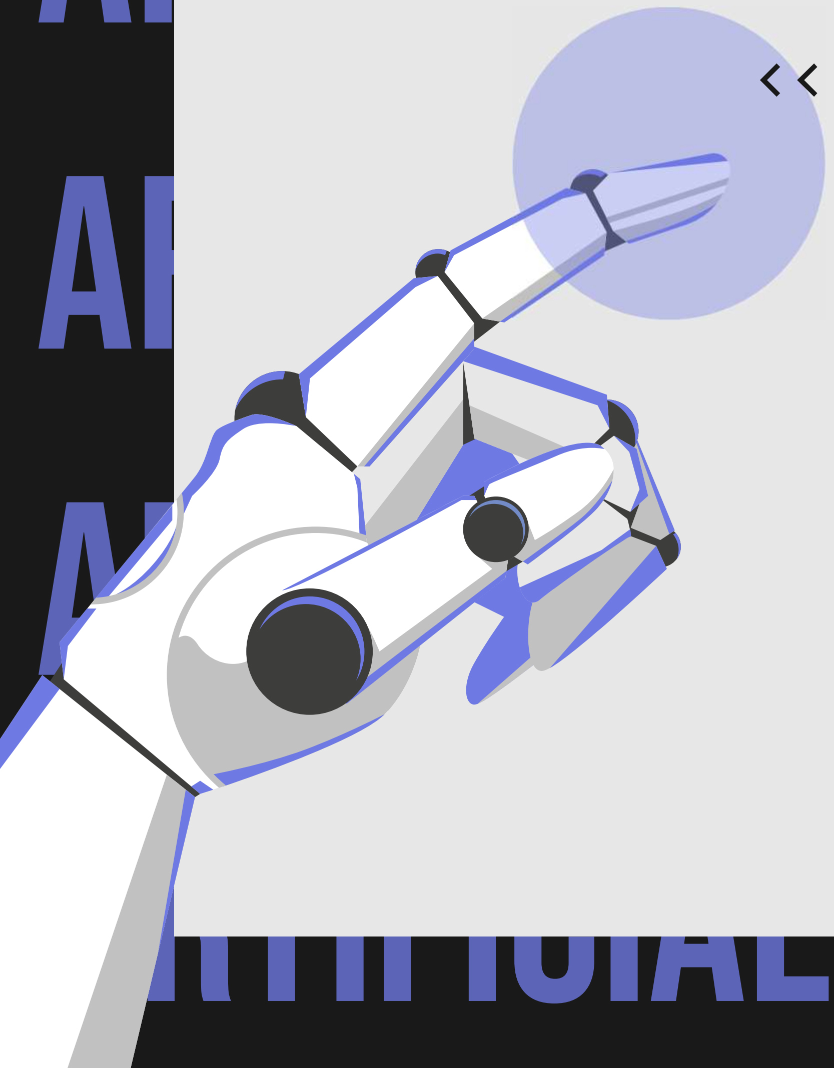


YOLO V4 코드

학부연구생 2단계 과정 세미나

18011816 김동영

TABLE OF CONTENTS

01.
BACKBONE
CSPDarknet53

02.
NECK
PANet

03.
LOSS FUNCTION
Bbox_ciou

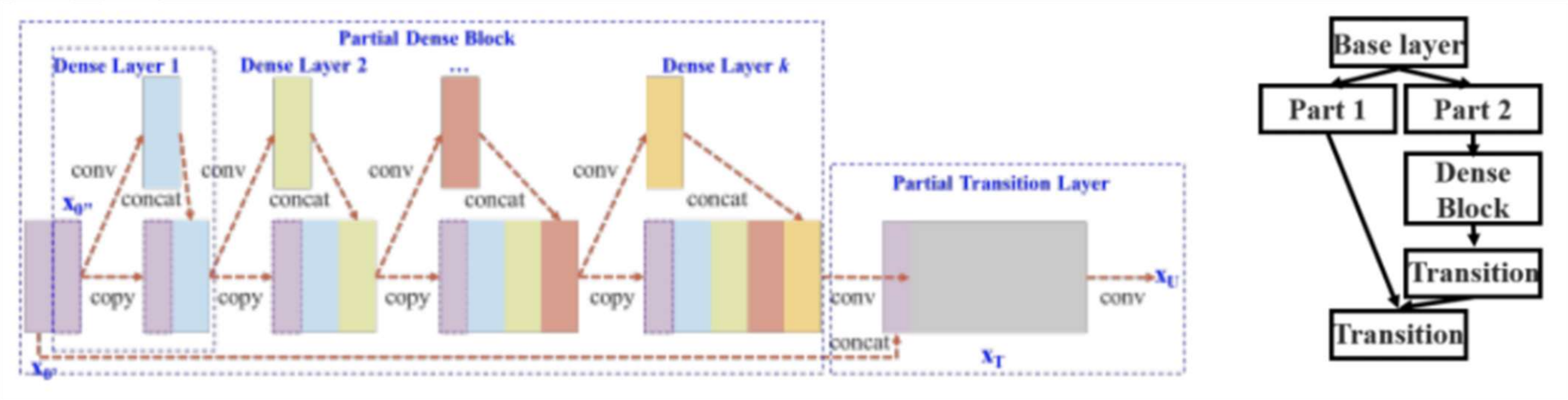
ARTIFICIAL INTELLIGENCE (AI)

01.

BACKBONE

CSPDarknet53

CSPDARKNET53



Layer도 깊고 파라미터 수도 많아 속도를 잡기 위해 CSPNet 사용

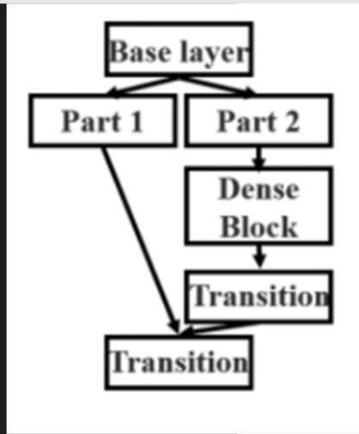
Input feature map을 두 파트로 나눠서 한 파트는 아무 연산하지 않고 전달

나머지 파트만 연산에 참여 후 Concat 해줌 -> 정확성 손실 X, 연산속도 빠름

이 CSP Layer를 Darknet53에 연결해 CSPDarknet53을 Backbone으로 사용

CSPDARKNET53

```
route = input_data
route = convolutional(route, (1, 1, 64, 64), activate_type="mish")
input_data = convolutional(input_data, (1, 1, 64, 64), activate_type="mish")
for i in range(1):
    input_data = residual_block(input_data,  64,  32, 64, activate_type="mish")
input_data = convolutional(input_data, (1, 1, 64, 64), activate_type="mish")
input_data = tf.concat([input_data, route], axis=-1)
input_data = convolutional(input_data, (1, 1, 128, 64), activate_type="mish")
input_data = convolutional(input_data, (3, 3, 64, 128), downsample=True, activate_type="mish")
```

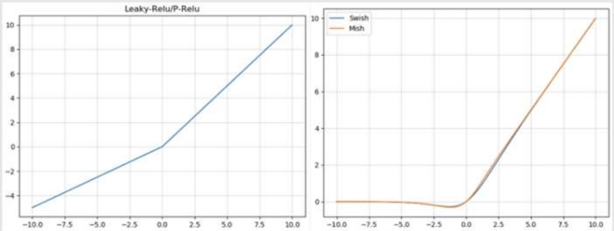


Input_data, route 두 파트로 나눔

Input_data 부분만 계산에 참여 후 concat을 통해 Input_data, route 두 파트를 합침

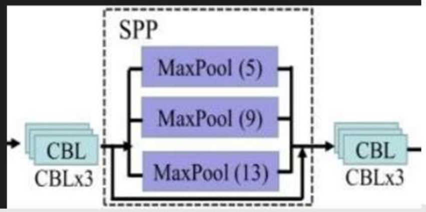
논문에서 봤듯이 activate_type 은 mish로 사용

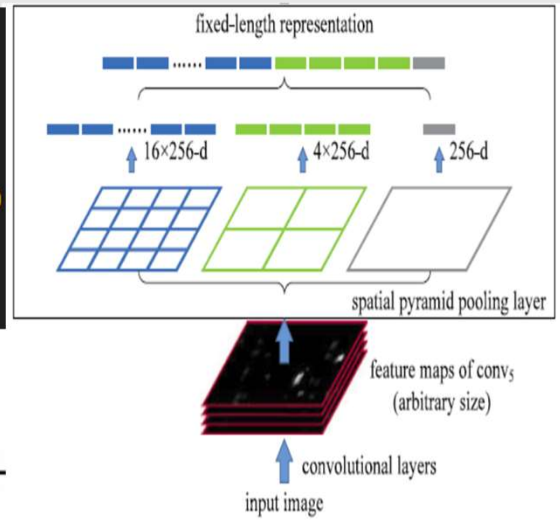
다음 부분을 위해 downsampling 함 (128 -> 64)



CSPDARKNET53

```
input_data = convolutional(input_data, (1, 1, 1024, 1024), activate_type="mish")
input_data = convolutional(input_data, (1, 1, 1024, 512))
input_data = convolutional(input_data, (3, 3, 512, 1024))
input_data = convolutional(input_data, (1, 1, 1024, 512))

max_pooling_1 = tf.keras.layers.MaxPool2D(pool_size=13, padding='SAME', strides=1)(input_data)
max_pooling_2 = tf.keras.layers.MaxPool2D(pool_size=9, padding='SAME', strides=1)(input_data)
max_pooling_3 = tf.keras.layers.MaxPool2D(pool_size=5, padding='SAME', strides=1)(input_data)
input_data = tf.concat([max_pooling_1, max_pooling_2, max_pooling_3, input_data], axis=-1)

input_data = convolutional(input_data, (1, 1, 2048, 512))
input_data = convolutional(input_data, (3, 3, 512, 1024))
input_data = convolutional(input_data, (1, 1, 1024, 512))

return route_1, route_2, input_data
```





SPP(Spatial Pyramid Pooling) layer

여러 개의 convolutional layer 통과 후 서로 다른 3가지 pool_size로 영역을 나눠

각 칸마다 MaxPool2D를 수행하여 가장 큰 값을 추출한 후 쭉 이어 concat 해줌

이어진 input_data를 다시 convolutional layer 통과하고 마지막으로 return

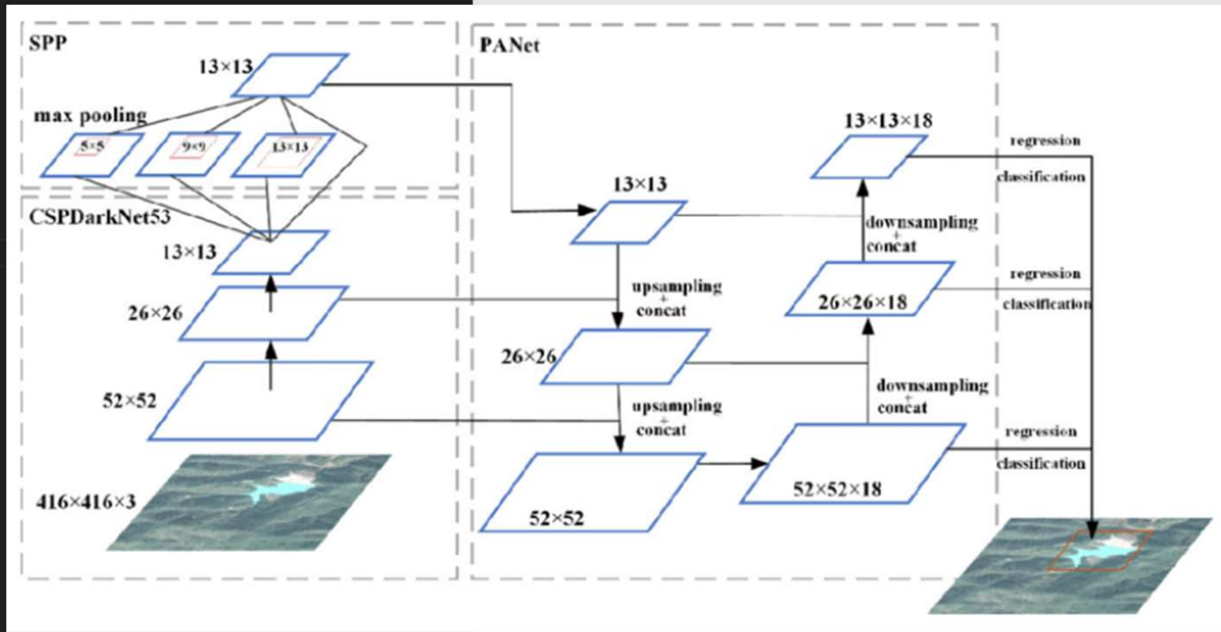
02.

NECK

PANet

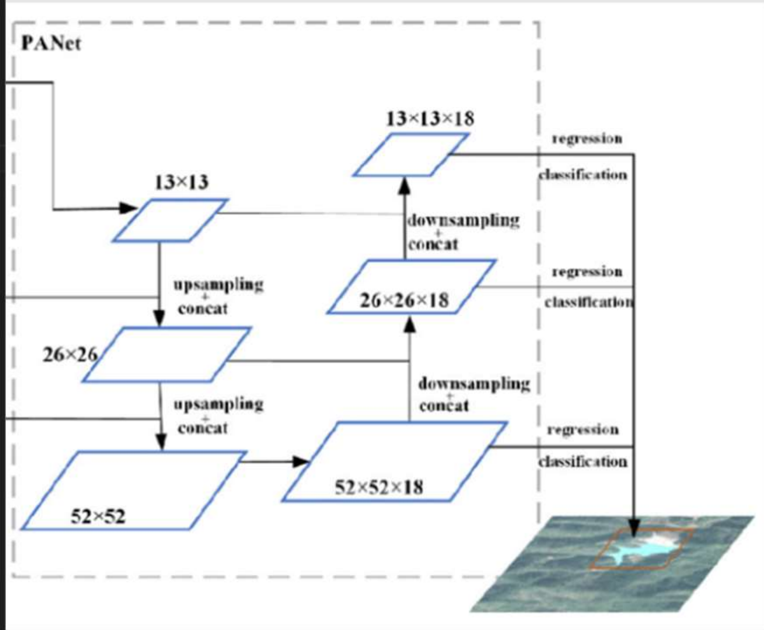
NECK (YOLOV4)

```
route_1, route_2, conv = cspdarknet53(input_layer)

route = conv
conv = convolutional(conv, (1, 1, 512, 256))
conv = upsample(conv)
route_2 = convolutional(route_2, (1, 1, 512, 256))
conv = tf.concat([route_2, conv], axis=-1)

conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))

route_2 = conv
conv = convolutional(conv, (1, 1, 256, 128))
conv = upsample(conv)
route_1 = convolutional(route_1, (1, 1, 256, 128))
conv = tf.concat([route_1, conv], axis=-1)

conv = convolutional(conv, (1, 1, 256, 128))
conv = convolutional(conv, (3, 3, 128, 256))
conv = convolutional(conv, (1, 1, 256, 128))
conv = convolutional(conv, (3, 3, 128, 256))
conv = convolutional(conv, (1, 1, 256, 128))

route_1 = conv
conv = convolutional(conv, (3, 3, 128, 256))
conv_sbbox = convolutional(conv, (1, 1, 256, 3 * (NUM_CLASS + 5)), activate=False, bn=False)
```



NECK (YOLOV4)

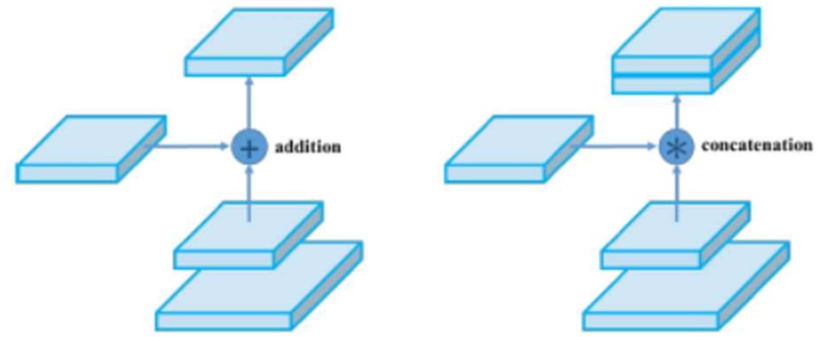
```
conv = convolutional(route_1, (3, 3, 128, 256), downsample=True)
conv = tf.concat([conv, route_2], axis=-1)

conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))

route_2 = conv
conv = convolutional(conv, (3, 3, 256, 512))
conv_mbbox = convolutional(conv, (1, 1, 512, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

conv = convolutional(route_2, (3, 3, 256, 512), downsample=True)
conv = tf.concat([conv, route], axis=-1)

conv = convolutional(conv, (1, 1, 1024, 512))
conv = convolutional(conv, (3, 3, 512, 1024))
conv = convolutional(conv, (1, 1, 1024, 512))
conv = convolutional(conv, (3, 3, 512, 1024))
conv = convolutional(conv, (1, 1, 1024, 512))

conv = convolutional(conv, (3, 3, 512, 1024))
conv_lbbox = convolutional(conv, (1, 1, 1024, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

return [conv_sbbox, conv_mbbox, conv_lbbox]
```



NECK (YOLOV4)

```
route_1, route_2, conv = cspdarknet53(input_layer)

route = conv
conv = convolutional(conv, (1, 1, 512, 256))
conv = upsample(conv)
route_2 = convolutional(route_2, (1, 1, 512, 256))
conv = tf.concat([route_2, conv], axis=-1)

conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))
conv = convolutional(conv, (3, 3, 256, 512))
conv = convolutional(conv, (1, 1, 512, 256))

route_2 = conv
conv = convolutional(conv, (1, 1, 256, 128))
conv = upsample(conv)
route_1 = convolutional(route_1, (1, 1, 256, 128))
conv = tf.concat([route_1, conv], axis=-1)

conv = convolutional(conv, (1, 1, 256, 128))
conv = convolutional(conv, (3, 3, 128, 256))
conv = convolutional(conv, (1, 1, 256, 128))
conv = convolutional(conv, (3, 3, 128, 256))
conv = convolutional(conv, (1, 1, 256, 128))

route_1 = conv
conv = convolutional(conv, (3, 3, 128, 256))
conv_sbbox = convolutional(conv, (1, 1, 256, 3 * (NUM_CLASS + 5)), activate=False, bn=False)
```

```
return [conv_sbbox, conv_mbbox, conv_lbbox]
```



(a) PAN [49]

(a) Our modified PAN

/ (AI)

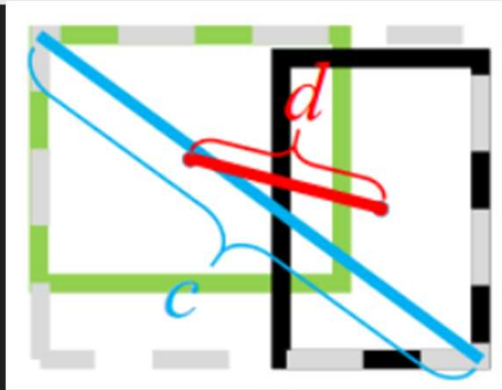
03.

LOSS FUNCTION

Bbox_ciou

LOSS FUNCTION

```
boxes1_coor = tf.concat([boxes1[..., :2] - boxes1[..., 2:] * 0.5,
                         boxes1[..., :2] + boxes1[..., 2:] * 0.5], axis=-1)
boxes2_coor = tf.concat([boxes2[..., :2] - boxes2[..., 2:] * 0.5,
                         boxes2[..., :2] + boxes2[..., 2:] * 0.5], axis=-1)

left = tf.maximum(boxes1_coor[..., 0], boxes2_coor[..., 0])
up = tf.maximum(boxes1_coor[..., 1], boxes2_coor[..., 1])
right = tf.maximum(boxes1_coor[..., 2], boxes2_coor[..., 2])
down = tf.maximum(boxes1_coor[..., 3], boxes2_coor[..., 3])

c = (right - left) * (right - left) + (up - down) * (up - down)
```



$$d = \rho(\mathbf{b}, \mathbf{b}^{gt})$$

```
u = (boxes1[..., 0] - boxes2[..., 0]) * (boxes1[..., 0] - boxes2[..., 0]) +
d = u / c
```

$$\frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$

boxes들에 대해 [x,y], [w,h] 식으로 나누어 왼쪽 위 오른쪽 아래 지점으로 나누어 concat한 후 boxes1_coor에 저장함
위에서 구한 boxes1과 boxes2의 coor변수의 저장된 값들을 비교해 큰 값을 가져와 가장 큰 대각선 길이 c의 제곱을 구함
u는 두 박스 중심 사이의 거리를 구해준 후, u/c를 d변수에 저장

LOSS FUNCTION

```
ar_gt = boxes2[..., 2] / boxes2[..., 3]
ar_pred = boxes1[..., 2] / boxes1[..., 3]

ar_loss = 4 / (np.pi * np.pi) * (tf.atan(ar_gt) - tf.atan(ar_pred)) *
alpha = ar_loss / (1 - iou + ar_loss + 0.000001)
ciou_term = d + alpha * ar_loss

return iou - ciou_term
```

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2. \quad \alpha = \frac{v}{(1 - IoU) + v},$$

v를 구해주기 위해 예측한 w/h와 GT w/h 값을 ar_pred, ar_gt에 넣어줌
이를 위에 보이는 식으로 대입해 ar_loss를 작성하여 v 얻음
a를 구해주기 위해 위의 식을 alpha에 대입
마지막으로 앞장에서 구한 d와 a*v를 더해주면서 ciou_term을 얻음

$$1 - \mathcal{L}_{CIoU} = IoU - \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} - \alpha v$$

최종적인 CIoU Loss return 값

ARTIFICIAL

INTE

(AI)

THANKS A LOT FOR LISTENING

(AI)