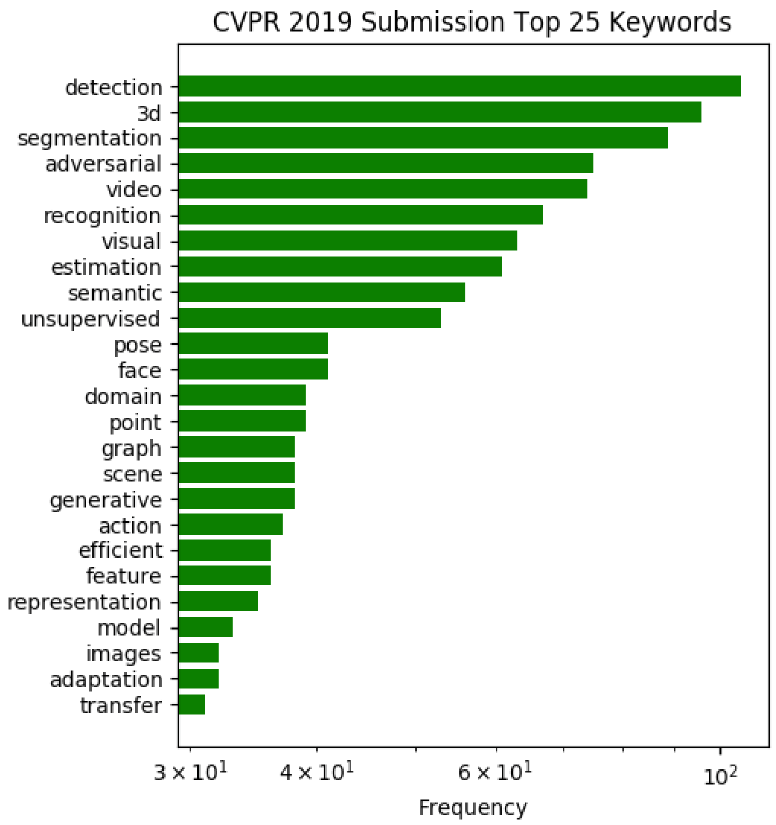
III. Anchor Free Methods

III. Other Methods

K. Trend

- Anchor free net is a trend 超参多
  - Lots of hyperparameters: sizes, aspect-ratios, number of anchors, ……
  - Hard to generalize: different datasets have different data shape, need to redesign 不好范化
  - Difficult to train: unbalanced positive / negative samples 类别不均衡
  - Complex calculation 计算复杂
  - Myriads of redundancy

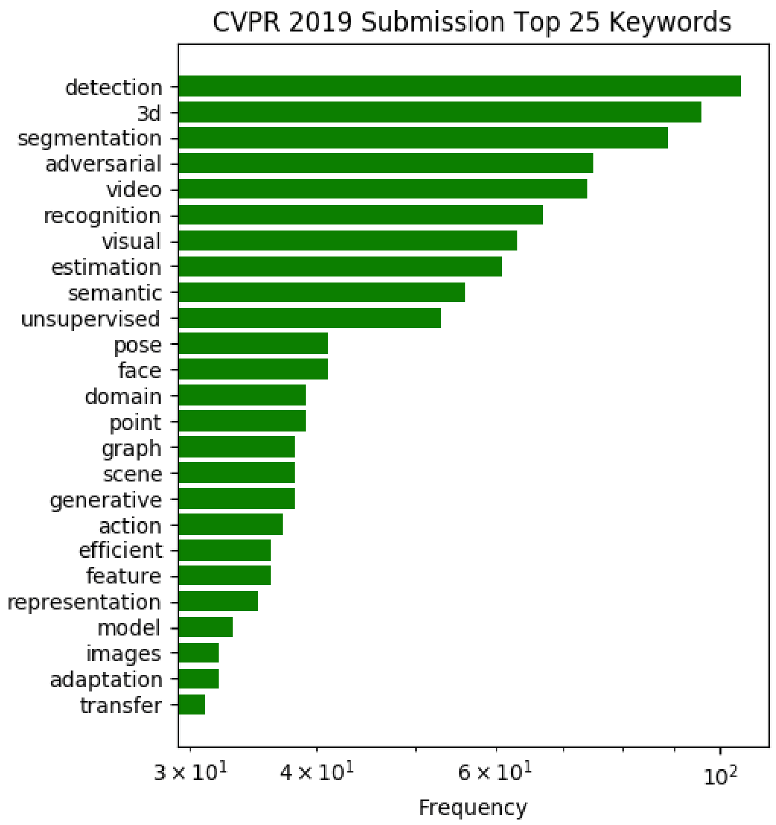


III. Other Methods

K. Trend

➢ Anchor free net is a trend

- CornerNet: 244, 03/2019 || CornerNet-Lite: 18, 04/2019
- FoveaBox: 29, 04/2019
- CenterNet: Objects as Points -78, 04/2019
  ~~Keypoint Triplets for Object Detection~~
- FCOS: 66, 08/2019

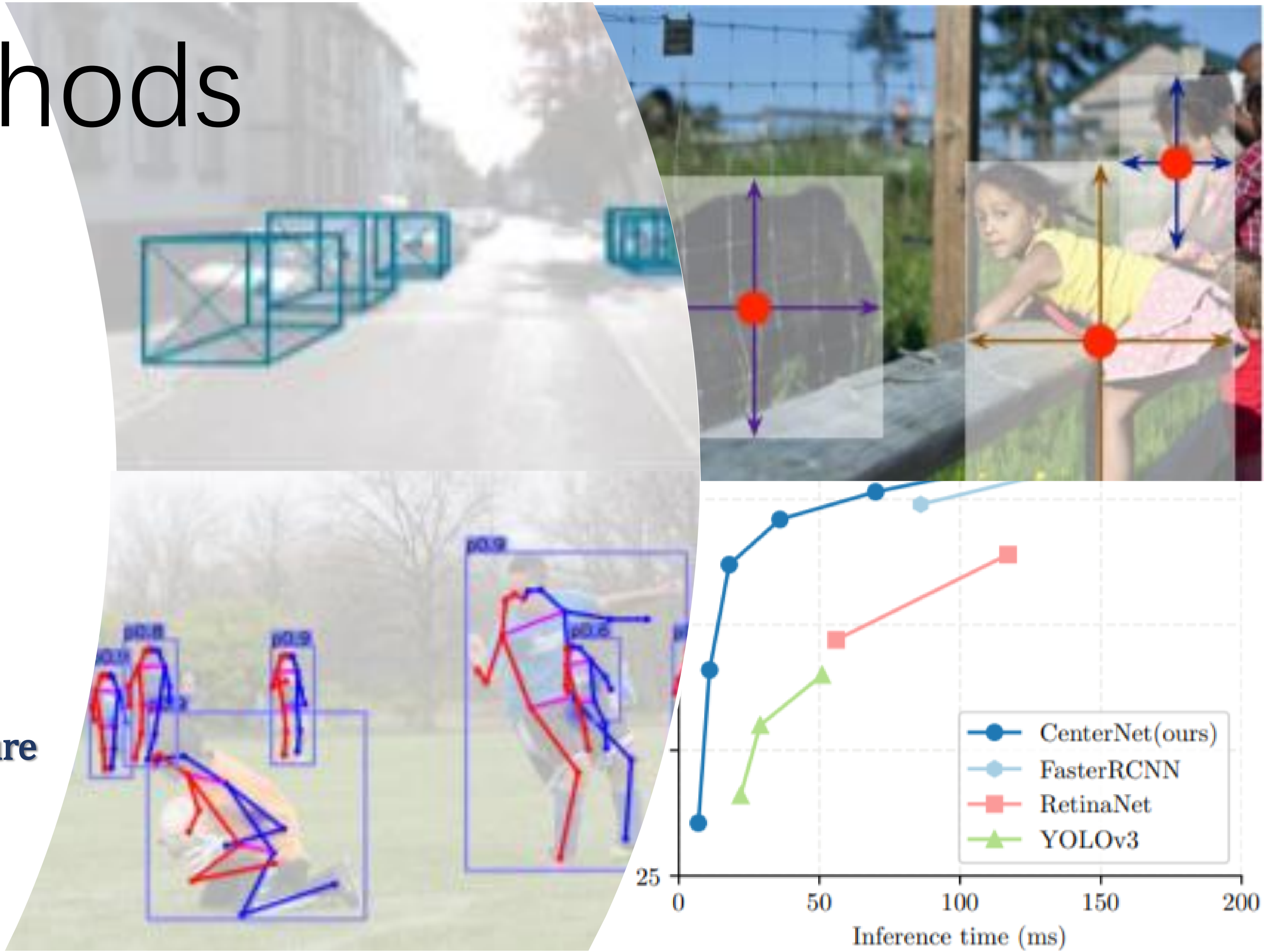


III. Other Methods

L. CenterNet [2019, Zhou]

➢ Features:

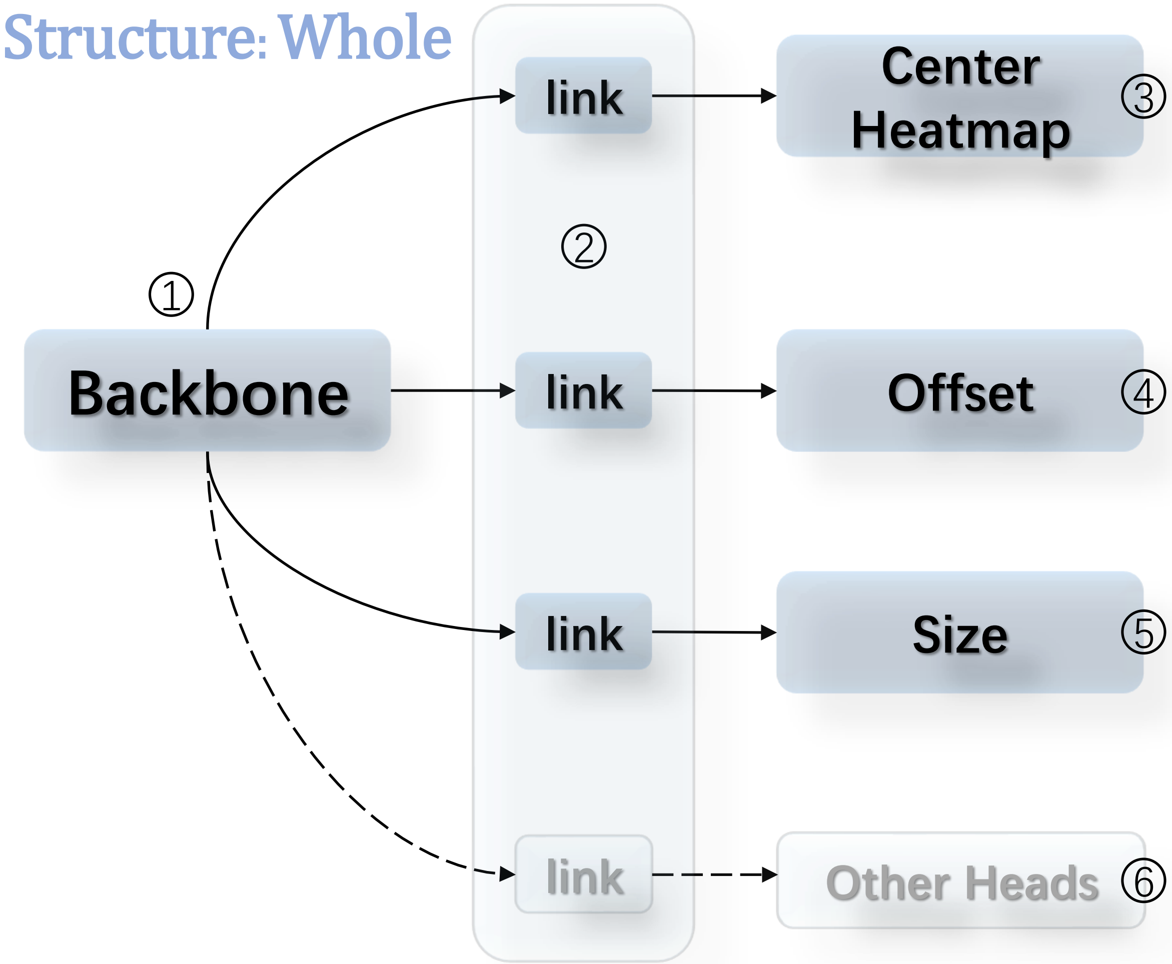
- More accurate & faster
- Detect "objects as points"
  [only detect center points]
- Multiple functions in one structure
- No need for post-processing
  [NMS etc.]



III. Other Methods

L. CenterNet [2019, Zhou]

➢ Structure: Whole



III. Other Methods

L. CenterNet [2019, Zhou]

- Hourglass: 1540, 2016, Newell
- Resnet + Transpose: 187, 2018, Xiao
- DLA: 145, 2018(2019), Yu
- Modified DLA: [This Paper]

➢ Structure: Backbone

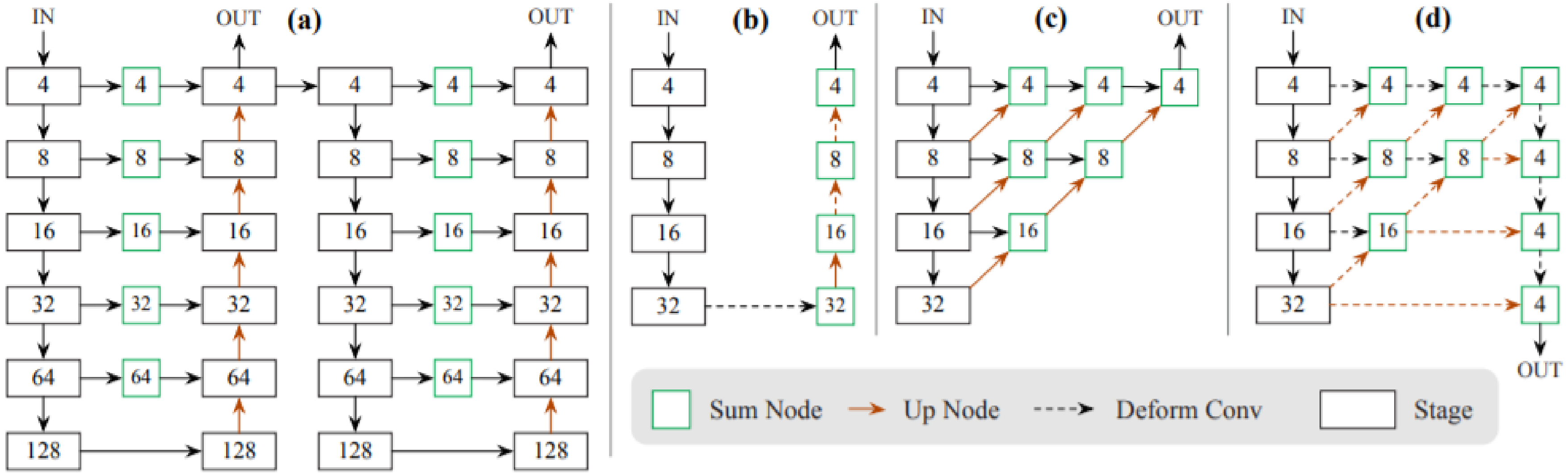
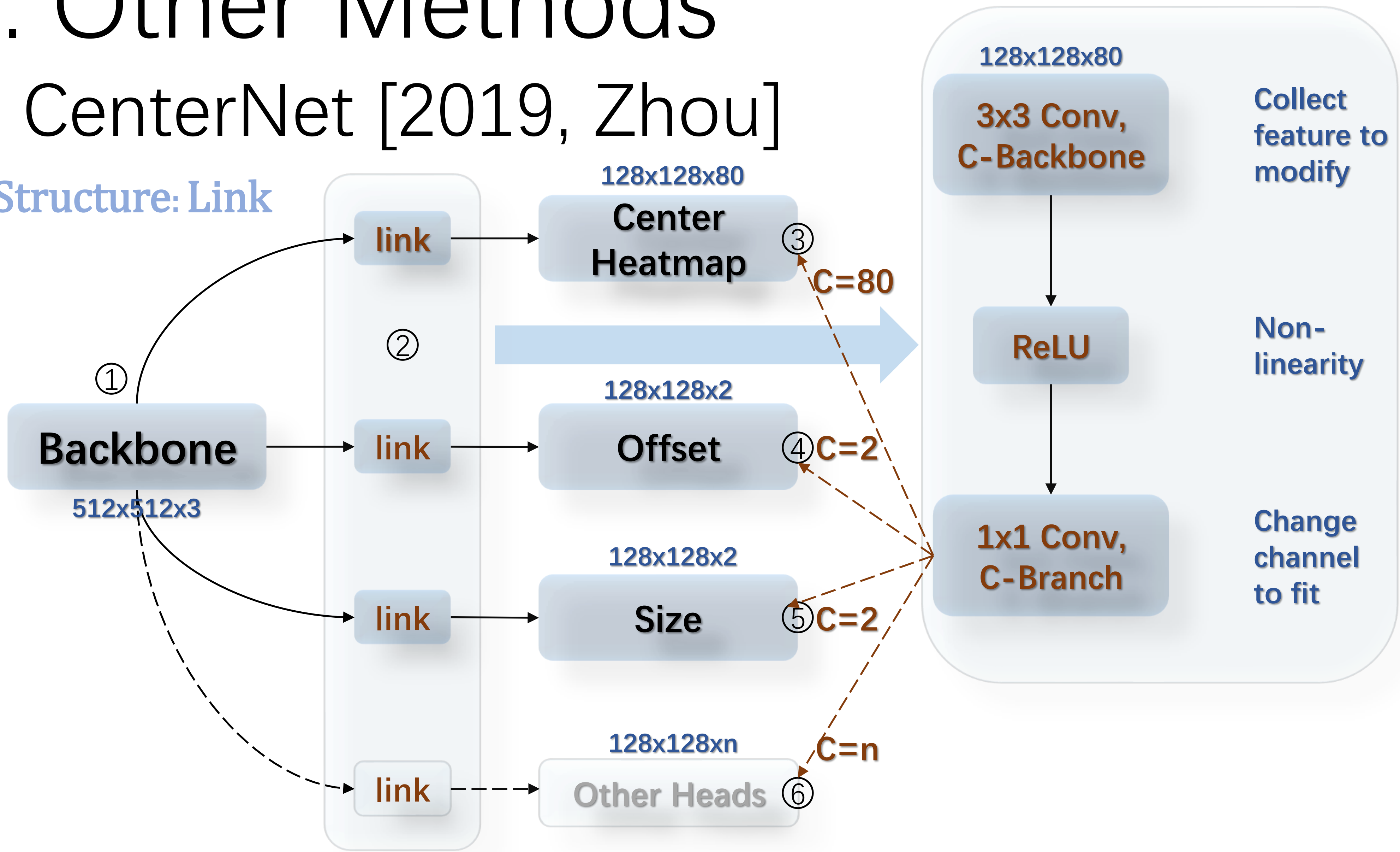


Figure 6: Model diagrams. The numbers in the boxes represent the stride to the image. (a): Hourglass Network [30]. We use it as is in CornerNet [30]. (b): ResNet with transpose convolutions [55]. We add one 3 × 3 deformable convolutional layer [63] before each up-sampling layer. Specifically, we first use deformable convolution to change the channels and then use transposed convolution to upsample the feature map (such two steps are shown separately in 32 → 16. We show these two steps together as a dashed arrow for 16 → 8 and 8 → 4). (c): The original DLA-34 [58] for semantic segmentation. (d): Our modified DLA-34. We add more skip connections from the bottom layers and upgrade every convolutional layer in upsampling stages to deformable convolutional layer.

III. Other Methods

L. CenterNet [2019, Zhou]

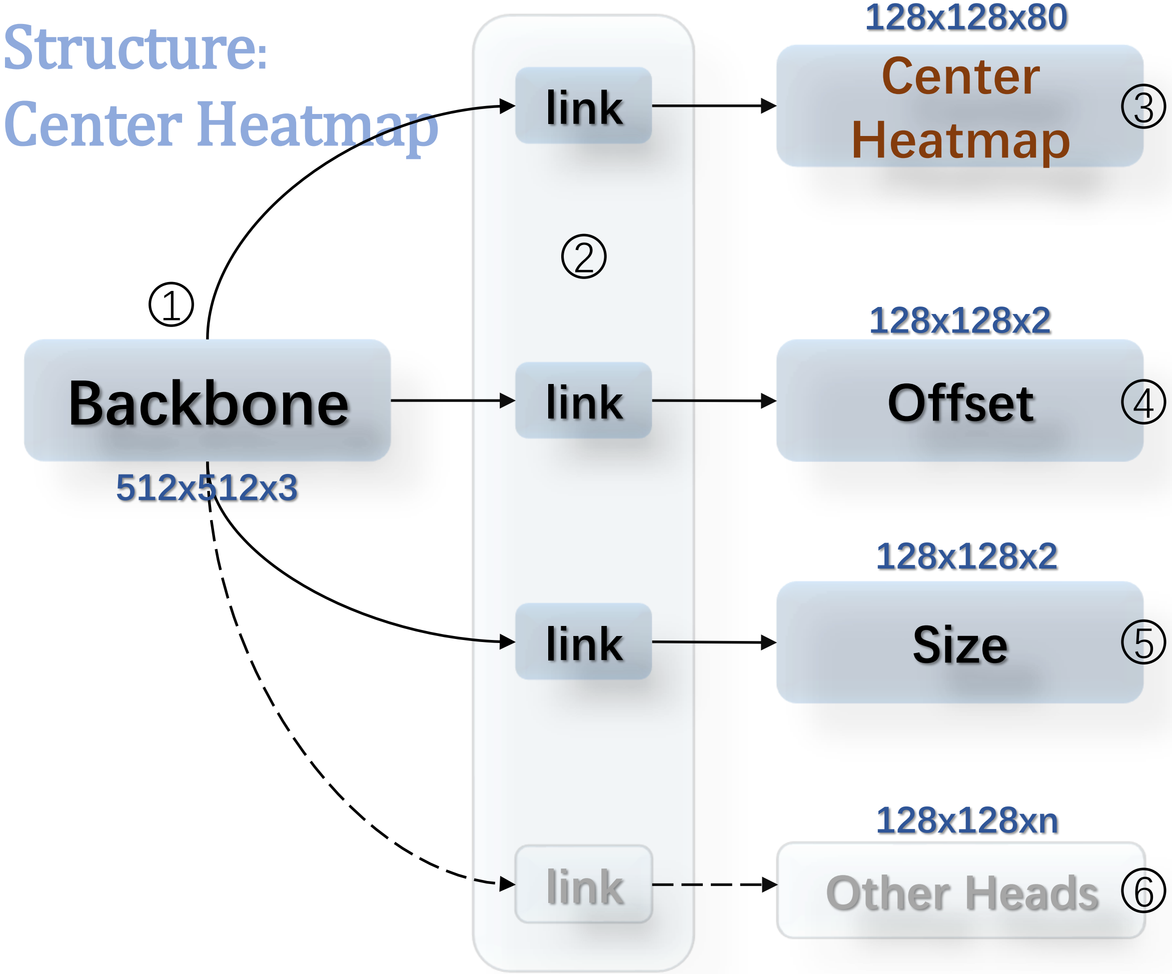
➢ Structure: Link

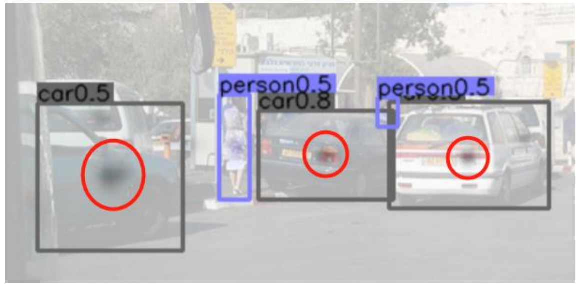


III. Other Methods

L. CenterNet [2019, Zhou]

➢ **Structure:**
Center Heatmap





- **Depict objects using their center points**
- **Describe center point by heatmap**

$Y_{xyc} = exp\left(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2}\right) \in [0,1]^{\frac{W}{R}\times\frac{H}{R}\times C},$

$\tilde{p} = \left\lfloor \frac{p=original\ center\ coord}{R=4} \right\rfloor,$

σ_p is determined by object size [CornerNet] [code]

If overlap, take element-wise maximum

- **One class one channel**
- **Regress focal loss: $\boldsymbol{\alpha = 2, \beta = 4}$**

$$L_k = \frac{-1}{N}\sum_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}\log(\hat{Y}_{xyc}), & Y_{xyc}=1\\ (1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha}\log(1-\hat{Y}_{xyc}), & Y_{xyc}\neq 1\end{cases}$$

- **Reference: +3x3 max pool → remove NMS**

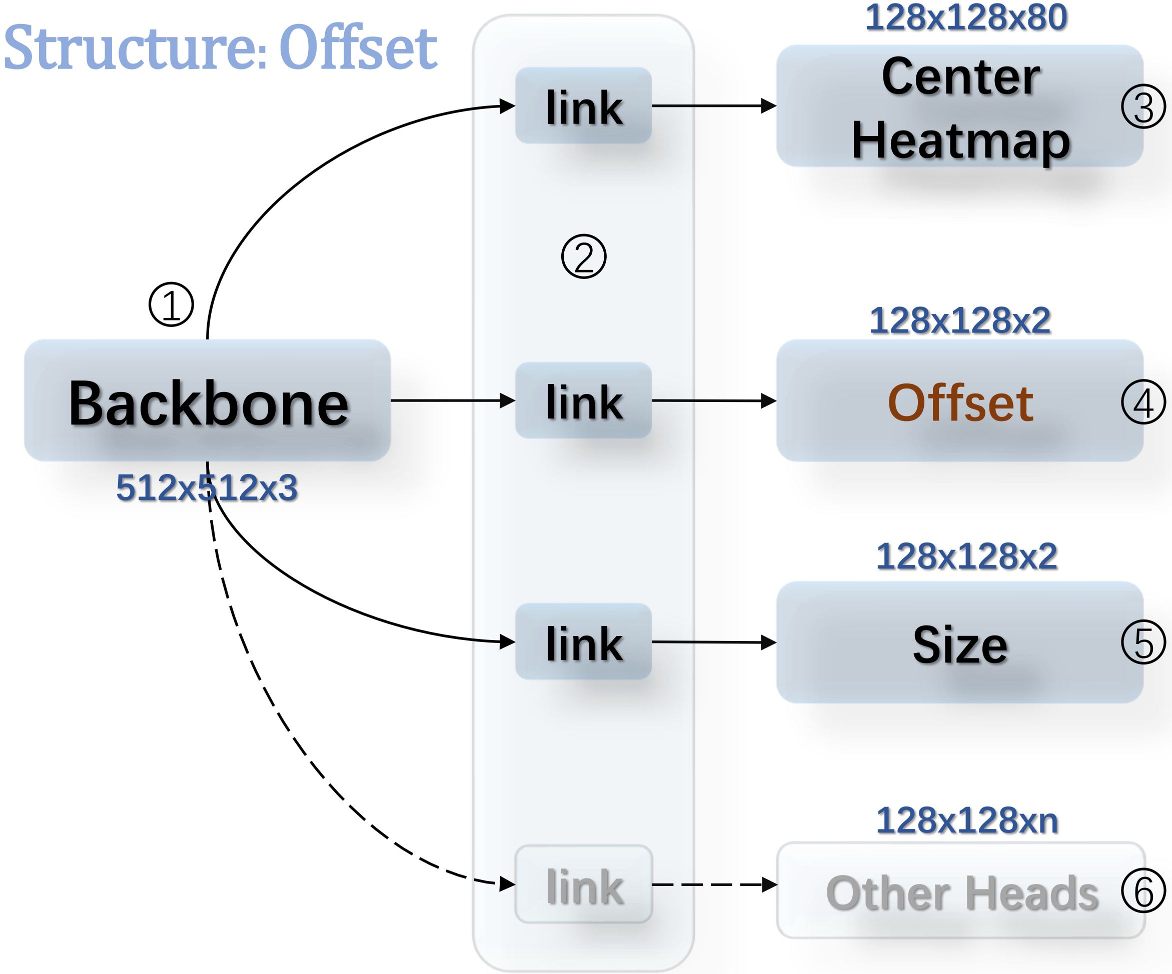
```
def _nms(heat, kernel=3):
    pad = (kernel - 1) // 2

    hmax = nn.functional.max_pool2d(
        heat, (kernel, kernel), stride=1, padding=pad)
    keep = (hmax == heat).float()
    return heat * keep
```

III. Other Methods

L. CenterNet [2019, Zhou]

➢ Structure: Offset

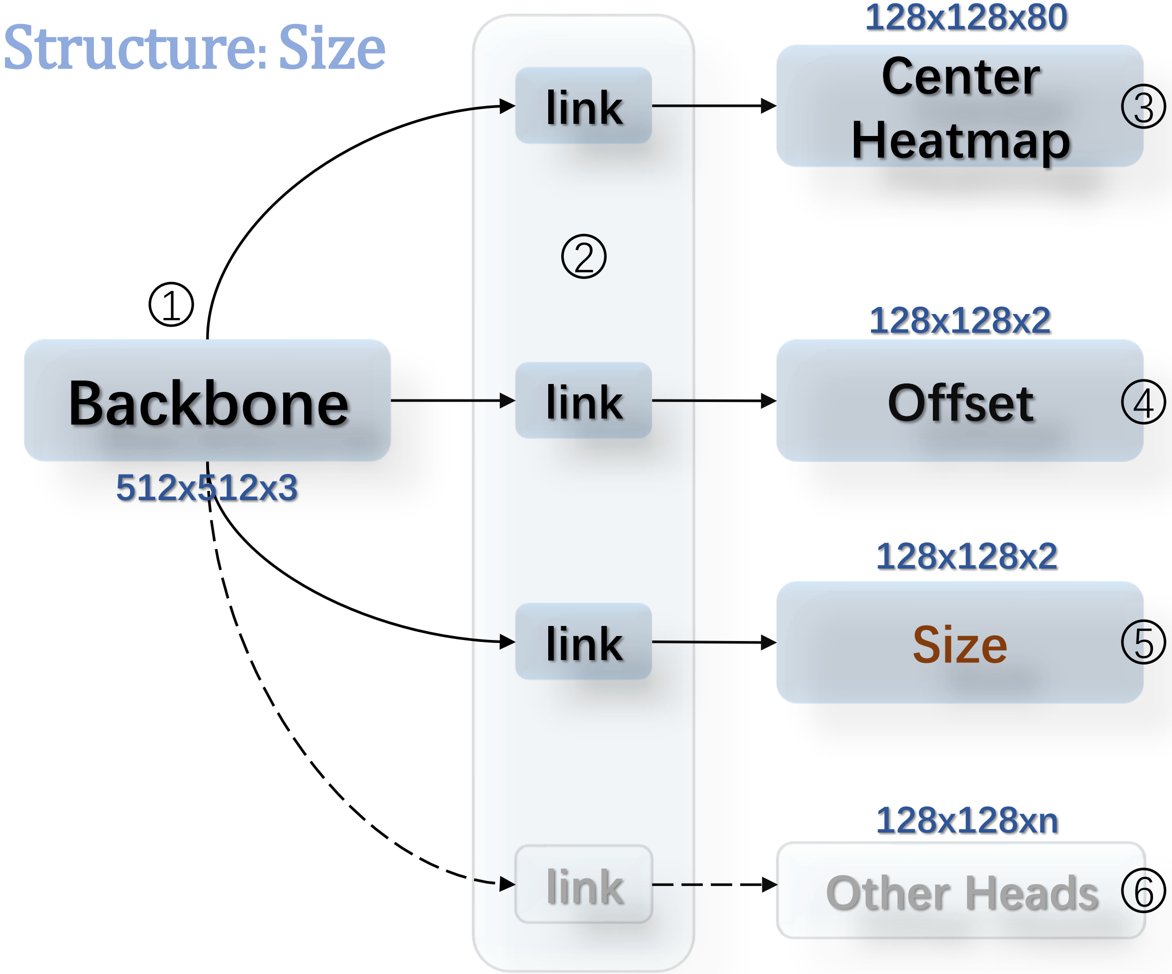


- **Target: Compensate discretization**
  (512→128, /4)
- **Predict 2-D offset:**
  $\hat{O}_{\hat{x}_i, \hat{y}_i} = (\delta\hat{x}_i, \delta\hat{y}_i)$
- **Regress L1 loss**
  $L_{off} = \frac{1}{N}\sum_p \left|\hat{O} - (\frac{p}{R} - \tilde{p})\right|$

III. Other Methods

L. CenterNet [2019, Zhou]

➢ Structure: Size



- **Predict BBox width & height**
- **Regress L1 loss**

$L_{size} = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{S}_{p_k} - s_k\right|,$

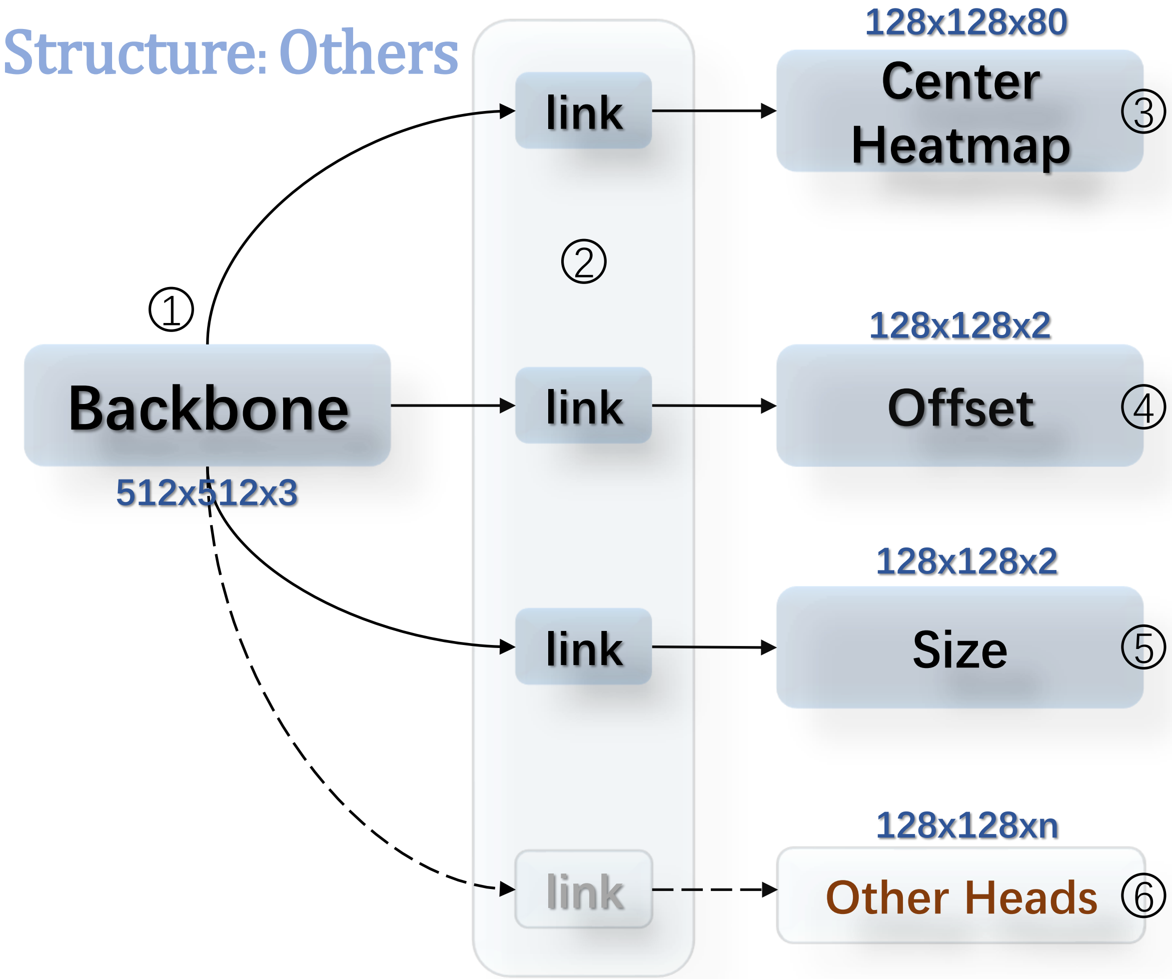
$s_k = \left(x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)}\right),$

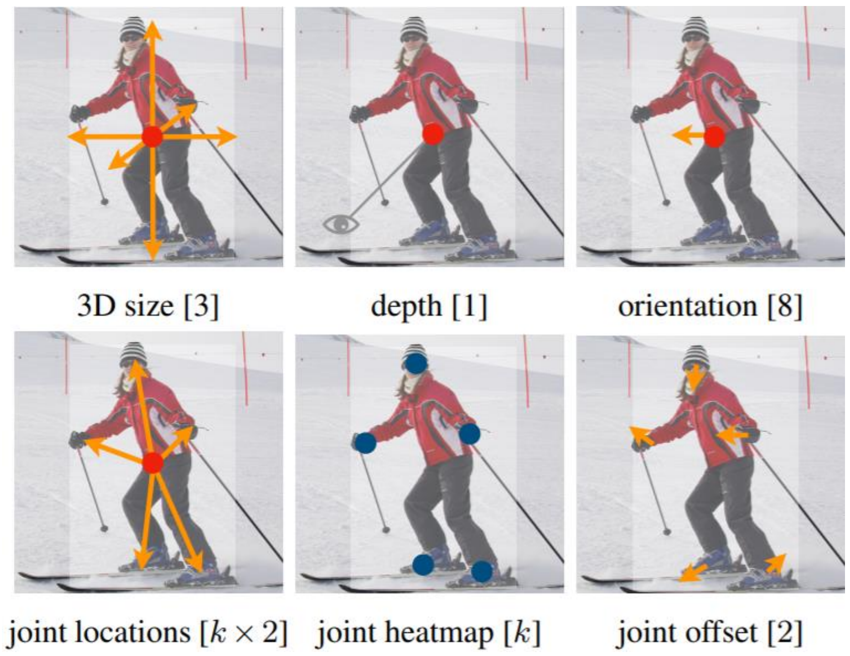
$k = id\ in\ class$

III. Other Methods

L. CenterNet [2019, Zhou]

➢ Structure: Others





III. Other Methods

L. CenterNet [2019, Zhou]

- **Train:**

128x128x80

Center Heatmap

$$L_k = \frac{-1}{N}\sum_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}\log(\hat{Y}_{xyc}), & Y_{xyc}=1\\(1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha}\log(1-\hat{Y}_{xyc}), & Y_{xyc}\neq 1\end{cases}$$

128x128x2

Offset

$$L_{off} = \frac{1}{N}\sum_p \left|\hat{O} - \left(\frac{p}{R} - \tilde{p}\right)\right|$$

128x128x2

Size

$$L_{size} = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{S}_{p_k} - s_k\right|$$

- Use Adam, 10x learning rate dropped during training twice.
- Random flip, scaling(0.6~1.3), cropping, color jittering as augmentation
- Down sampling network structures are pretrained using ImageNet
- Different net has different LR initialization

$$\boldsymbol{L_{det} = L_k + \lambda_{size}L_{size} + \lambda_{off}L_{off}, (\lambda_{size} = 0.1, \lambda_{off} = 1)}$$

III. Other Methods

L. CenterNet [2019, Zhou]

- 3 test augmentations:
  x, flip, flip + multi-scale
  (0.5, 0.75, 1, 1.25, 1.5)
- Do NMS when multi-scale

➢ **Test:**

	AP			AP_{50}			AP_{75}			Time (ms)			FPS		
	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS
Hourglass-104	**40.3**	**42.2**	**45.1**	**59.1**	**61.1**	**63.5**	**44.0**	**46.0**	**49.3**	71	129	672	14	7.8	1.4
DLA-34	37.4	39.2	41.7	55.1	57.0	60.1	40.8	42.7	44.9	19	36	248	52	28	4
ResNet-101	34.6	36.2	39.3	53.0	54.8	58.5	36.9	38.7	42.0	22	40	259	45	25	4
ResNet-18	28.1	30.0	33.2	44.9	47.5	51.5	29.6	31.6	35.1	**7**	**14**	**81**	**142**	**71**	**12**

Resolution	AP	AP_{50}	AP_{75}	Time
Original	**36.3**	54.0	**39.6**	19
512	36.2	**54.3**	38.7	16
384	33.2	50.5	35.0	**11**

(a) Testing resolution: Lager resolutions perform better but run slower.

λ_{size}	AP	AP_{50}	AP_{75}
0.2	33.5	49.9	36.2
0.1	**36.3**	54.0	**39.6**
0.02	35.4	**54.6**	37.9

(b) Size regression weight. $\lambda_{size} \leq 0.1$ yields good results.

Loss	AP	AP_{50}	AP_{75}
l1	**36.3**	**54.0**	**39.6**
smooth l1	33.9	50.9	36.8

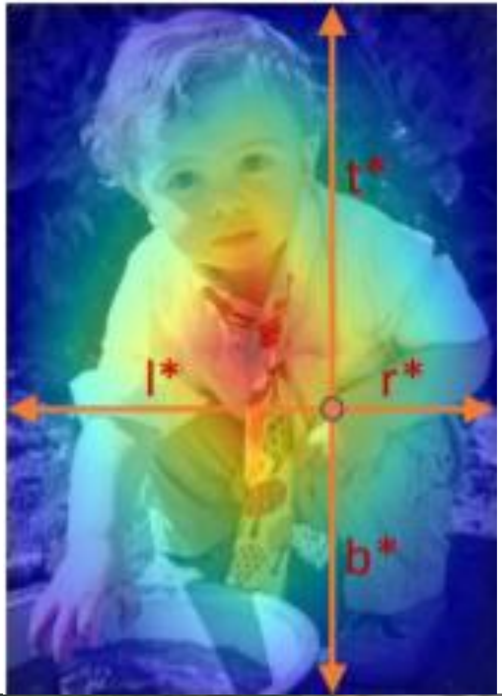
(c) Regression loss. L1 loss works better than Smooth L1.

Epoch	AP	AP_{50}	AP_{75}
140	36.3	54.0	39.6
230	**37.4**	**55.1**	**40.8**

(d) Training schedule. Longer performs better.

III. Other Methods

M. FCOS [2019, Tian]

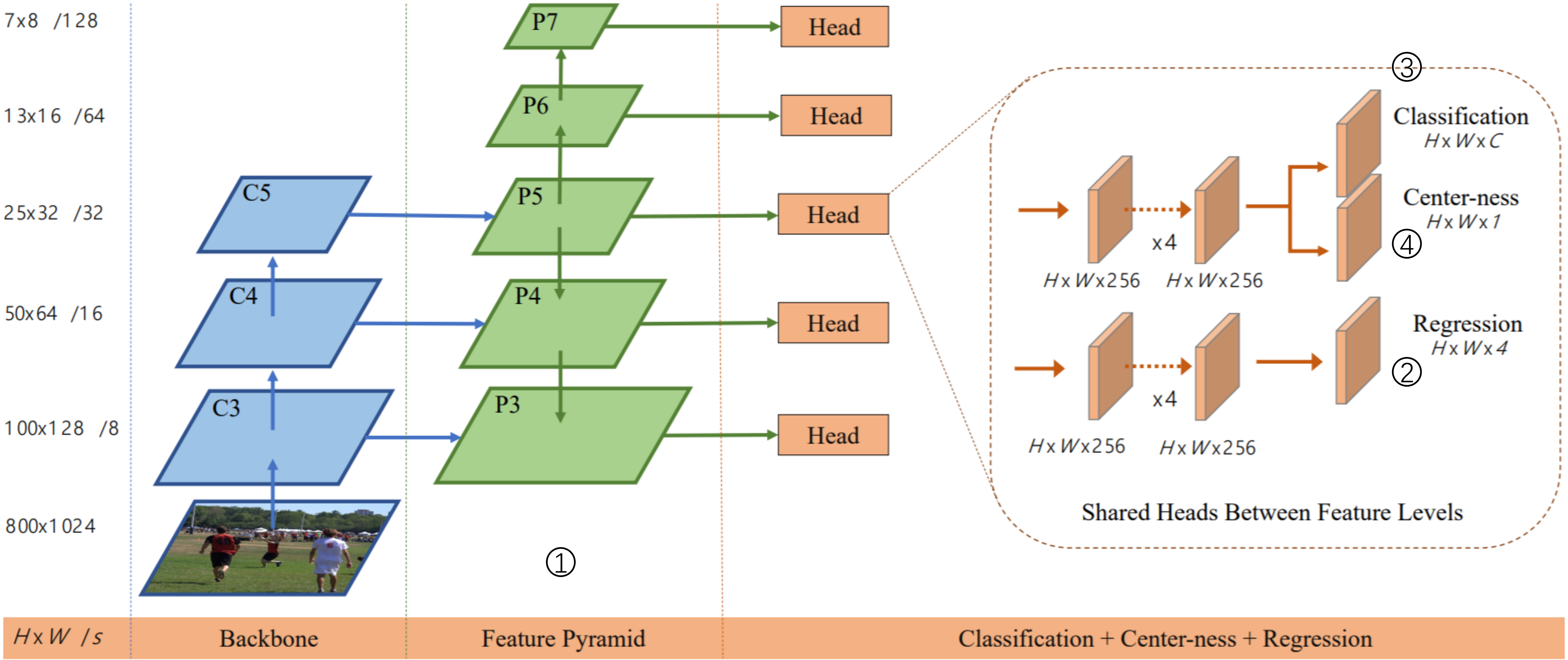


- **Features:**
  - Detect objects by pixels
    [detect all points with an object]
  - FPN as backbone
  - Detect by scale
  - Fully convolution one-stage detection
  - Need NMS for post-processing

III. Other Methods

M. FCOS [2019, Tian]

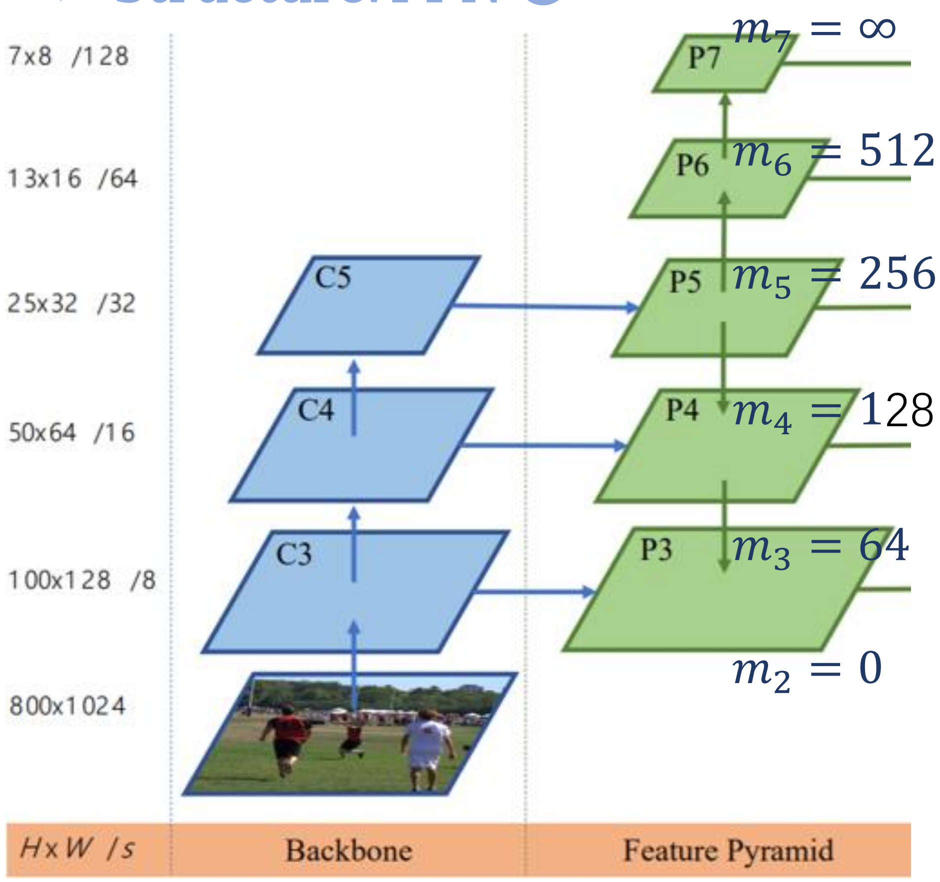
➢ Structure: Whole

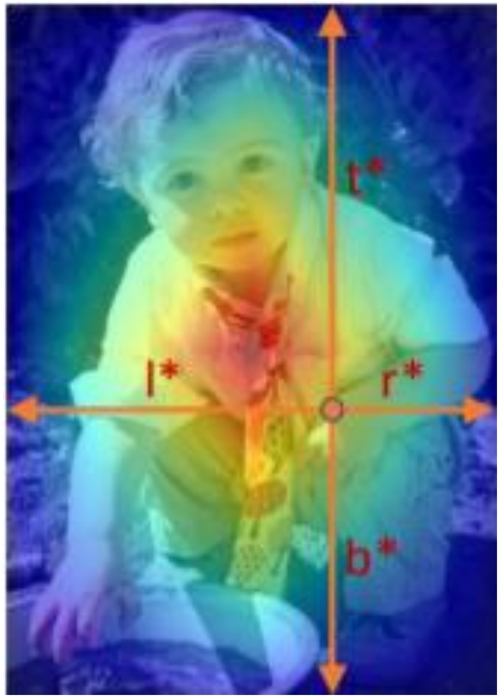


III. Other Methods

M. <u>FCOS</u> [2019, Tian]

➢ Structure: FPN ①





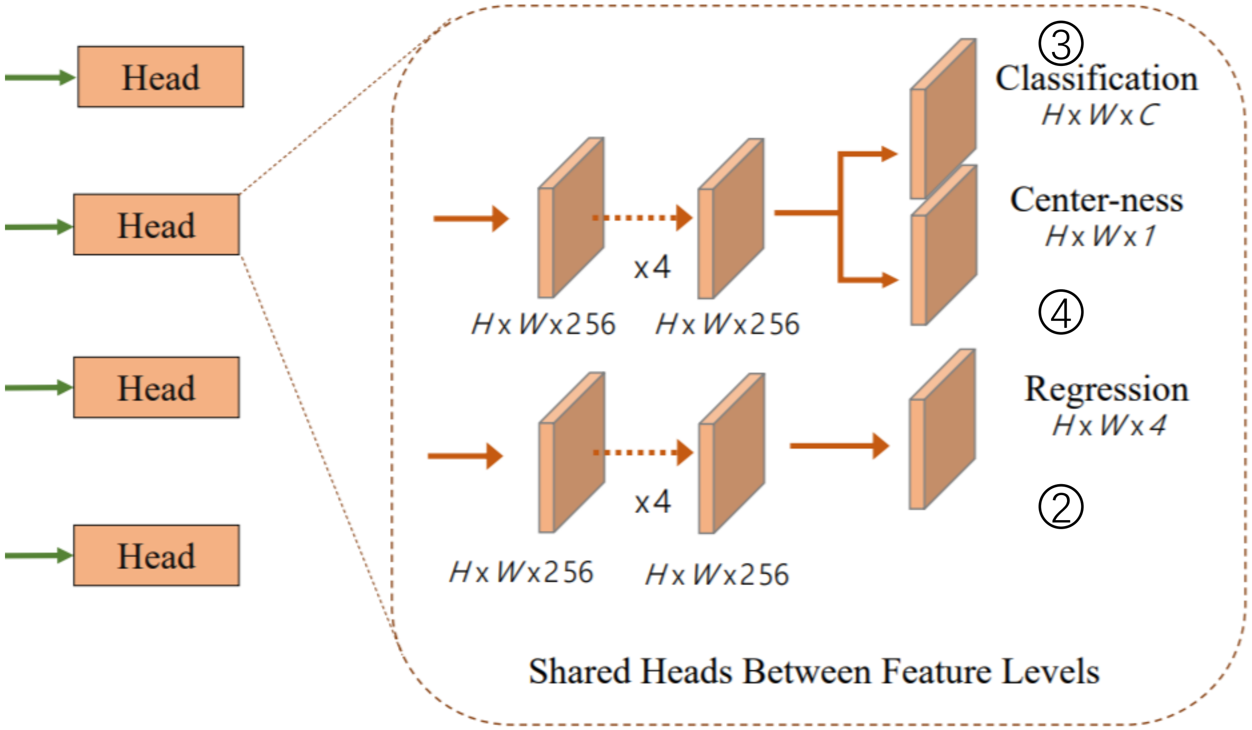
- **<u>FPN</u>: [Lin, 2016]**
- $Pi\colon m_{i-1} < max(l^*, r^*, t^* b^*) < m_i$
- $(l^*, r^*, t^* b^*)$**, denoted by** t^***, is calculated on feature map for each feature pixel which can be mapped inside the ground truth bbox by using method:** $(\lfloor \frac{s}{2} + xs \rfloor, \lfloor \frac{s}{2} + ys \rfloor)$**,** s **is the stride**
- $*$**: ground truth**

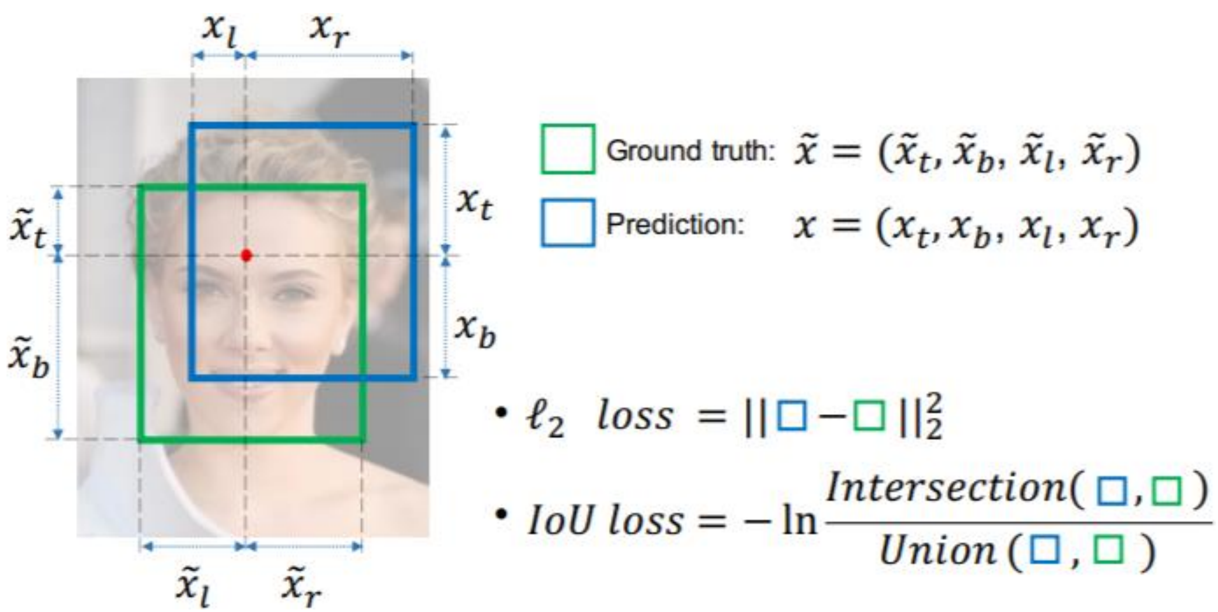
III. Other Methods

M. FCOS [2019, Tian]

➢ Structure: Reg ②

- *IoU loss*: UnitBox [Yu, CVPR 2016]
- $L_{reg} = IoU\ loss(t_{x,y}, t^*_{x,y})$

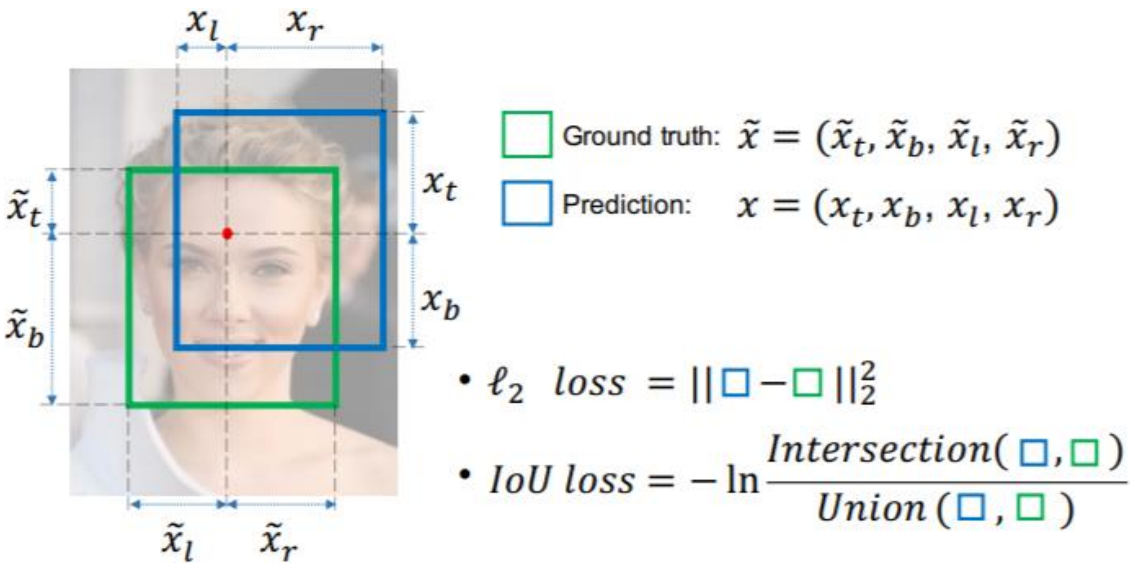




III. Other Methods

M. FCOS [2019, Tian]

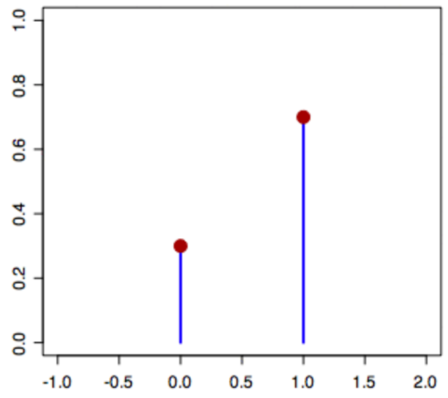
➢ **Structure: Reg ② — IoU Loss Forward**



Algorithm 1: *IoU* loss Forward

Input: $\widetilde{x}$ as bounding box ground truth
Input: x as bounding box prediction
Output: $\mathcal{L}$ as localization error
for *each pixel* (i, j) **do**
  if $\widetilde{x} \neq \mathbf{0}$ **then**
    $X = (x_t + x_b) * (x_l + x_r)$
    $\widetilde{X} = (\widetilde{x}_t + \widetilde{x}_b) * (\widetilde{x}_l + \widetilde{x}_r)$
    $I_h = min(x_t, \widetilde{x}_t) + min(x_b, \widetilde{x}_b)$
    $I_w = min(x_l, \widetilde{x}_l) + min(x_r, \widetilde{x}_r)$
    $I = I_h * I_w$
    $U = X + \widetilde{X} - I$
    $IoU = \frac{I}{U}$
    $\mathcal{L} = -ln(IoU)$
  else
    $\mathcal{L} = 0$
  end
end

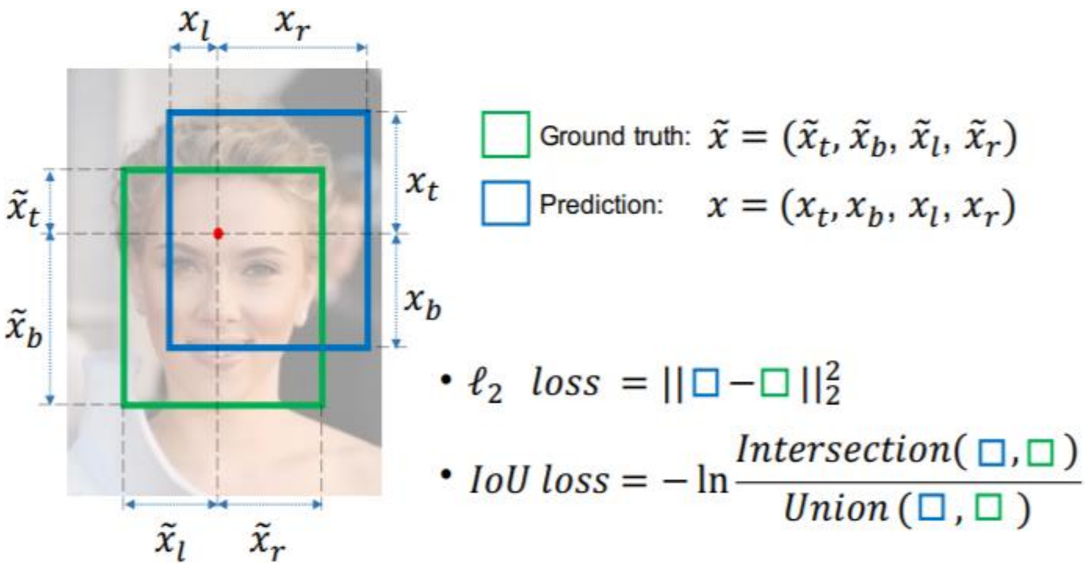
- $\tilde{x} \neq 0$, **pixel(i,j) falls inside a valid bbox**
- $IoU \in [0,1]$
- $L = -ln(IoU) = -pln(IoU) - (1-p)ln(1-IoU)$
- **Let** $p(IoU = 1) = 1$ **when** IoU **under Bernoulli distribution**



III. Other Methods

M. FCOS [2019, Tian]

➢ **Structure: Reg ② — IoU Loss Backward**



Algorithm 1: IoU loss Forward

Input: $\tilde{x}$ as bounding box ground truth
Input: x as bounding box prediction
Output: $\mathcal{L}$ as localization error
for *each pixel* (i, j) **do**
  if $\tilde{x} \neq \mathbf{0}$ **then**
    $X = (x_t + x_b) * (x_l + x_r)$
    $\tilde{X} = (\tilde{x}_t + \tilde{x}_b) * (\tilde{x}_l + \tilde{x}_r)$
    $I_h = min(x_t, \tilde{x}_t) + min(x_b, \tilde{x}_b)$
    $I_w = min(x_l, \tilde{x}_l) + min(x_r, \tilde{x}_r)$
    $I = I_h * I_w$
    $U = X + \tilde{X} - I$
    $IoU = \frac{I}{U}$
    $\mathcal{L} = -ln(IoU)$
  else
    $\mathcal{L} = 0$
  end
end

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{I(\nabla_x X - \nabla_x I) - U \nabla_x I}{U^2 IoU} = \frac{1}{U}\nabla_x X - \frac{U+I}{UI}\nabla_x I.$$

How to make your own loss function?

$$\frac{\partial X}{\partial x_t (\mathbf{or}\ \partial x_b)} = x_l + x_r$$

$$\frac{\partial X}{\partial x_l (\mathbf{or}\ \partial x_r)} = x_t + x_b$$

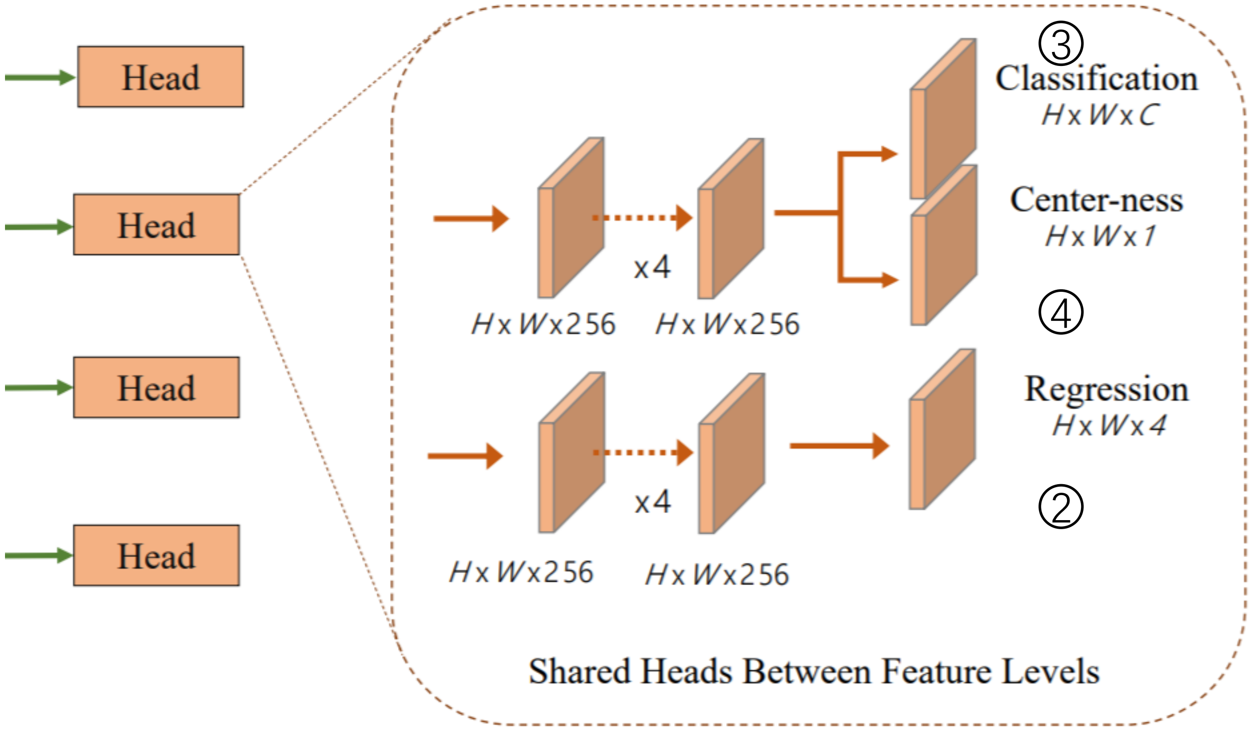
$$\frac{\partial I}{\partial x_t (\mathbf{or}\ \partial x_b)} = \begin{cases} I_w, & \text{if } x_t < \tilde{x}_t (\mathbf{or}\ x_b < \tilde{x}_b) \\ 0, & \text{otherwise,} \end{cases}$$

$$\frac{\partial I}{\partial x_l (\mathbf{or}\ \partial x_r)} = \begin{cases} I_h, & \text{if } x_l < \tilde{x}_l (\mathbf{or}\ x_r < \tilde{x}_r) \\ 0, & \text{otherwise.} \end{cases}$$

III. Other Methods

M. FCOS [2019, Tian]

➤ Structure: Cls ③

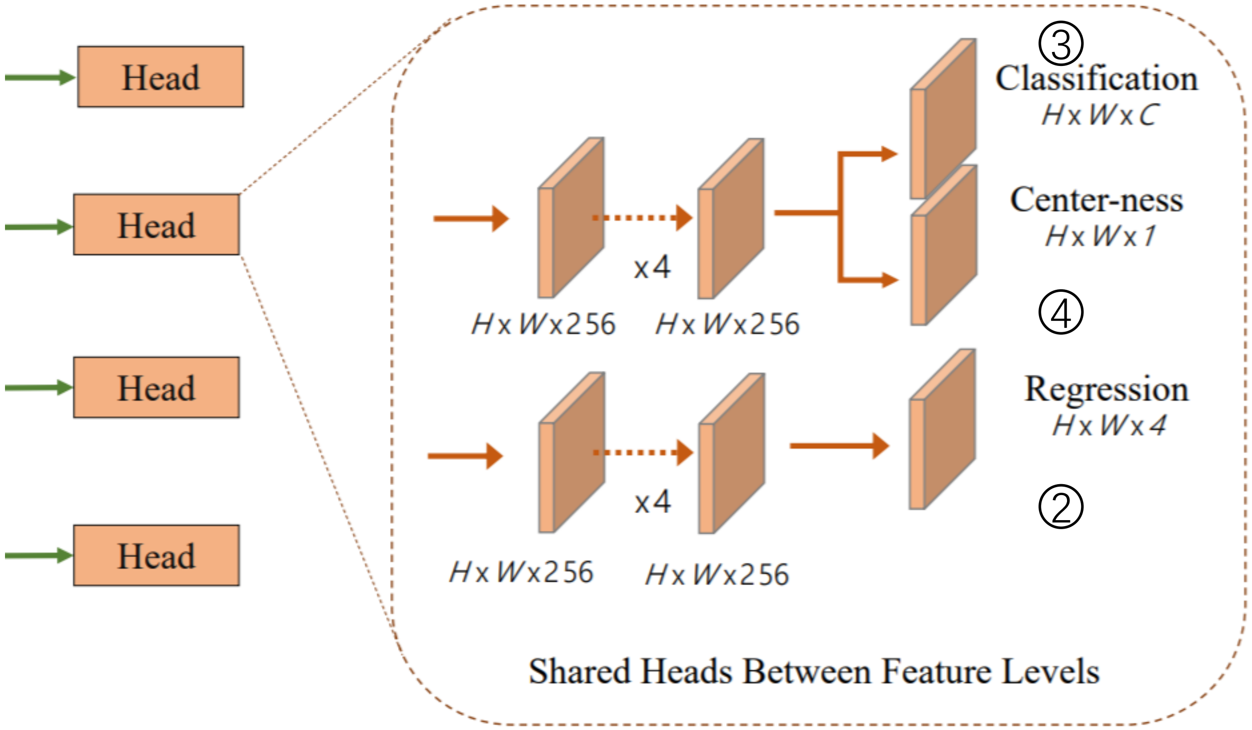


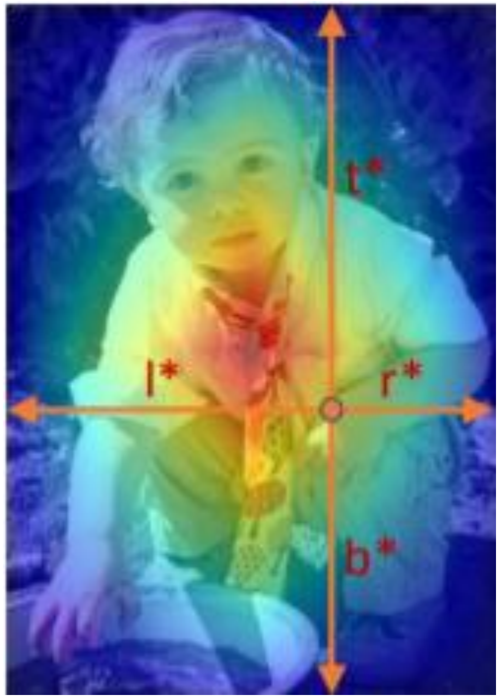
- *Cls loss*: **Focal Loss**
- $L_{cls} = -\alpha_t(1-p_t)^\gamma \log(p_t)$
  $[\gamma = 2, \alpha = 0.25]$

III. Other Methods

M. FCOS [2019, Tian]

➤ **Structure: Center-ness ④**





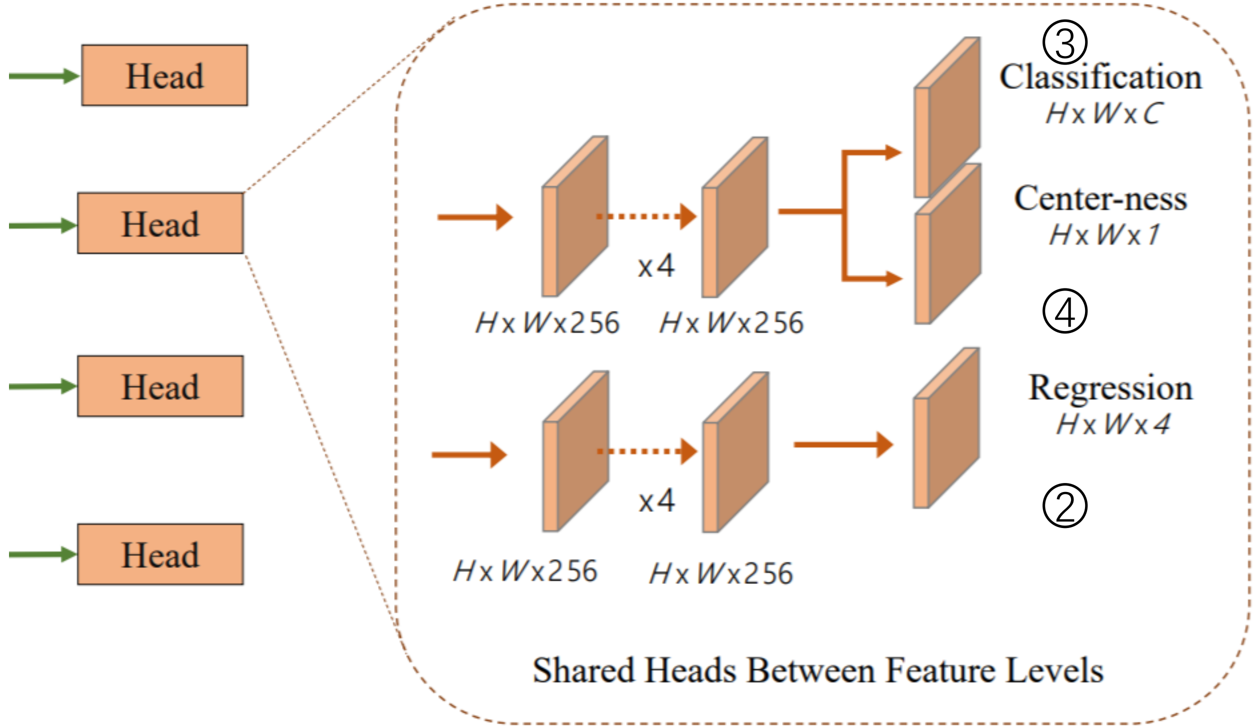
- **Target: fix low-quality border detection**
- **Set a score to each pixel**
- $centerness^* = \sqrt{\frac{\min(l^*,r^*)}{\max(l^*,r^*)} \times \frac{\min(t^*,b^*)}{\max(t^*,b^*)}}$

III. Other Methods

M. FCOS [2019, Tian]

➤ **Train:**

$$L_{p,t} = \frac{1}{N_{pos}} \sum_{x,y} L_{cls}(p_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{pos}} \mathbb{I}_{\{c^*_{x,y}>0\}} \left(\sum_{x,y} L_{reg}(t_{x,y}, t^*_{x,y}) + \sum_{x,y} L_{cent}(ct_{x,y}, ct^*_{x,y})\right)$$



$$L_{cls} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

$$L_{cent} = BCE(ct^*, ct)$$

$$L_{reg} = IoU\ loss(t_{x,y}, t^*_{x,y})$$