Computer vision tasks and corresponding NNs

Alex Net VGG16 Res Net	U Net	Yolo Faster RCNN	Mask RCNN	OpenPose Keypoints RCNN

Classification

Semantic Segmentation

GRASS, CAT, TREE, SKY

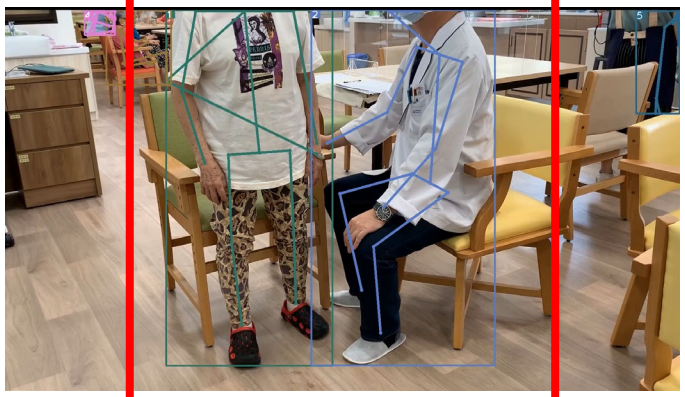
Object Detection

DOG, DOG, CAT

Instance Segmentation

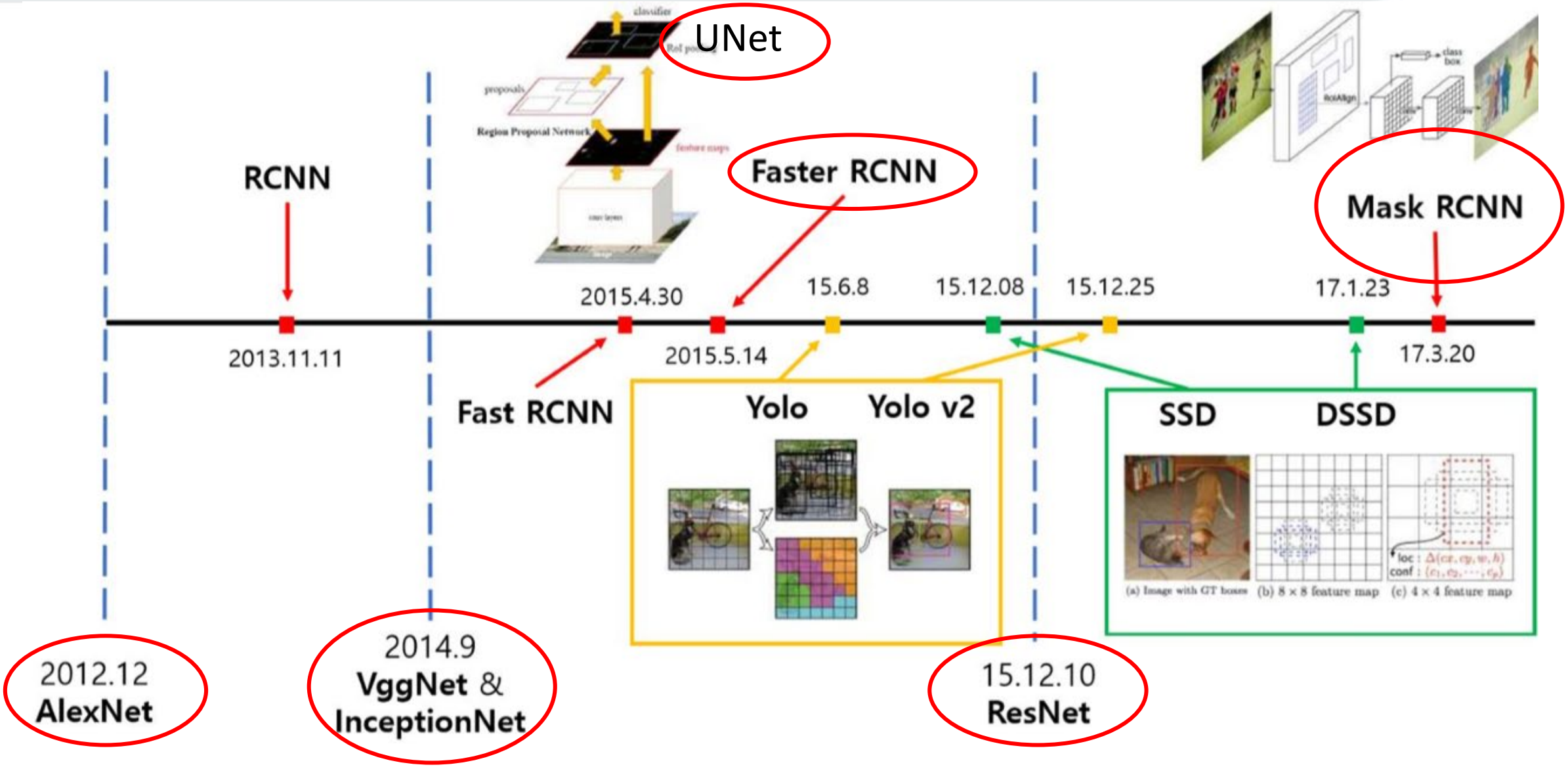
DOG, DOG, CAT

Joint detection



圖片來源: https://kharshit.github.io/blog/2019/08/23/quick-intro-to-instance-segmentation

CNN families for CV tasks



圖來源: 李春煌 FasterRCNN講義 https://youtu.be/2i9CcmJp2yI

MaskRCNN

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick

Facebook AI Research (FAIR)

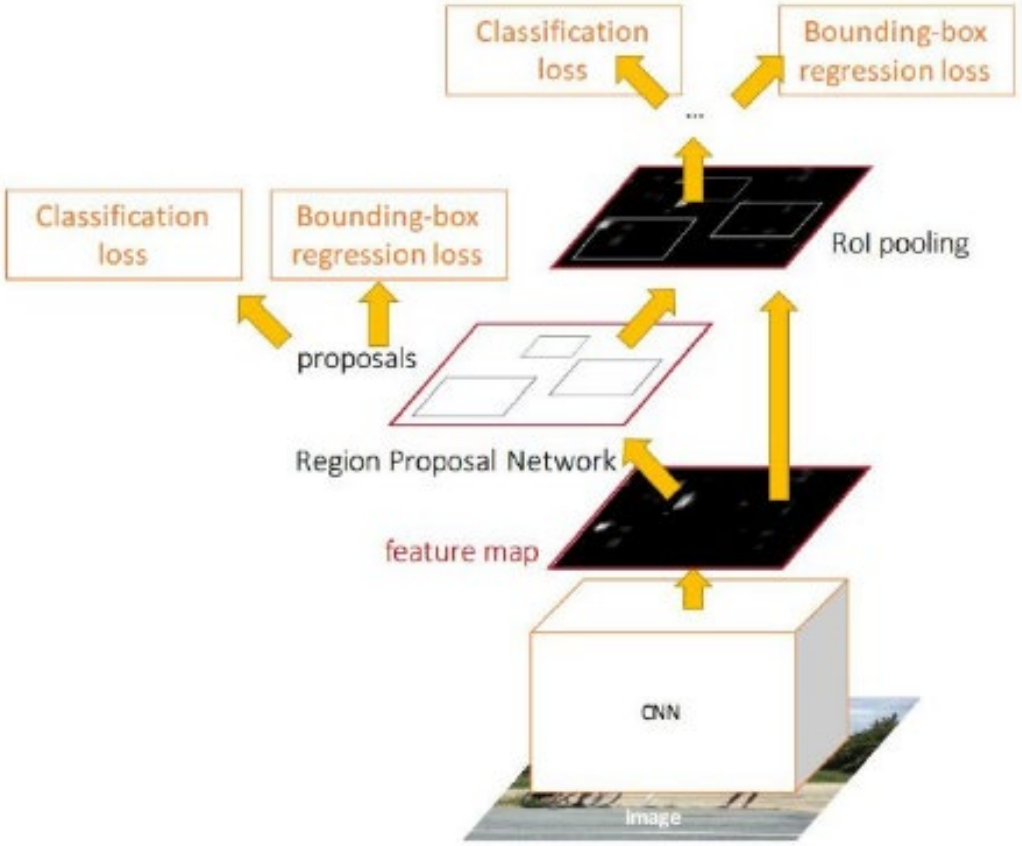
He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

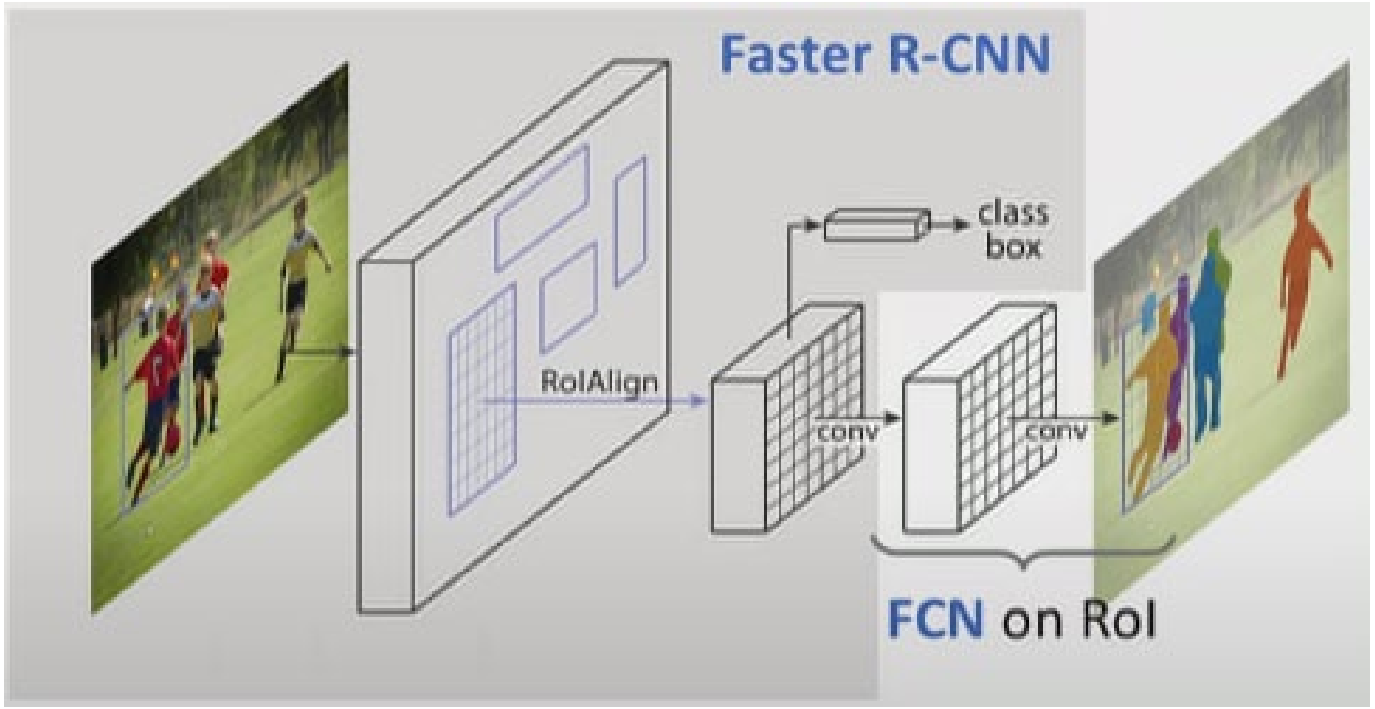
1703.06870.pdf (arxiv.org)

Class practice

MaskRCNN detect person.ipynb

MaskRCNN = FasterRCNN with FCN on ROIs

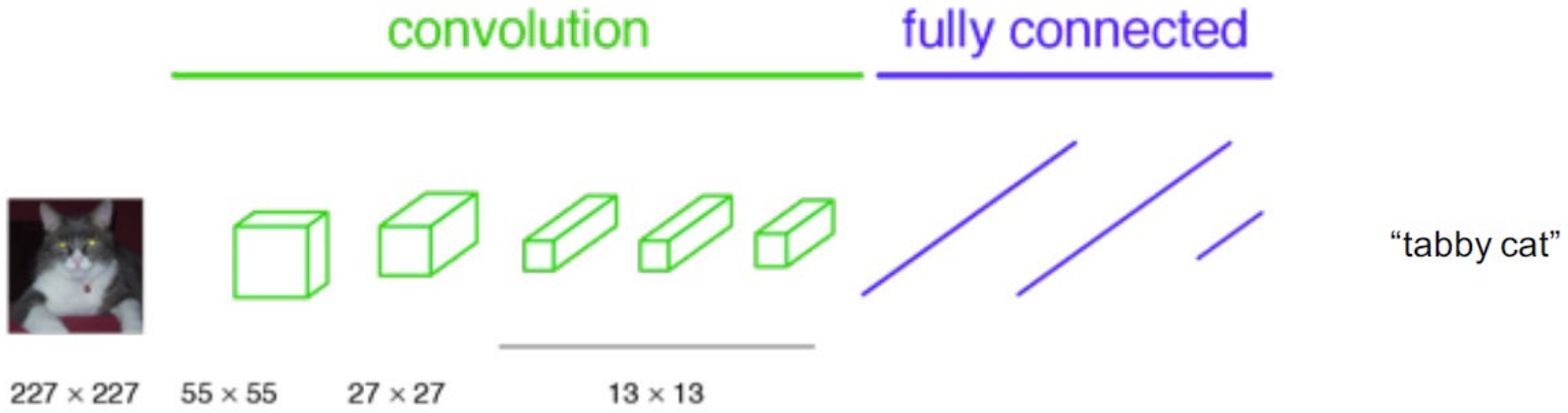




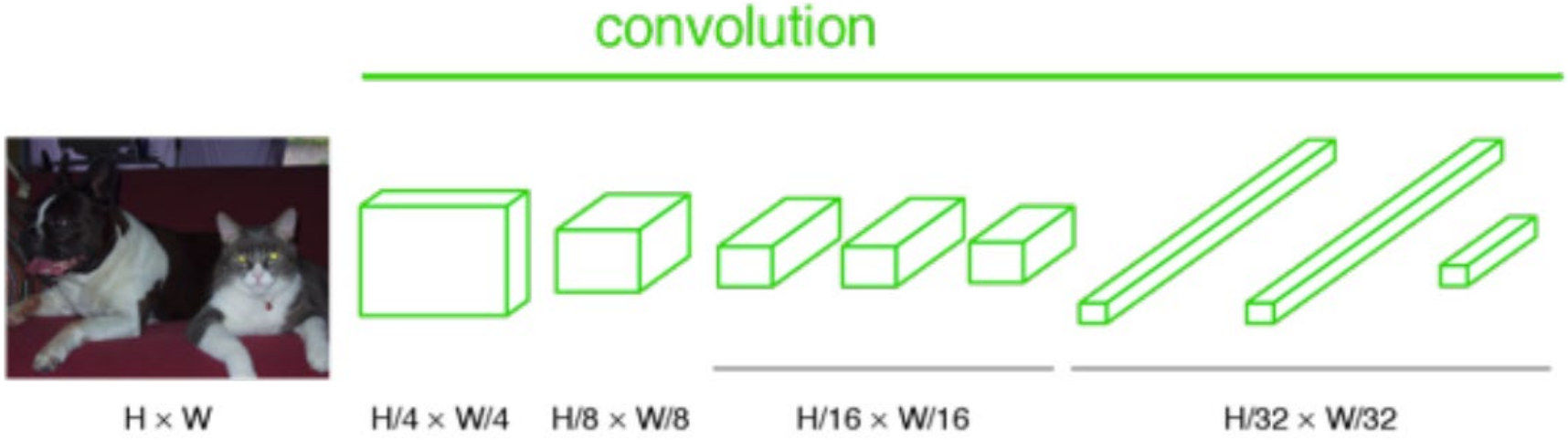
FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

https://youtu.be/g7z4mkfRjI4

CNN vs FCN (Fully Convolutional Network)



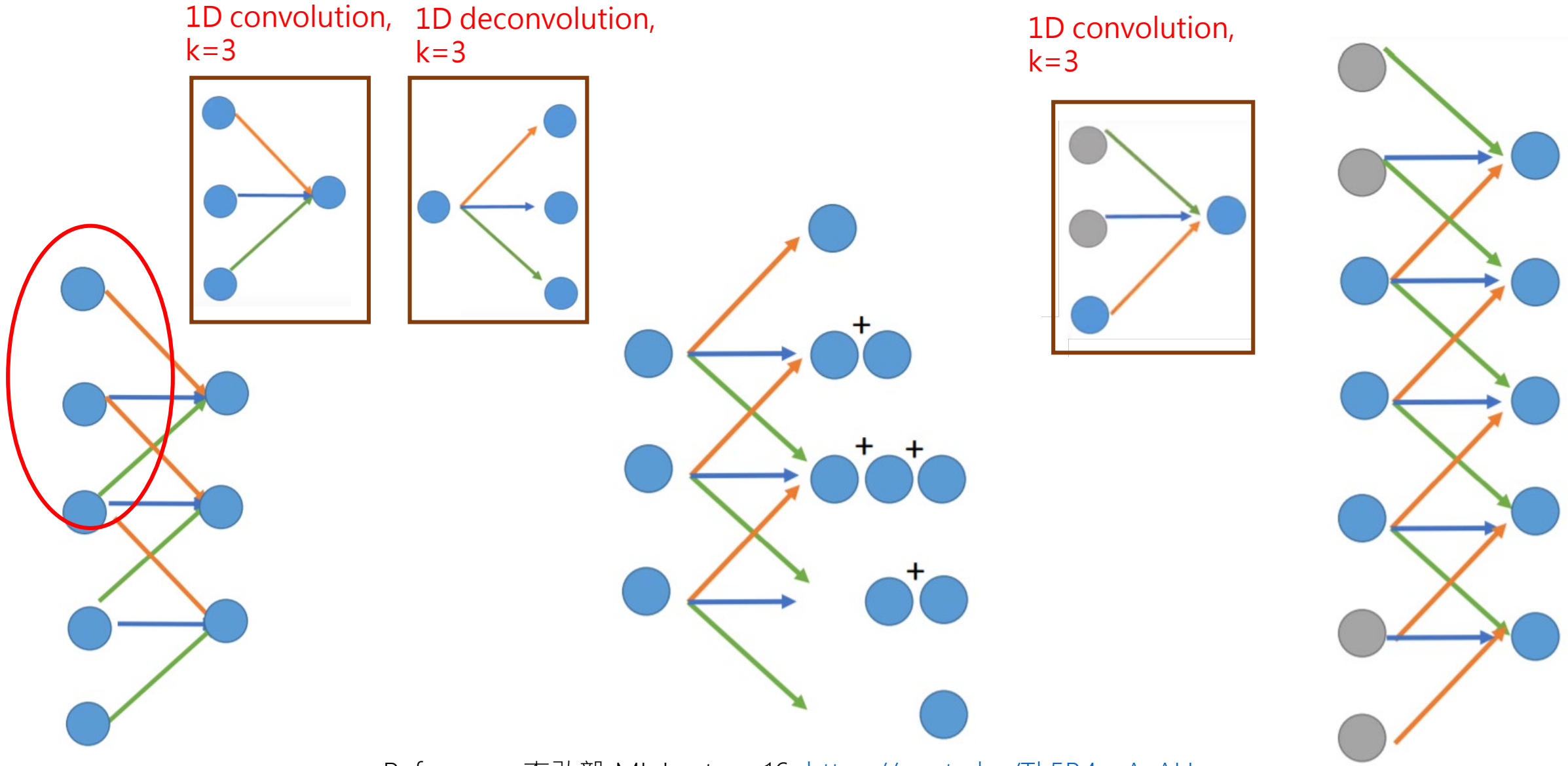
CNN for image classification



FCN – all layers are convolutional layers

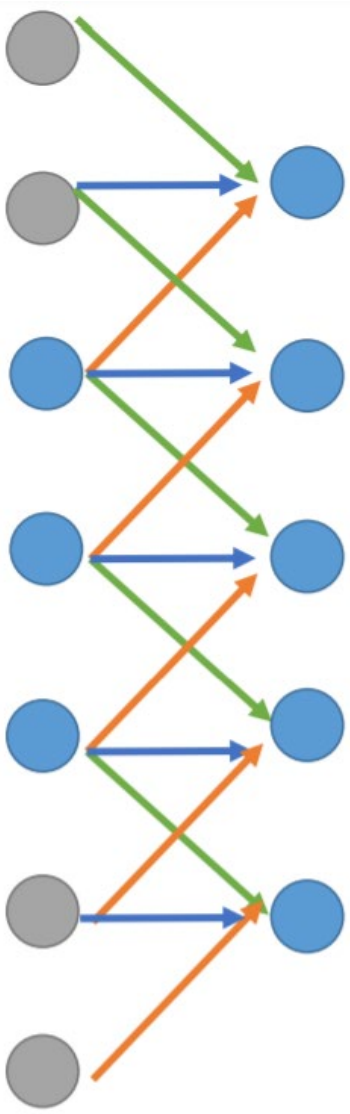
https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1

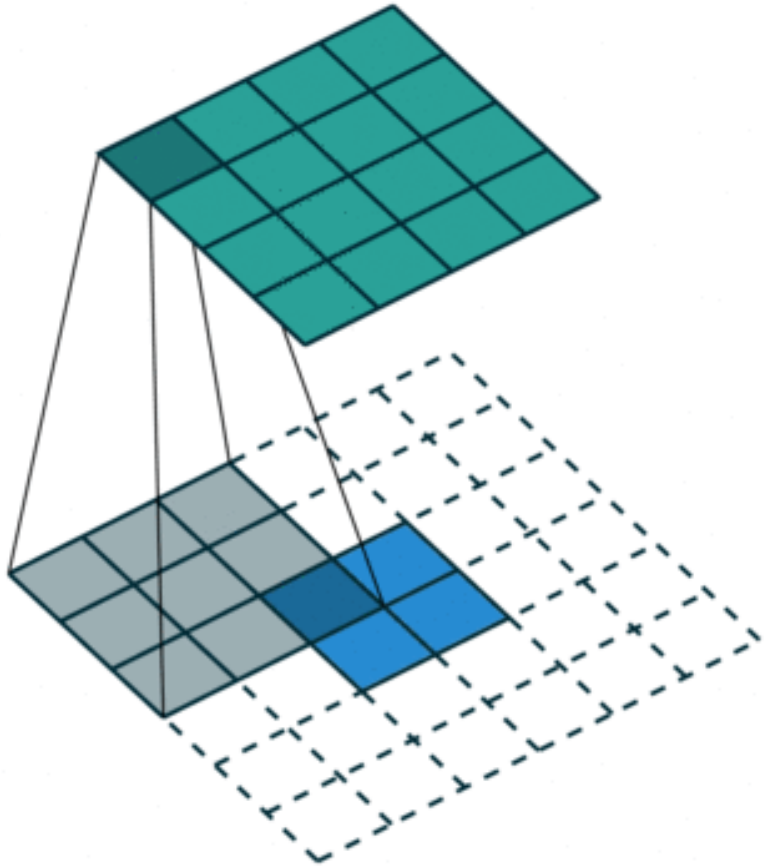
Up Sampling – 1D deconvolution



Reference: 李弘毅 ML Lecture 16 https://youtu.be/Tk5B4seA-AU

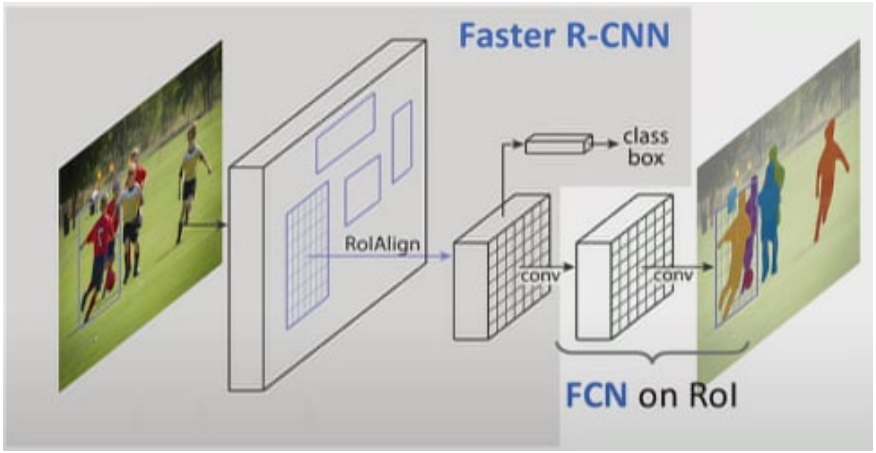
Up Sampling – 2D deconvolution





https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1

PyTorch implementation of FCN on ROIs



Convolution down sampling

Deconvolution up sampling

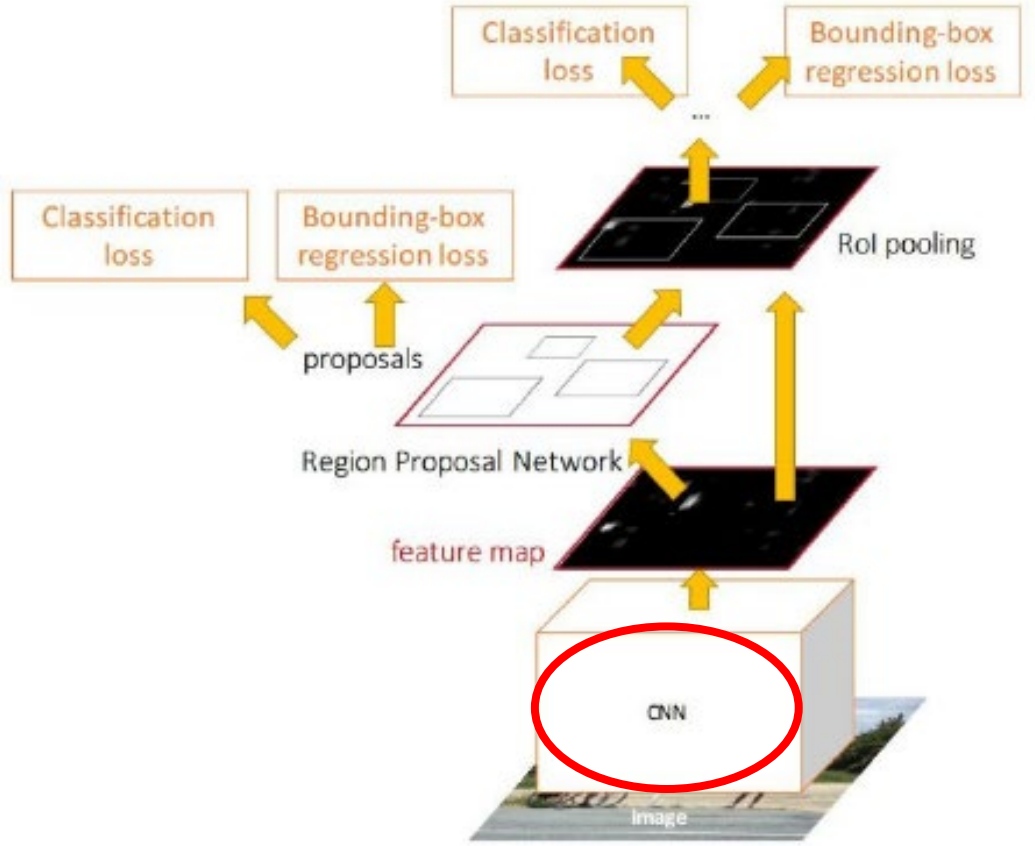
```
(mask_head): MaskRCNNHeads(
  (mask_fcn1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1
  (relu1): ReLU(inplace=True)
  (mask_fcn2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1
  (relu2): ReLU(inplace=True)
  (mask_fcn3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1
  (relu3): ReLU(inplace=True)
  (mask_fcn4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1
  (relu4): ReLU(inplace=True)
)
(mask_predictor): MaskRCNNPredictor(
  (conv5_mask): ConvTranspose2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
  (relu): ReLU(inplace=True)
  (mask_fcn_logits): Conv2d(256, 91, kernel_size=(1, 1), stride=(1, 1))
```

FPN vs CNN

MaskRCNN detect person.ipynb

```
(backbone): BackboneWithFPN(
  (body): IntermediateLayerGetter(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2,
    (bn1): FrozenBatchNorm2d(64, eps=0.0)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stric
        (bn1): FrozenBatchNorm2d(64, eps=0.0)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stric
        (bn2): FrozenBatchNorm2d(64, eps=0.0)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stri
        (bn3): FrozenBatchNorm2d(256, eps=0.0)
        (relu): ReLU(inplace=True)
```

Compare with FasterRCNN



FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

Feature pyramid network (FPN)

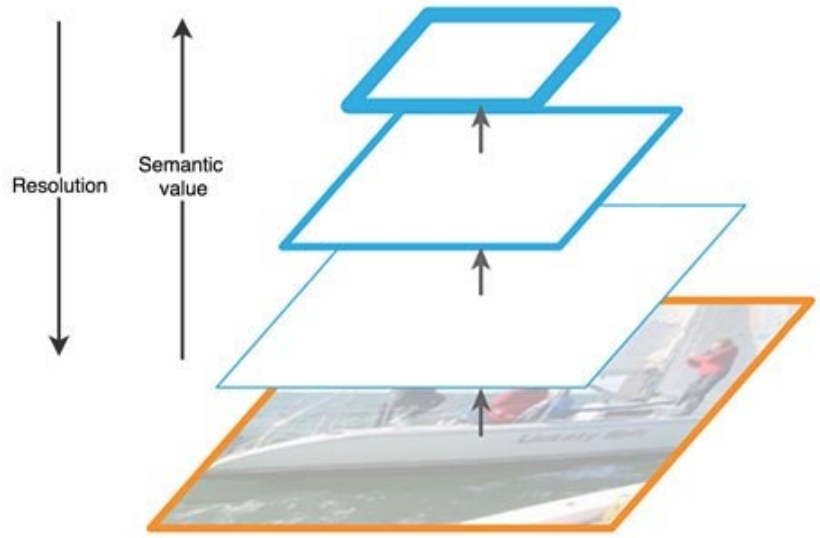
Feature Pyramid Networks for Object Detection

Tsung-Yi Lin[1,2], Piotr Dollár[1], Ross Girshick[1],
Kaiming He[1], Bharath Hariharan[1], and Serge Belongie[2]

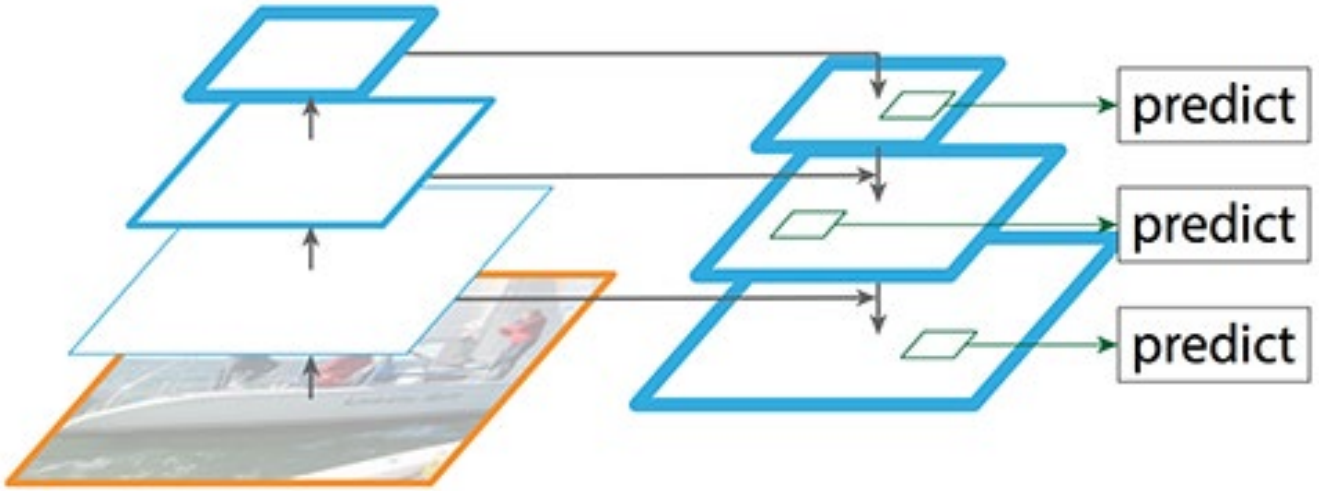
[1]Facebook AI Research (FAIR)
[2]Cornell University and Cornell Tech

https://arxiv.org/pdf/1612.03144.pdf

Feature pyramid network (FPN)



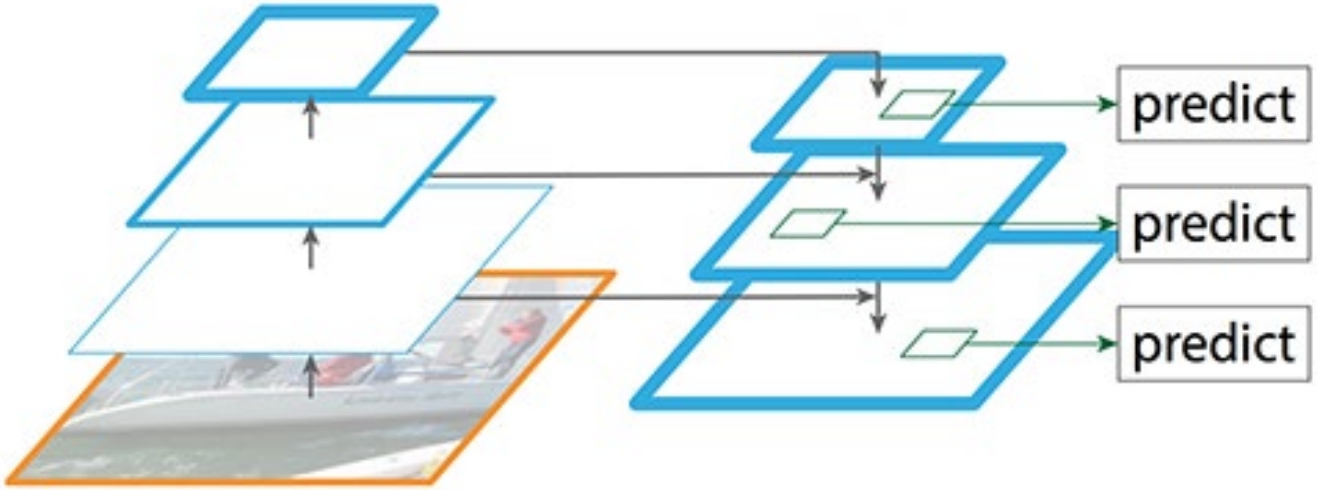
Bottom up pathway – Usual CNN for feature extraction (spatial resolution decreases, semantic value increases)



Top down pathway to construct higher resolution layers from semantic rich layer.

https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

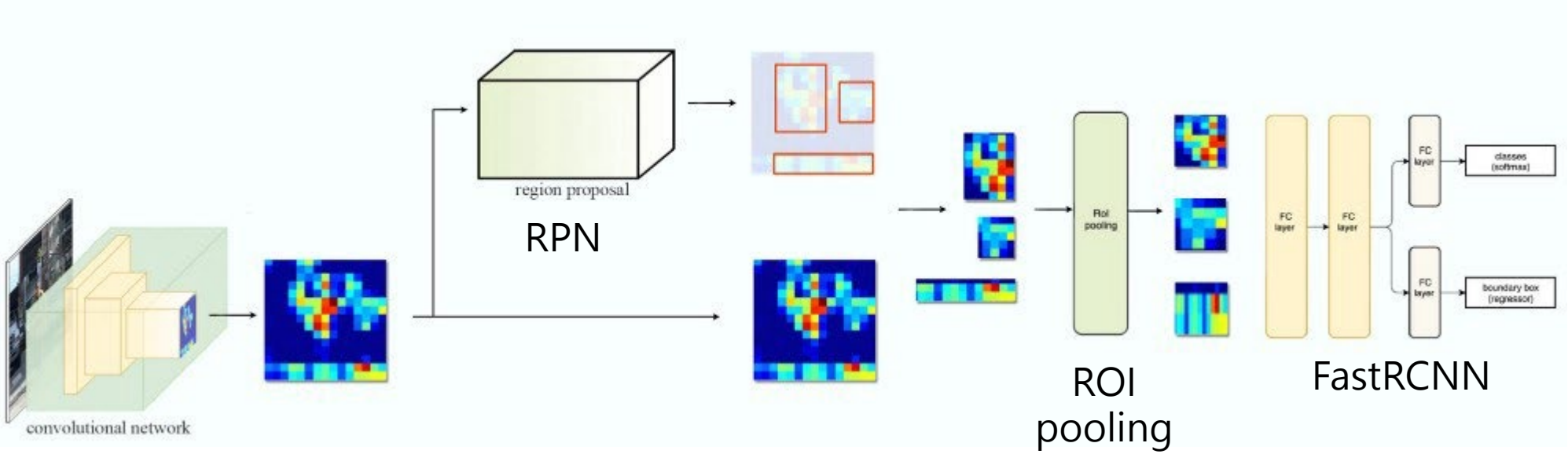
Feature pyramid network (FPN)



The locations of objects are not precise after all the down sampling and up sampling. Lateral connections between reconstructed layers and the corresponding feature maps are added to help the detector to predict the location betters. It also acts as skip connections to make training easier (similar to what ResNet does).

https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

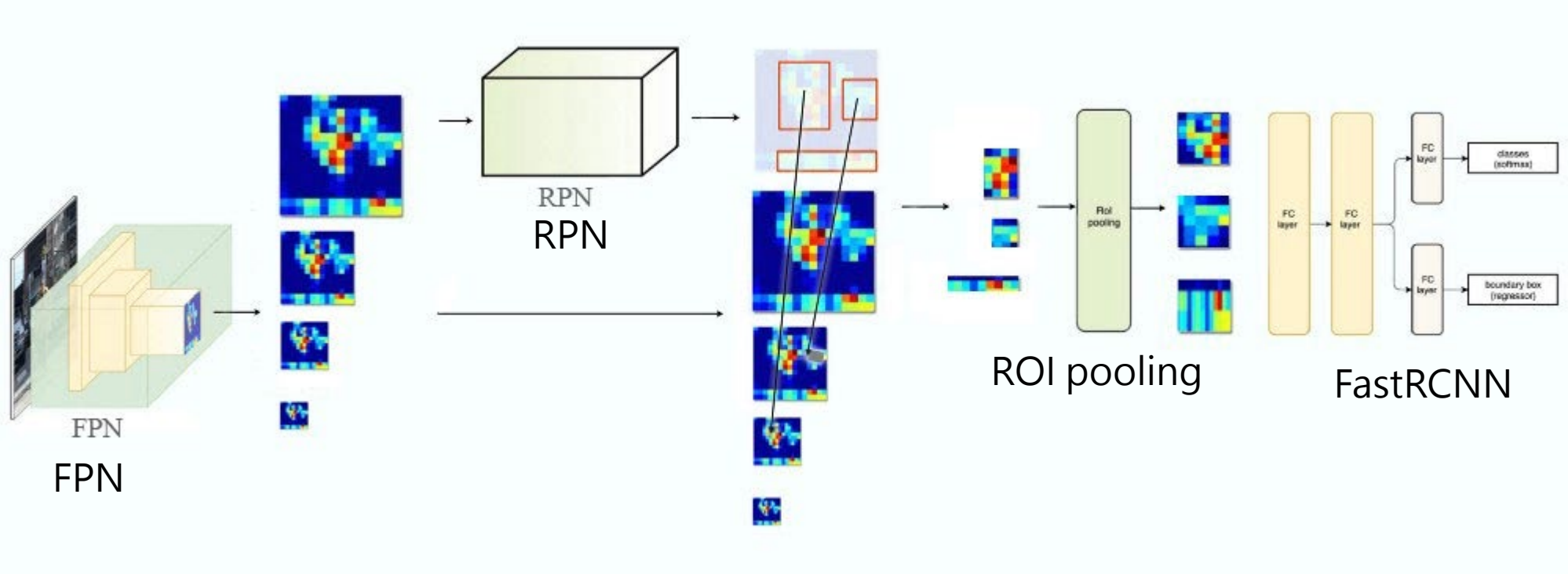
Original FasterRCNN



FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

FasterRCNN with FPN



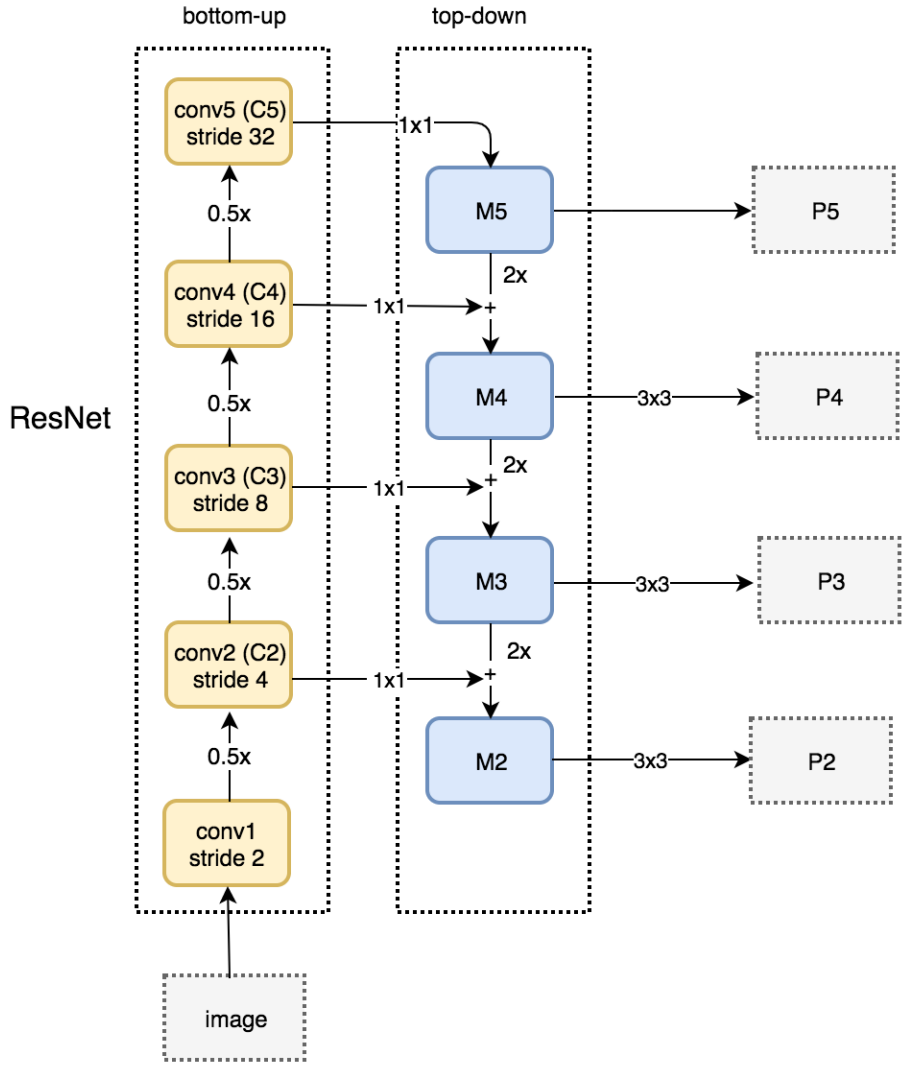
We apply the RPN to generate ROIs. Based on the size of the ROI, we select the feature map layer in the most proper scale to extract the feature patches.

https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

PyTorch implementation of FPN

MaskRCNN detect person.ipynb

```
(fpn): FeaturePyramidNetwork(
  (inner_blocks): ModuleList(
    (0): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
    (2): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
    (3): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
  )
  (layer_blocks): ModuleList(
    (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
    (1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
    (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
    (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
  )
  (extra_blocks): LastLevelMaxPool()
)
```



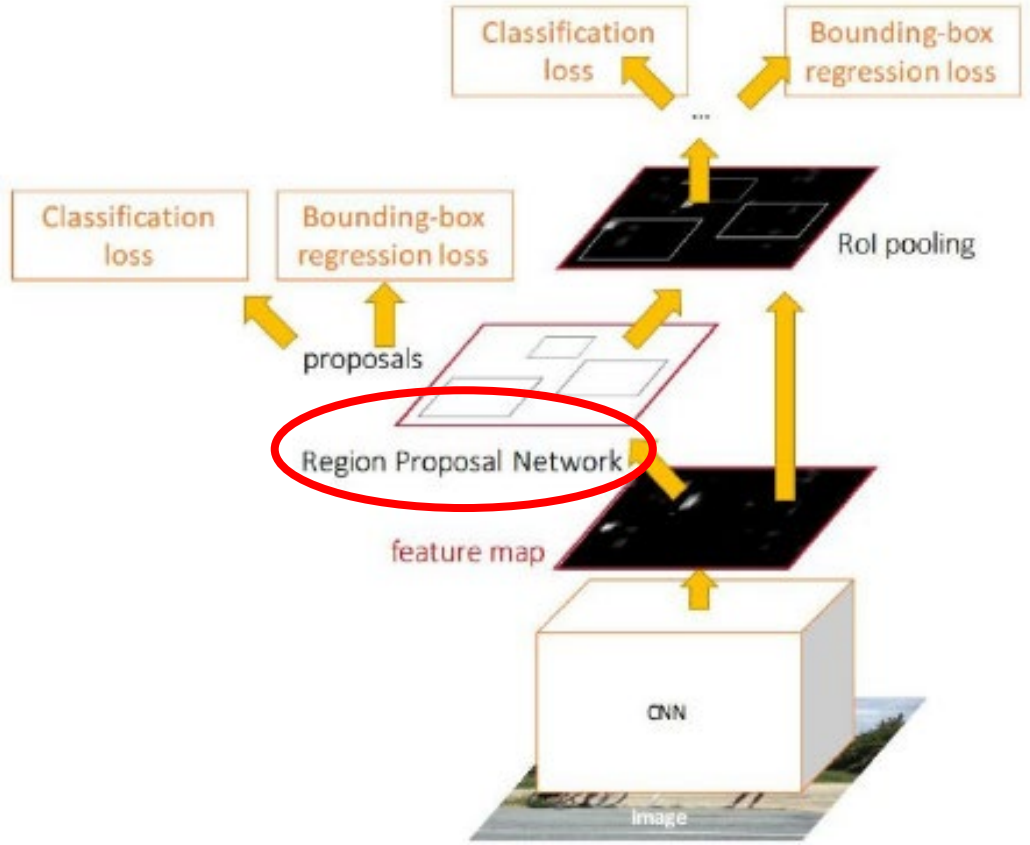
https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

RPN

MaskRCNN detect person.ipynb

```
(rpn): RegionProposalNetwork(
  (anchor_generator): AnchorGenerator()
  (head): RPNHead(
    (conv): Conv2d(256, 256, kernel_size=(3, 3), stri
    (cls_logits): Conv2d(256, 3, kernel_size=(1, 1),
    (bbox_pred): Conv2d(256, 12, kernel_size=(1, 1),
  )
)
```

Compare with FasterRCNN



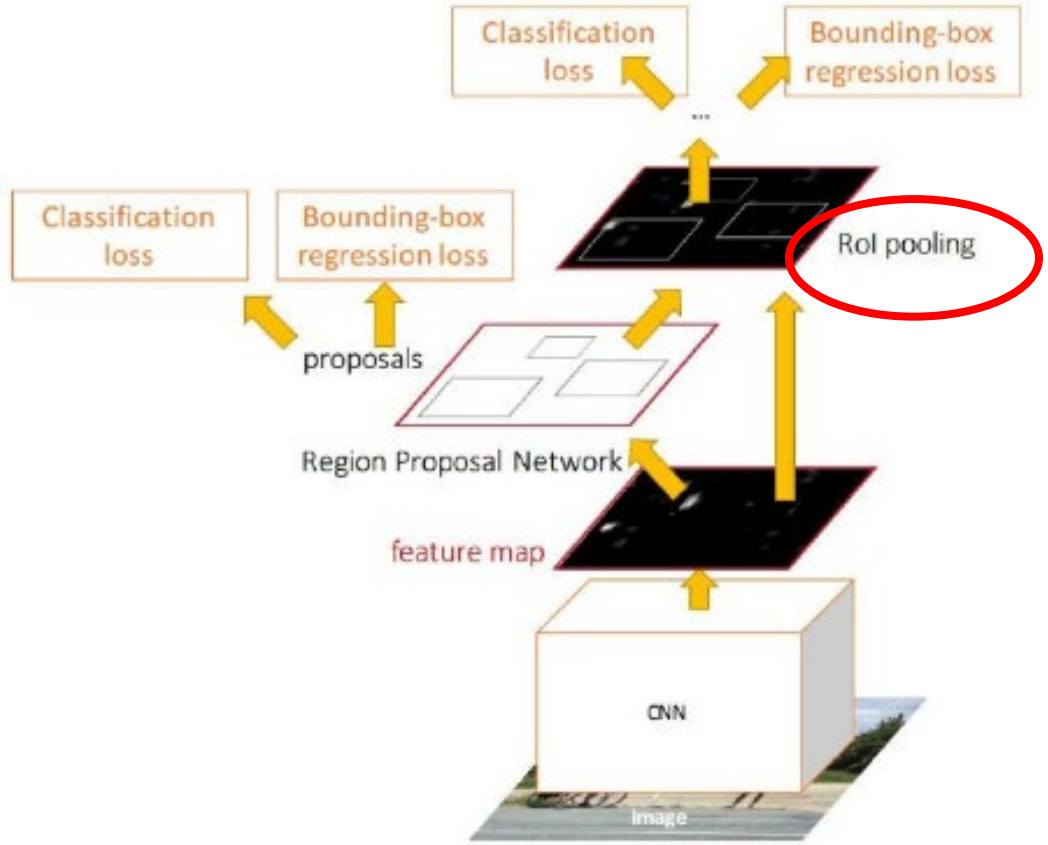
FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

ROI align vs ROI pooling

MaskRCNN detect person.ipynb

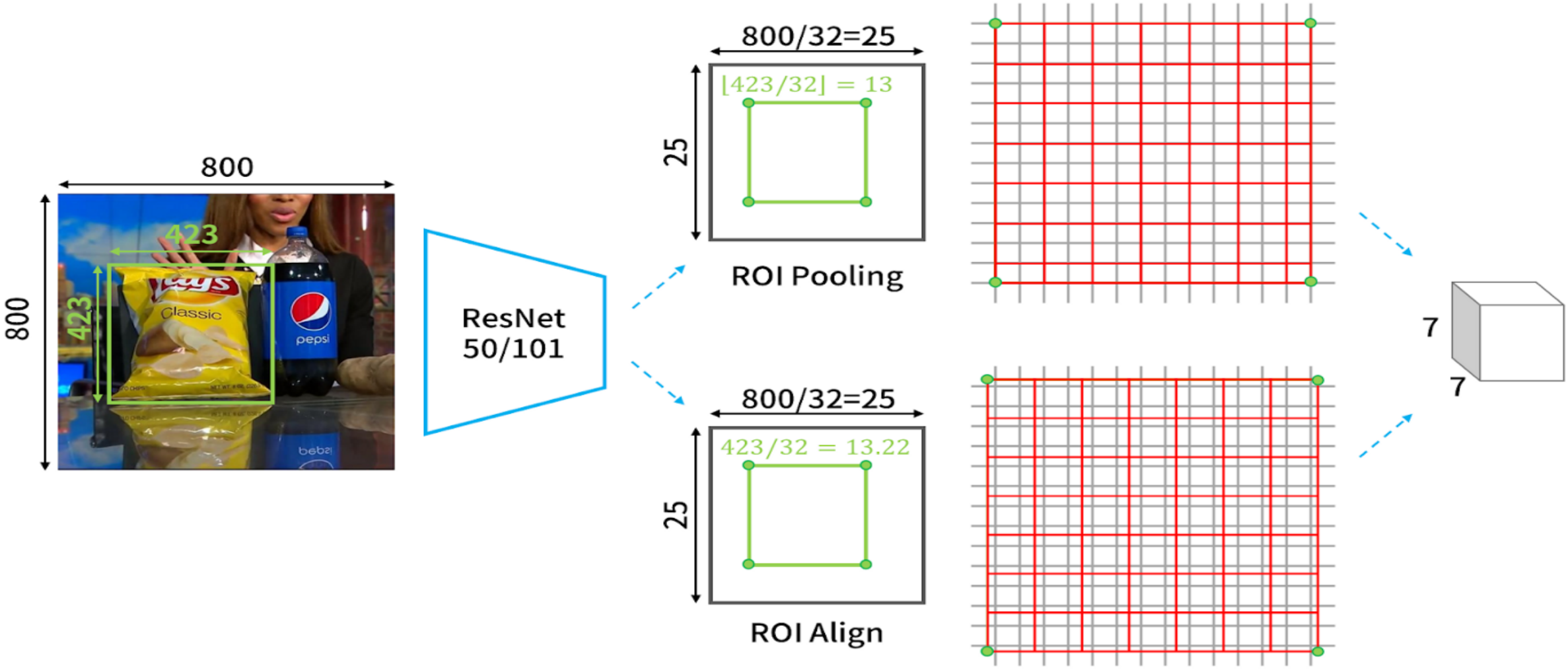
```
(mask_roi_pool): MultiScaleRoIAlign(featmap_names=['0', '1
(mask_head): MaskRCNNHeads(
  (mask_fcn1): Conv2d(256, 256, kernel_size=(3, 3), stride
  (relu1): ReLU(inplace=True)
  (mask_fcn2): Conv2d(256, 256, kernel_size=(3, 3), stride
  (relu2): ReLU(inplace=True)
  (mask_fcn3): Conv2d(256, 256, kernel_size=(3, 3), stride
  (relu3): ReLU(inplace=True)
  (mask_fcn4): Conv2d(256, 256, kernel_size=(3, 3), stride
  (relu4): ReLU(inplace=True)
```

Compare with FasterRCNN



FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

ROI align to reduce rounding error problem



https://ivan-eng-murmur.medium.com/%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%AC-s9-mask-r-cnn-%E7%B0%A1%E4%BB%8B-99370c98de28

ROI align

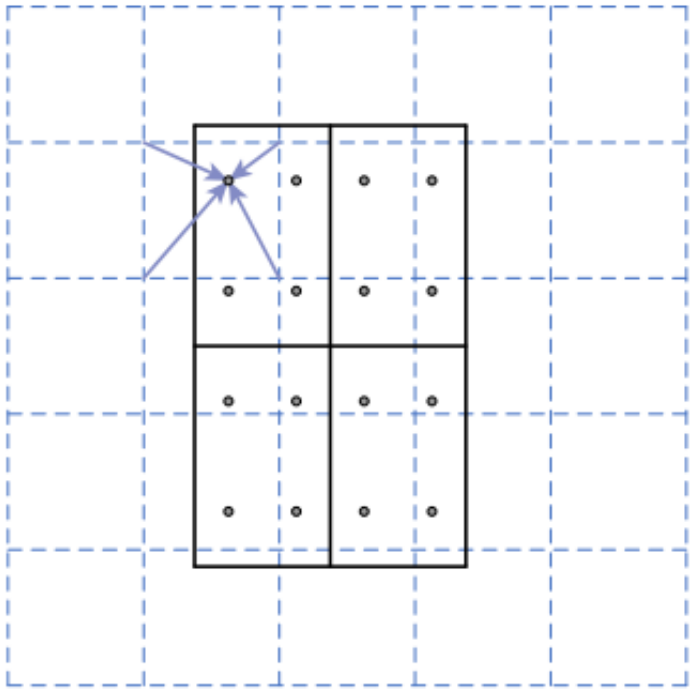


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

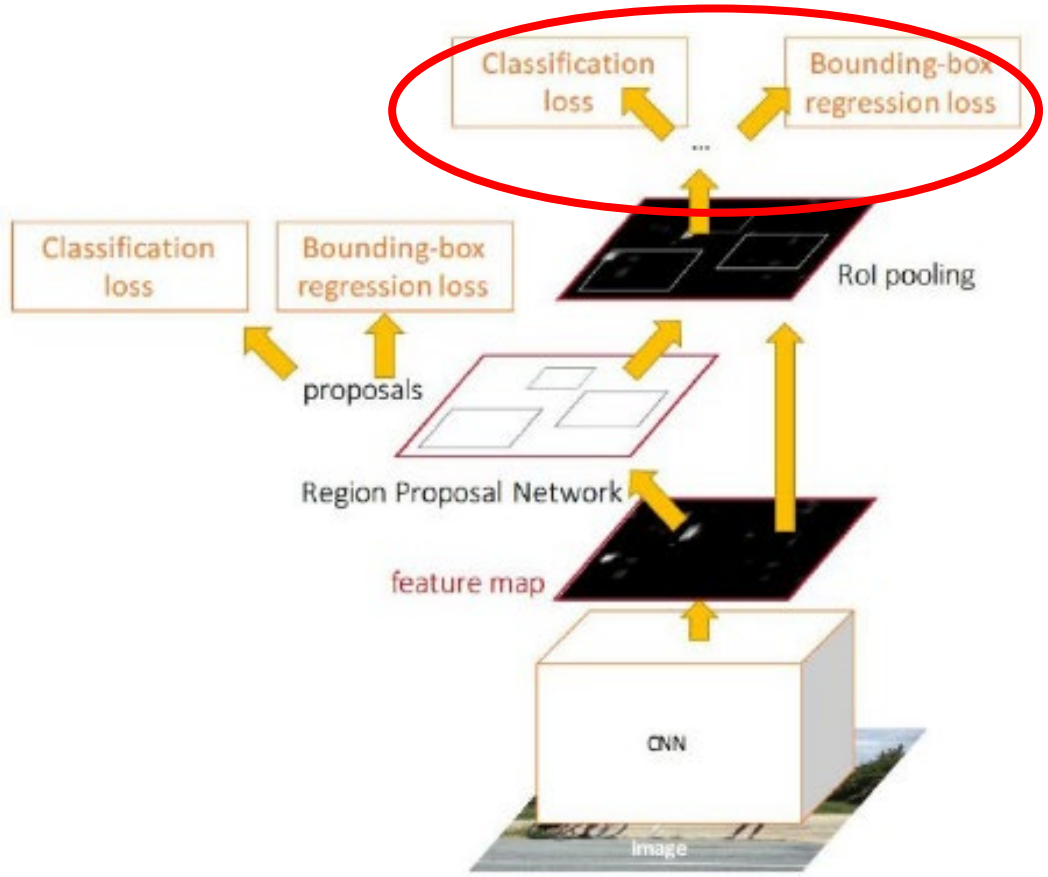
https://arxiv.org/pdf/1703.06870.pdf

FastRCNN predictor

MaskRCNN detect person.ipynb

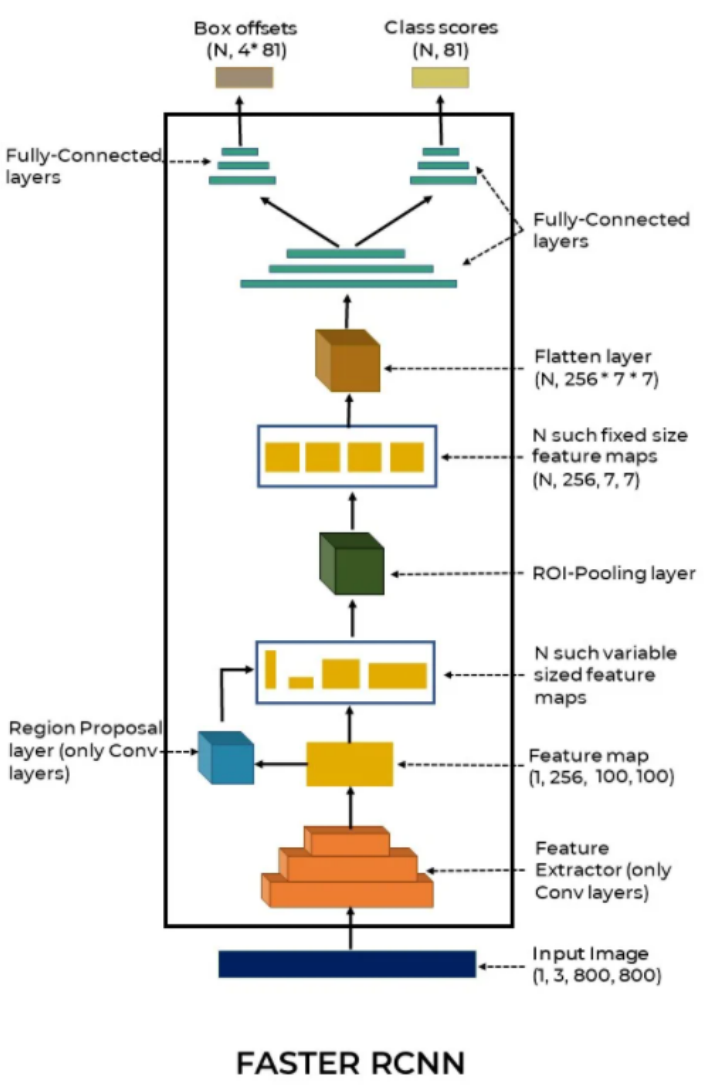
```
,
(roi_heads): RoIHeads(
  (box_roi_pool): MultiScaleRoIAlign(featmap_names=[
  (box_head): TwoMLPHead(
    (fc6): Linear(in_features=12544, out_features=10
    (fc7): Linear(in_features=1024, out_features=102
  )
  (box_predictor): FastRCNNPredictor(
    (cls_score): Linear(in_features=1024, out_featur
    (bbox_pred): Linear(in_features=1024, out_featur
  )
```

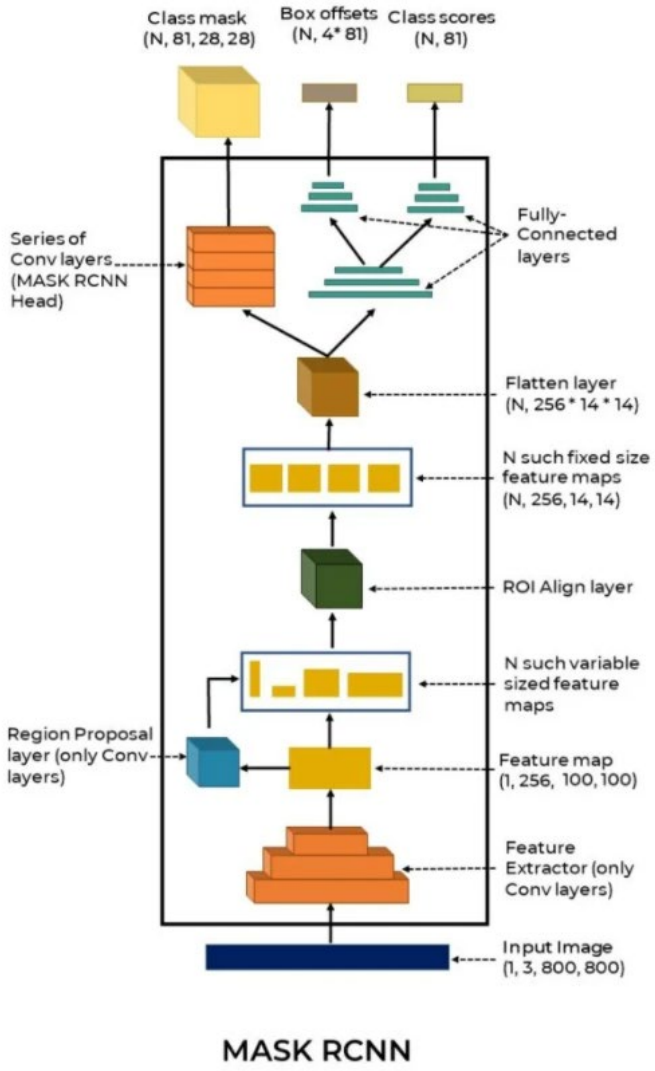
Compare with FasterRCNN

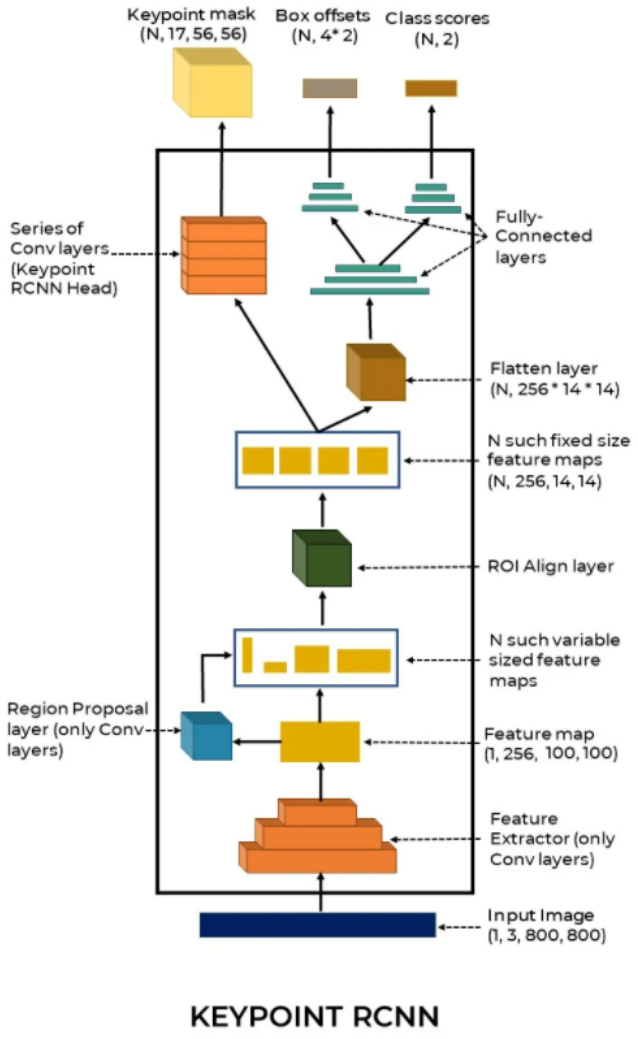


FasterRCNN = CNN backbone + RPN + ROI pooling + FastRCNN detector

FasterRCNN, MaskRCNN and KeypointsRCNN







https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/?ck_subscriber_id=297191382

Class practice

Keypoints RCNN.ipynb

Backbone with FPN

```
(backbone): BackboneWithFPN(
  (body): IntermediateLayerGetter(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding
    (bn1): FrozenBatchNorm2d(64)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bi
        (bn1): FrozenBatchNorm2d(64)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), pa
        (bn2): FrozenBatchNorm2d(64)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), b
```

RPN

```
,
(rpn): RegionProposalNetwork(
  (anchor_generator): AnchorGenerator()
  (head): RPNHead(
    (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (cls_logits): Conv2d(256, 3, kernel_size=(1, 1), stride=(1, 1))
    (bbox_pred): Conv2d(256, 12, kernel_size=(1, 1), stride=(1, 1))
  )
)
```

FastRCNN predictor

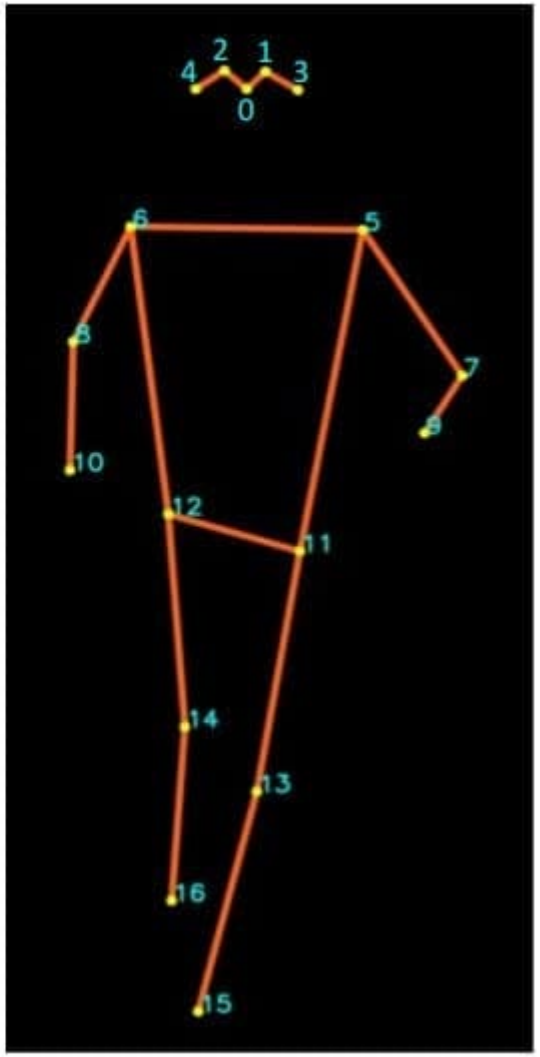
```
(roi_heads): RoIHeads(
  (box_roi_pool): MultiScaleRoIAlign()
  (box_head): TwoMLPHead(
    (fc6): Linear(in_features=12544, out_features=1024, bias=True)
    (fc7): Linear(in_features=1024, out_features=1024, bias=True)
  )
  (box_predictor): FastRCNNPredictor(
    (cls_score): Linear(in_features=1024, out_features=2, bias=True)
    (bbox_pred): Linear(in_features=1024, out_features=8, bias=True)
  )
```

KeypointRCNN head

```
(keypoint_roi_pool): MultiScaleRoIAlign()
(keypoint_head): KeypointRCNNHeads(
  (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): ReLU(inplace=True)
  (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU(inplace=True)
  (8): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU(inplace=True)
  (10): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace=True)
)
(keypoint_predictor): KeypointRCNNPredictor(
  (kps_score_lowres): ConvTranspose2d(512, 17, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
)
```

Keypoints available in COCO-dataset for humans

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/?ck_subscriber_id=297191382