名企 Project

AI for CV Group
2021

Week 6-7
Advanced Detection Tricks

Contents:

Contents:

I. Data Augmentation

I. Data Augmentation

A. Mixup[2018, Zhang]

- **Features:**

a. **Mixed up to 2 images is good enough**
   [3 or more, similar effect, more time]

b. **Mixed within one batch is enough**
   [No need to mix within the whole dataset: Same effect]

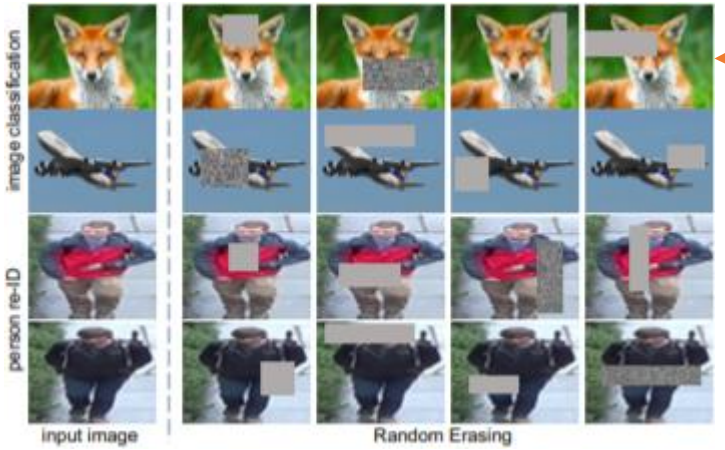
c. **Better to mix within different labeled imgs**
   [No promotion within same labels]

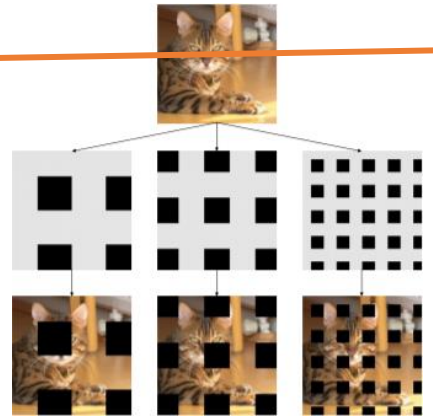
d. **Two types of implementation**
   [Let's see code later]

e. **Mainly for classification**

I. Data Augmentation

B. Cutout [2017, DeVires]





- **Features:**

a. **Normalize first, then cutout**
[Reduce the potential effect]

b. **(Size > shape) of the cutout patch**
[Cut square patch;
Rectangle: Random erasing (2017, Zhong)]

c. **Too simple, might erase useful info**
[Improvement: GridMask (2020, Chen)]

I. Data Augmentation

C. CutMix [2019, Yun]

	Mixup	Cutout	CutMix
Usage of full image region	✔	✘	✔
Regional dropout	✘	✔	✔
Mixed image & label	✔	✘	✔

- **Features:**

a. **Cutout + Mixup**
[Better than mixup apparently]

b. **Combine 2 images**
[2 is enough]

c. **Algorithm in details**
[Improvement: GridMask (2020, Chen)]

I. Data Augmentation

C. CutMix [2019, Yun]

Algorithm A1 Pseudo-code of CutMix

```
for each iteration do
    input, target = get_minibatch(dataset)          ▷ input is N×C×W×H size tensor, target is N×K size tensor.
    if mode == training then
        input_s, target_s = shuffle_minibatch(input, target)          ▷ CutMix starts here.
        lambda = Unif(0,1)
        r_x = Unif(0,W)
        r_y = Unif(0,H)
        r_w = Sqrt(1 - lambda) ·W
        r_h = Sqrt(1 - lambda) ·H
        x1 = Round(Clip(r_x - r_w / 2, min=0))
        x2 = Round(Clip(r_x + r_w / 2, max=W))
        y1 = Round(Clip(r_y - r_h / 2, min=0))
        y2 = Round(Clip(r_y + r_h / 2, min=H))
        input[:, :, x1:x2, y1:y2] = input_s[:, :, x1:x2, y1:y2]
        lambda = 1 - (x2-x1)*(y2-y1)/(W*H)          ▷ Adjust lambda to the exact area ratio.
        target = lambda * target + (1 - lambda) * target_s          ▷ CutMix ends.
    end if
    output = model_forward(input)
    loss = compute_loss(output, target)
    model_update()
end for
```

Forgotten by original paper

I. Data Augmentation

D. Mosaic


aug_-319215602_0_-238783579.jpg


aug_-1271888501_0_-749611674.jpg


aug_1462167959_0_-1659206634.jpg


aug_1474493600_0_-45389312.jpg


aug_1715045541_0_603913529.jpg


aug_1779424844_0_-589696888.jpg

- **Features:**

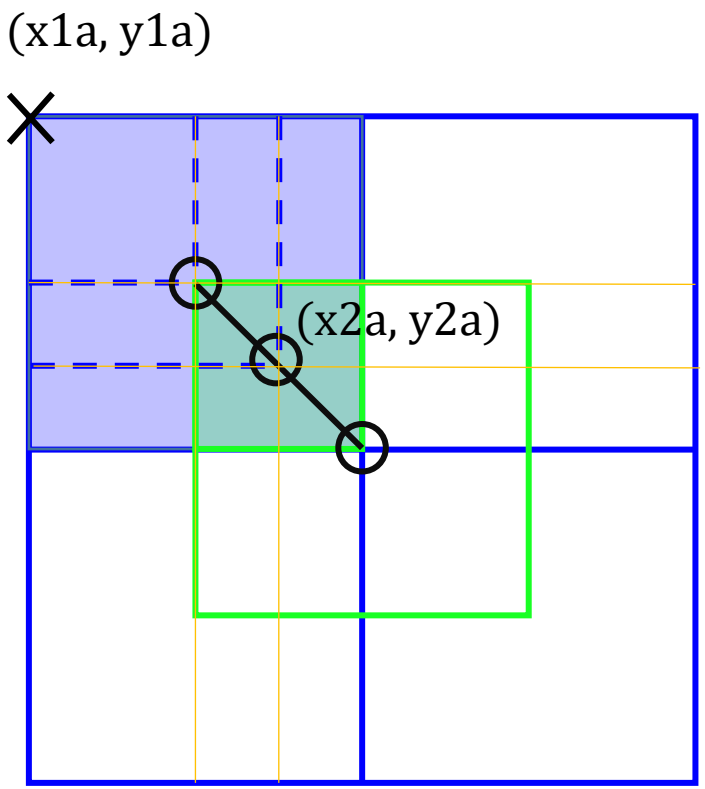
a. **Enhance background complex**

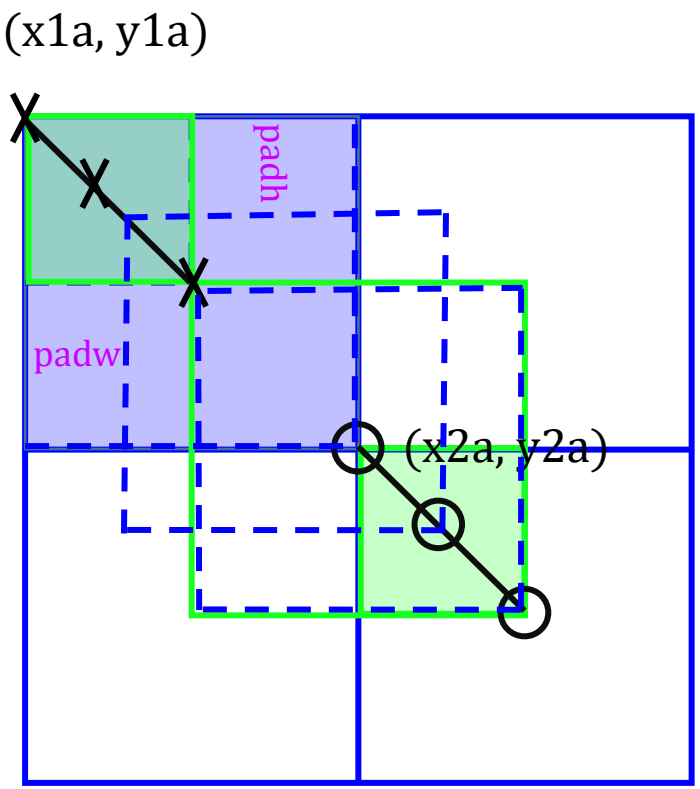
b. **⇔ Increase batch size**

c. **Several types of implementation**
[We'll see one]

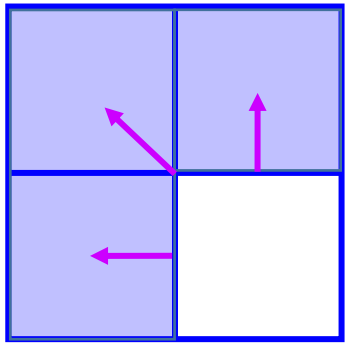
I. Data Augmentation

D. Mosaic





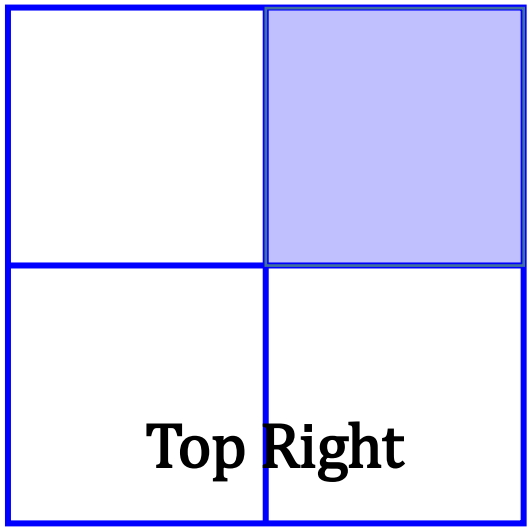
Top Left

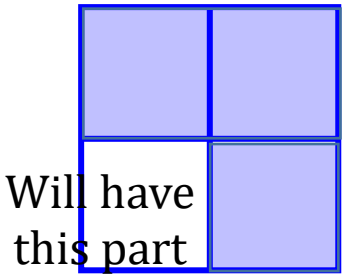


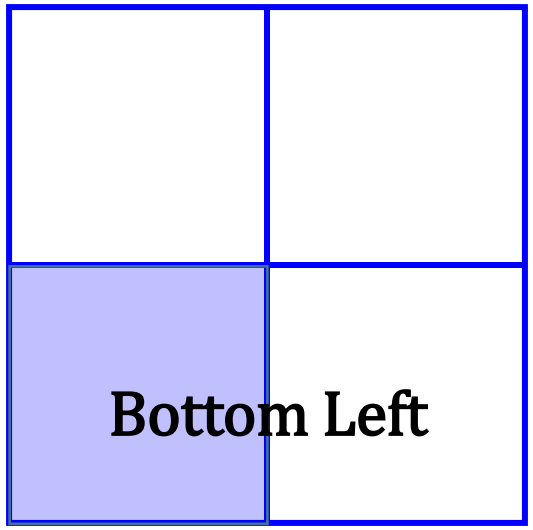
Will have this part

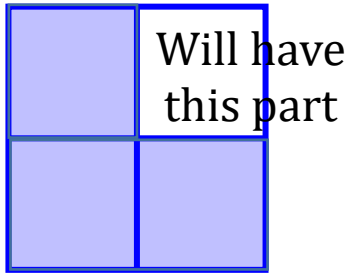
I. Data Augmentation

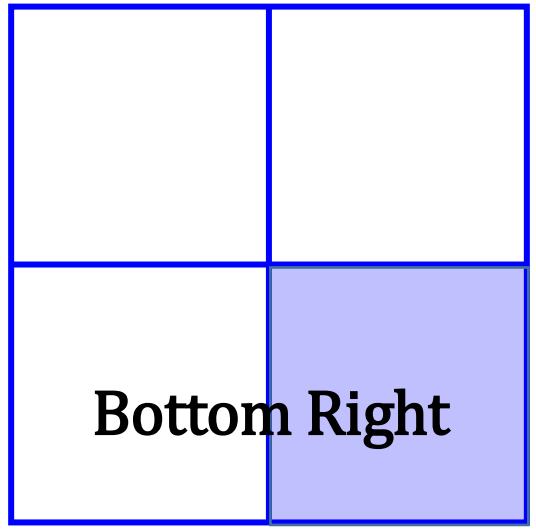
D. Mosaic

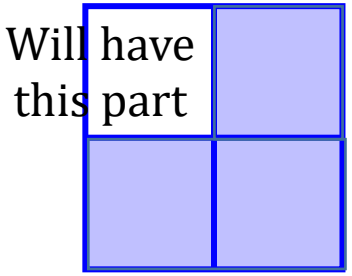












II. Regularization

II. Regularization

E. Label Smoothing [2015, InceptionV2, Szegedy]

- **Equations:**

$$\begin{cases} y_i' = (1-\epsilon)\cdot y_i + \epsilon u(K) \\ u(K) = \dfrac{1}{K}\ , (K = number\ of\ classes) \end{cases}$$

$$\Rightarrow \begin{cases} 1-\epsilon+\epsilon u(K), y = y_i \\ \epsilon u(K), \quad otherwise \\ u(K) = \dfrac{1}{K} \end{cases}$$

$$\epsilon = 0.1\ (by\ default)$$

- **Derivation:**

II. Regularization

E. Label Smoothing [2015, InceptionV2, Szegedy]

- **Cross Entropy Loss:**

$$H(y,p) = -\sum_i y_i log p_i \quad \text{— originally}$$

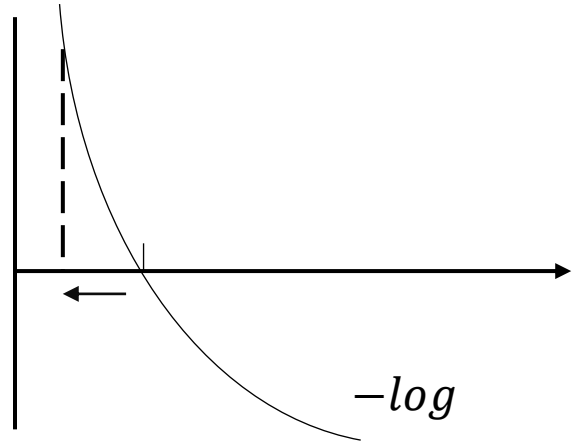
$$H(y',p) = -\sum_i y_i log p_i$$

$$= -\sum_i [(1-\epsilon)\cdot y_i + \epsilon u(K)] log p_i$$

$$= \sum_i \{(1-\epsilon)[-y_i log p_i] + \epsilon[-u(k) log p_i]\}$$

$$= (1-\epsilon)H(y,p) + \epsilon H(u,p)$$

- **Nature of Regularization:**



· $p \to 1, -logp \to 0$

· $\exists u(K), loss \uparrow$
$\Leftrightarrow H(y,p)\ will\ be\ pulled\ by\ H(u,p)$
$\Leftrightarrow move\ along\ "\leftarrow"$

II. Regularization

F. DropBlock [2018, Ghiasi]

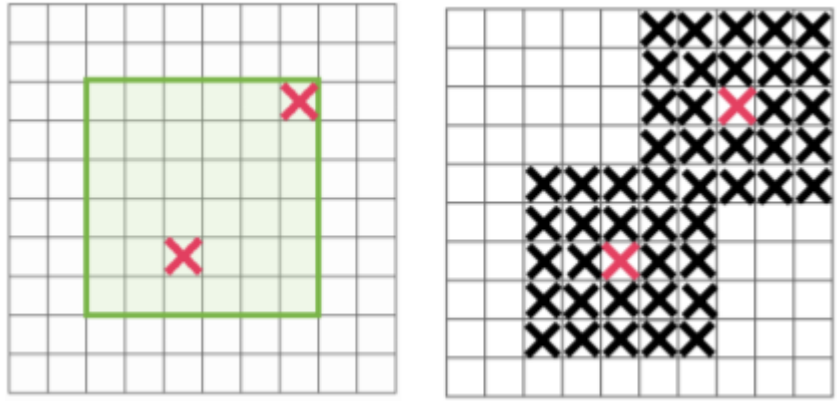
- **Feature:**

a. Structuralized Dropout for Conv

b. Dropout is random.
But a Conv has shared local info

c. Meaningless to drop randomly.
Drop based on patches.

- **Illustration:**



II. Regularization

F. DropBlock [2018, Ghiasi]

- **Algorithm:**

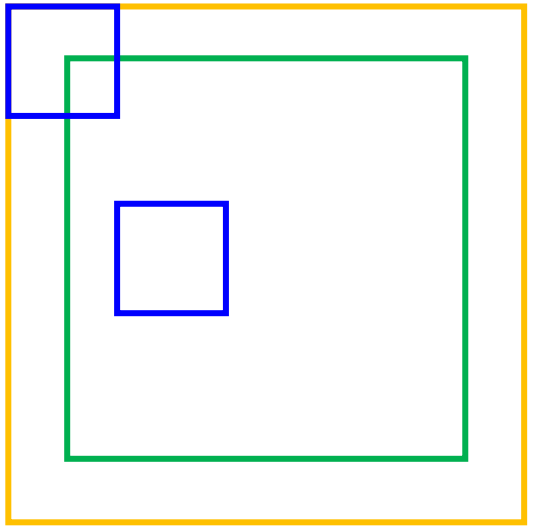
Algorithm 1 DropBlock

1: **Input:** output activations of a layer (A), $block\_size$, γ, $mode$
2: **if** $mode$ == *Inference* **then**
3: return A
4: **end if**
5: Randomly sample mask M: $M_{i,j} \sim Bernoulli(\gamma)$
6: For each zero position $M_{i,j}$, create a spatial square mask with the center being $M_{i,j}$, the width, height being $block\_size$ and set all the values of M in the square to be zero (see Figure 2).
7: Apply the mask: $A = A \times M$
8: Normalize the features: $A = A \times \textbf{count}(M)/\textbf{count\_ones}(M)$

a.k.a 0-1 distribution: X: p, Y: q = 1 - p

II. Regularization

F. DropBlock [2018, Ghiasi]



- **Math:**

$drop\_rate = 1 - keep\_prob$ $(all\ droped\ feature\ points)$

$drop\_rate * feat\_size^2$
$= \gamma * block\_size^2 * (feate\_size - block\_size + 1)^2$

$\Rightarrow$

$$\gamma = \frac{drop\_rate * feat\_size^2}{block\_size^2 * (feat\_size - block\_size + 1)^2}$$

$\approx$

$$\gamma = \frac{drop\_rate}{block\_size^2}\ (all\ droped\ feature\ center\ points)$$

- **Generate Mask:**
  a. Regard γ as threshold
  b. Generate mask $\in (0, 1)$,
     Crop patches with $block\_size$
     around $mask_{i,j} < \gamma$
  c. Set:
     $mask_{i,j} = 0,\ \ mask_{i,j} < \gamma$
     $mask_{i,j} = 1,\ \ mask_{i,j} \geq \gamma$

III. Activation Function

III. Activation Function

G. ReLU-Rethinking

- **Cons:**

a. **Non-differentiable at 0**

b. **Non-zero centered, zigzag when bp**

c. **0 when x<0**

III. Activation Function

G. ReLU-Rethinking

- **Cons:**

a. **Non-differentiable at 0**

b. **Non-zero centered, zigzag when bp**

$$f = \sum w_i x_i + b, \qquad x_i > 0 \ (x_i \ is \ activated)$$

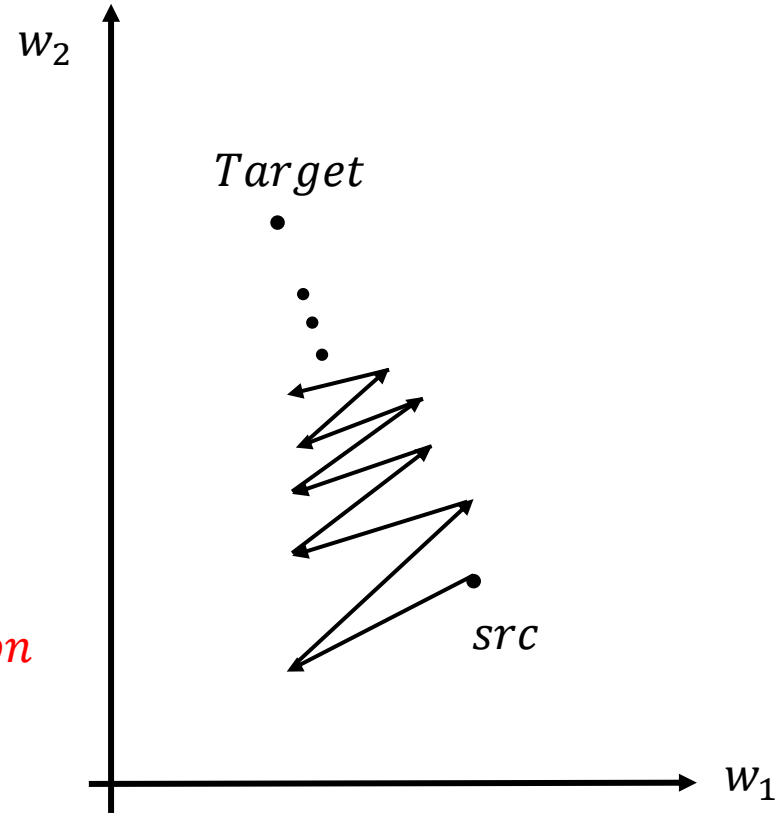
$$\frac{\partial f}{\partial w_i} = x_i > 0$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_i} = \frac{\partial L}{\partial f} x_i$$

$$\because x_i > 0$$

$$\therefore sign\left(\frac{\partial L}{\partial f}\right) = sign\left(\frac{\partial L}{\partial w_i}\right), w_i \ moves \ along \ same \ direction$$

c. **0 when x<0**



III. Activation Function

H. Swish-Improvement [2017, Ramachandran]

- **Name:**

  Swish: Self-gated activation function

- **Form:**

  $f(x) = x \cdot \sigma(x)$

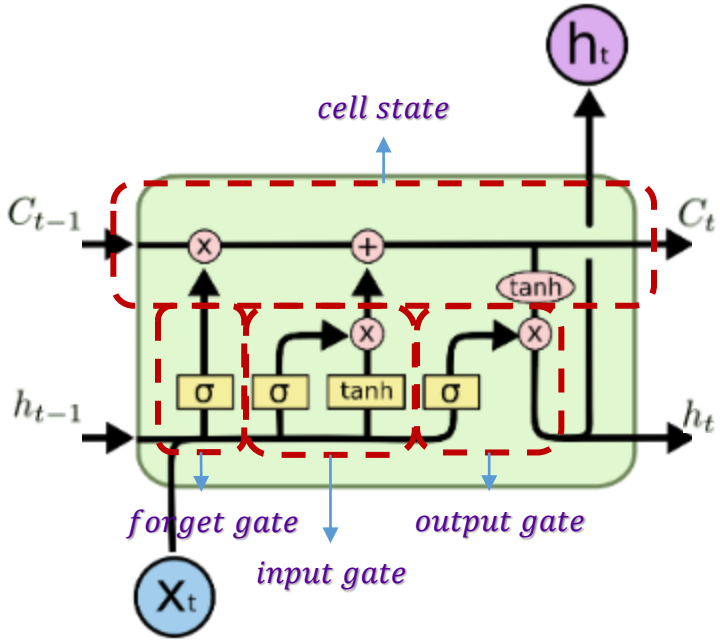
  $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

- **Alternative:**

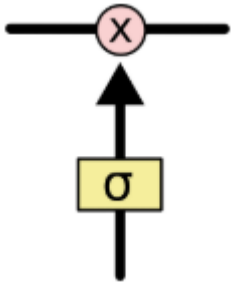
  $f(x;\beta) = 2x \cdot \sigma(\beta x)$ — swish-β

- **Inspired by:**

  LSTM's gating idea
  (a good explanation)



gate: $C_{t-1} \times \sigma(f(h_{t-1}))$



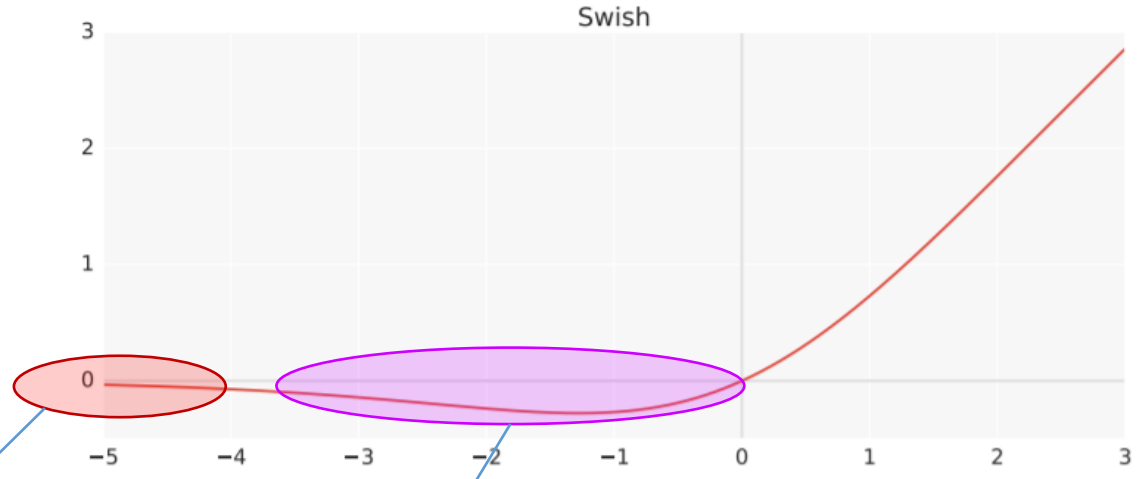
III. Activation Function

H. Swish-Improvement [2017, Ramachandran]

- **Derivative:**

$$f(x) = x \cdot \sigma(x)$$

$$f'(x)$$
$$= \sigma(x) + x \cdot \sigma(x) \cdot \big(1 - \sigma(x)\big)$$
$$= x \cdot \sigma(x) + \sigma(x)\big(1 - x \cdot \sigma(x)\big)$$
$$= f(x) + \sigma(x) \cdot (1 - f(x))$$



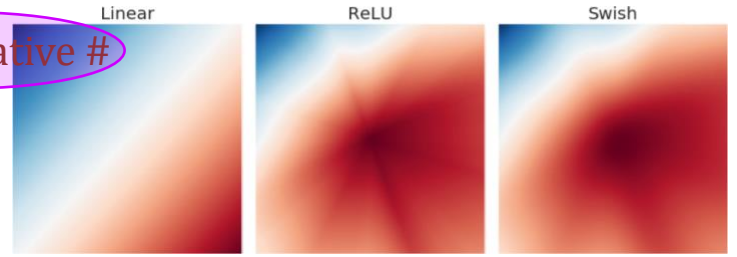
- **Properties:**

➢ Unbounded above
Avoid saturation

➢ Bounded below
Strong regularization
(especially for large negative numbers)

➢ Non-monotonicity
No died neuron for small negative #

➢ Smoothness
Easier to train:
less sensitive to init. & LR



III. Activation Function

I. Mish - Perhaps Best Now [2019, Misra]

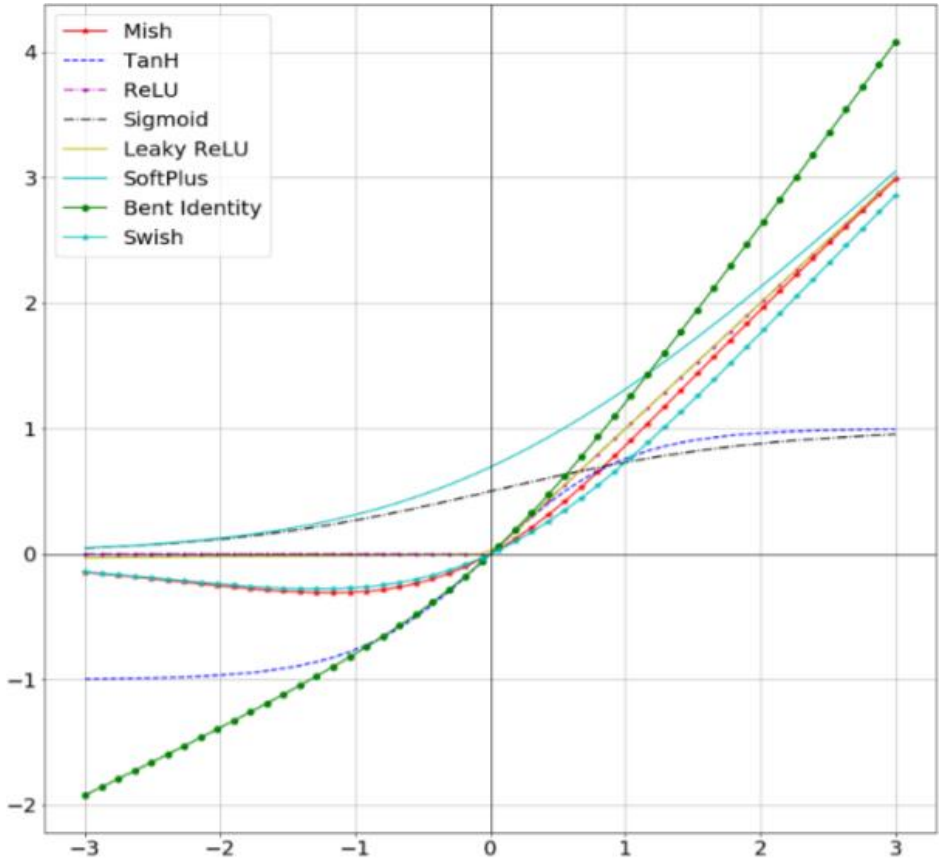
- **Form:**

$$f(x) = x \cdot tanh(\varsigma(x))$$
$$\varsigma(x) = ln(1 + e^x) \text{ —— } softplus$$

- **Derivative:**

$$f'(x) = \frac{e^x \omega}{\delta^2}$$
$$\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^x(4x+6)$$
$$\delta = 2e^x + e^{2x} + 2$$

- **Properties:**
  - Same with Swish
  - Slower than ReLU
  - Cannot 100% guarantee
  - Could pair with Ranger Optimizer (2019, Hinton)



IV. Loss

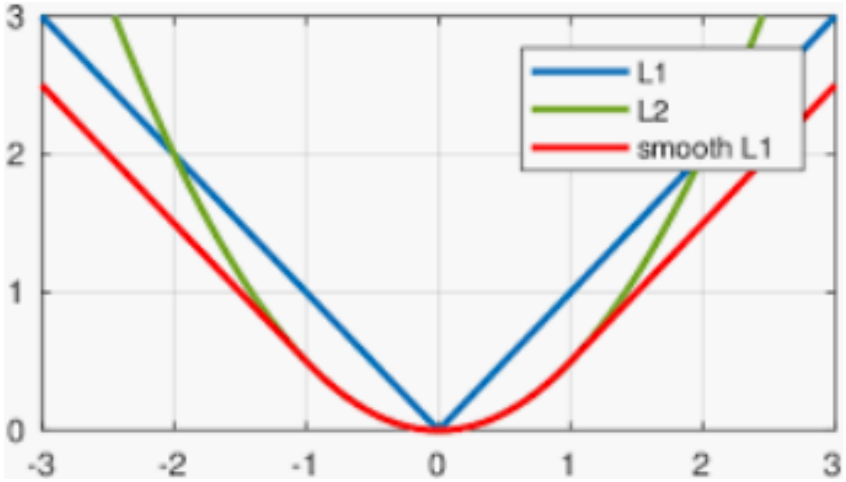
IV. Loss

J. L1, L2 & Smooth L1 Loss

- **L1 Loss (MAE: Mean Absolute Error):**
  - Main problem:
    derivative = C,
    hard to converge with high accuracy
    shake around optimal value

- **L2 Loss (MSE: Mean Squared Error):**
  - Main problem:
    derivative's changing,
    not stable, especially at early stage
    (the greater the loss, the greater the d)
    slower and slower

- **Smooth L1 Loss:**
  - Target:
    Overcome those problems
  - Huber Loss (1964)
    Similar to Huber Loss (Discussion)



IV. Loss

J. L1, L2 & Smooth L1 Loss

- **L1 Loss (MAE: Mean Absolute Error):**
  - Main problem:
    derivative = C,
    hard to converge with high accuracy
    shake around optimal value

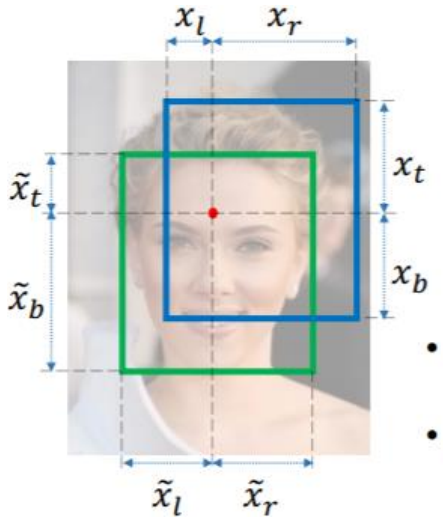
- **L2 Loss (MSE: Mean Squared Error):**
  - Main problem:
    derivative's changing,
    not stable, especially at early stage
    (the greater the loss, the greater the d)
    slower and slower

- **Smooth L1 Loss:**
  - Target:
    Overcome those problems
  - Huber Loss (1964)
    Similar to Huber Loss (Discussion)

- **Summary:**
  - Main problems to detect bboxes:
    1. Coordinates participate in computing separately
    2. Different predicted bboxes have same losses

IV. Loss

K. IoU Loss [2016, UnitBox, Yu]



Ground truth: $\tilde{x} = (\tilde{x}_t, \tilde{x}_b, \tilde{x}_l, \tilde{x}_r)$

Prediction: $x = (x_t, x_b, x_l, x_r)$

- $\ell_2 \ \ loss = ||\square - \square||_2^2$
- $IoU\ loss = -\ln\frac{Intersection(\square, \square)}{Union(\square, \square)}$

not required

Algorithm 1: *IoU* loss Forward

Input: $\tilde{\boldsymbol{x}}$ as bounding box ground truth
Input: $\boldsymbol{x}$ as bounding box prediction
Output: $\mathcal{L}$ as localization error
for *each pixel* (i, j) **do**
  if $\tilde{\boldsymbol{x}} \neq \mathbf{0}$ **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

IV. Loss

K. IoU Loss [2016, UnitBox, Yu]

- **Derivative:**

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

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

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

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

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

How to make your own loss function?

Any Problems?

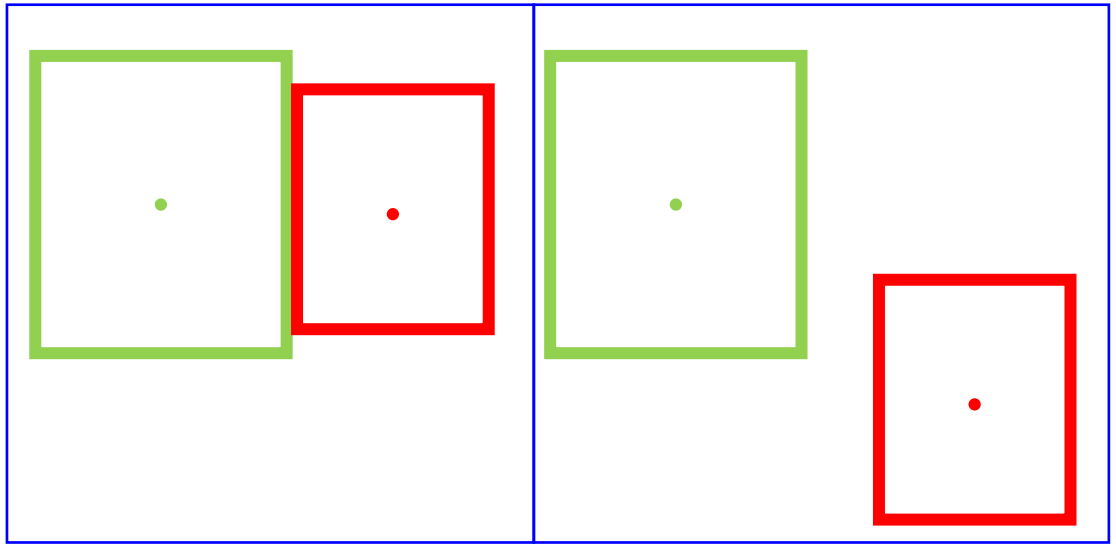
Algorithm 1: *IoU* loss Forward

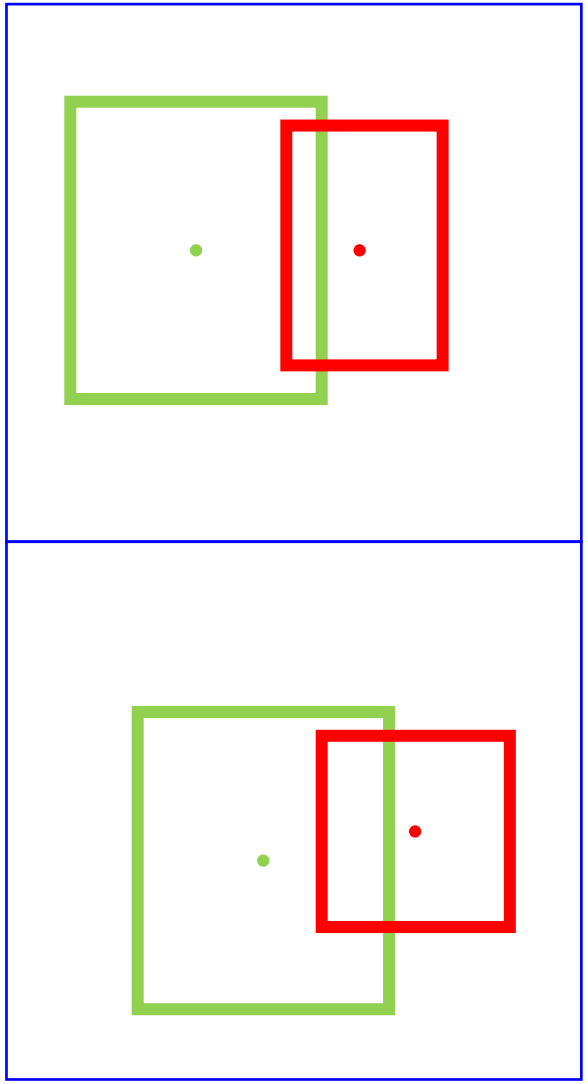
Input: $\widetilde{\boldsymbol{x}}$ as bounding box ground truth
Input: $\boldsymbol{x}$ as bounding box prediction
Output: $\mathcal{L}$ as localization error
for *each pixel* (i, j) **do**
  if $\widetilde{\boldsymbol{x}} \neq \mathbf{0}$ **then**
    $X = (x_t + x_b) * (x_l + x_r)$
    $\widetilde{X}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

IV. Loss

K. IoU Loss [2016, UnitBox, Yu]

- **Problems:**
  - What if $x, \tilde{x}$ has no intersection?
  - Intersections have different shapes with same area.





IV. Loss

L. GIoU Loss [2019]

input : Predicted B^p and ground truth B^g bounding box coordinates:

$$B^p = (x_1^p, y_1^p, x_2^p, y_2^p), \quad B^g = (x_1^g, y_1^g, x_2^g, y_2^g).$$

output: $\mathcal{L}_{IoU}$, $\mathcal{L}_{GIoU}$.

1 For the predicted box B^p, ensuring $x_2^p > x_1^p$ and $y_2^p > y_1^p$:

$$\hat{x}_1^p = \min(x_1^p, x_2^p), \quad \hat{x}_2^p = \max(x_1^p, x_2^p),$$
$$\hat{y}_1^p = \min(y_1^p, y_2^p), \quad \hat{y}_2^p = \max(y_1^p, y_2^p).$$

2 Calculating area of B^g: $A^g = (x_2^g - x_1^g) \times (y_2^g - y_1^g)$.

3 Calculating area of B^p: $A^p = (\hat{x}_2^p - \hat{x}_1^p) \times (\hat{y}_2^p - \hat{y}_1^p)$.

4 Calculating intersection $\mathcal{I}$ between B^p and B^g:

$$x_1^{\mathcal{I}} = \max(\hat{x}_1^p, x_1^g), \quad x_2^{\mathcal{I}} = \min(\hat{x}_2^p, x_2^g),$$
$$y_1^{\mathcal{I}} = \max(\hat{y}_1^p, y_1^g), \quad y_2^{\mathcal{I}} = \min(\hat{y}_2^p, y_2^g),$$
$$\mathcal{I} = \begin{cases} (x_2^{\mathcal{I}} - x_1^{\mathcal{I}}) \times (y_2^{\mathcal{I}} - y_1^{\mathcal{I}}) & \text{if} \quad x_2^{\mathcal{I}} > x_1^{\mathcal{I}}, y_2^{\mathcal{I}} > y_1^{\mathcal{I}} \\ 0 & \text{otherwise.} \end{cases}$$

5 Finding the coordinate of smallest enclosing box B^c:

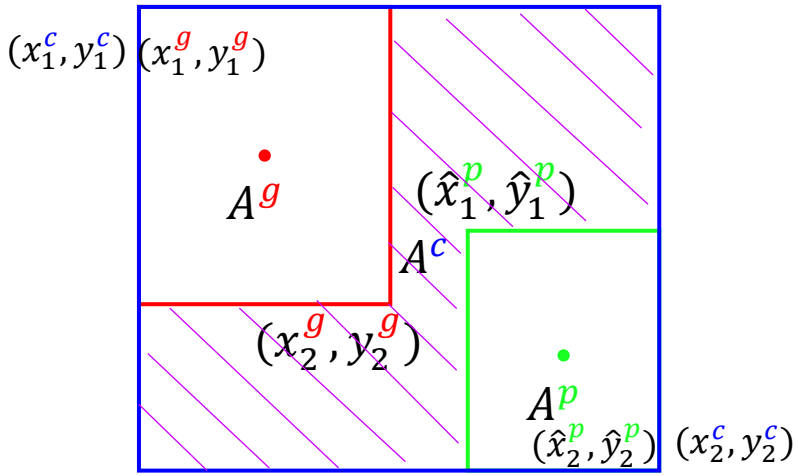
$$x_1^c = \min(\hat{x}_1^p, x_1^g), \quad x_2^c = \max(\hat{x}_2^p, x_2^g),$$
$$y_1^c = \min(\hat{y}_1^p, y_1^g), \quad y_2^c = \max(\hat{y}_2^p, y_2^g).$$

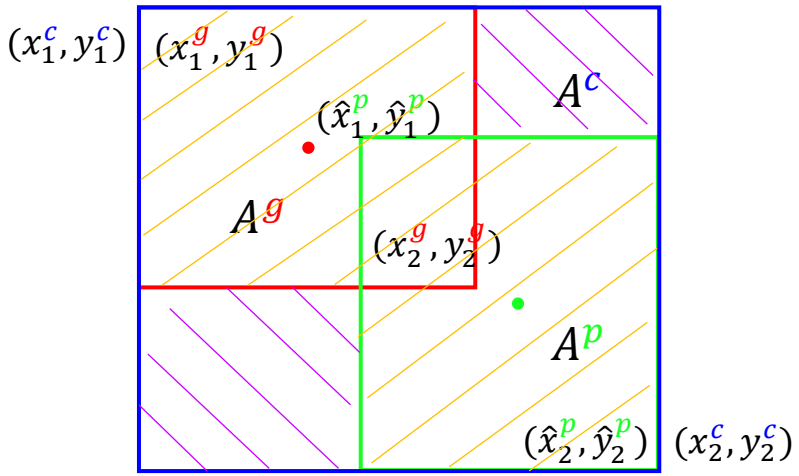
6 Calculating area of B^c: $A^c = (x_2^c - x_1^c) \times (y_2^c - y_1^c)$.

7 $IoU = \frac{\mathcal{I}}{\mathcal{U}}$, where $\mathcal{U} = A^p + A^g - \mathcal{I}$.

8 $GIoU = IoU - \frac{A^c - \mathcal{U}}{A^c}$.

9 $\mathcal{L}_{IoU} = 1 - IoU, \quad \mathcal{L}_{GIoU} = 1 - GIoU.$

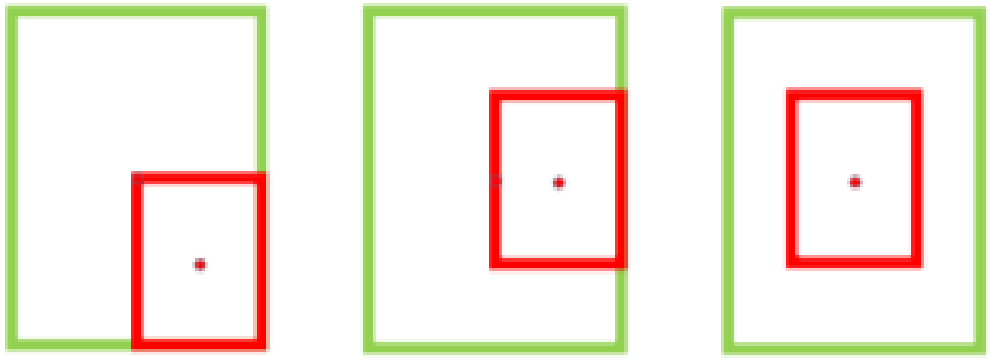




IV. Loss

L. GIoU Loss [2019, Rezatofighi]

- **Problems:**
  - What if intersections have same area localized in different position



$\mathcal{L}_{IoU} = 0.75$ $\mathcal{L}_{IoU} = 0.75$ $\mathcal{L}_{IoU} = 0.75$

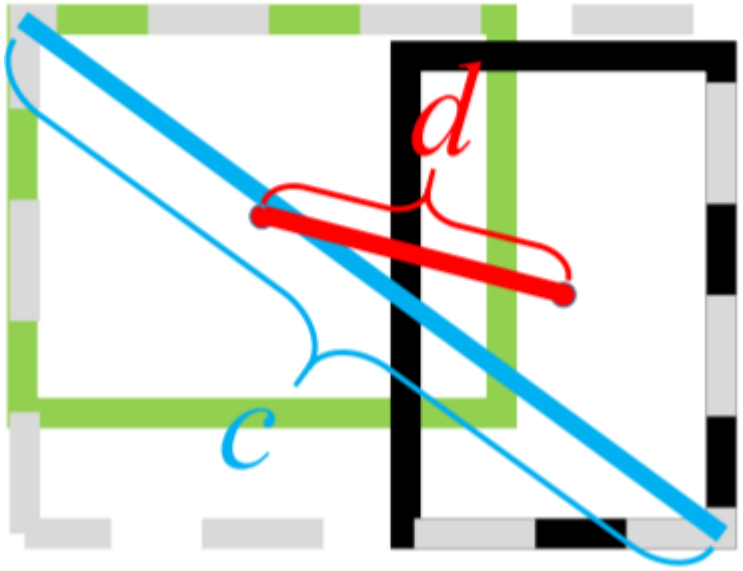
$\mathcal{L}_{GIoU} = 0.75$ $\mathcal{L}_{GIoU} = 0.75$ $\mathcal{L}_{GIoU} = 0.75$

$\mathcal{L}_{DIoU} = 0.81$ $\mathcal{L}_{DIoU} = 0.77$ $\mathcal{L}_{DIoU} = 0.75$

IV. Loss

M. DIoU Loss [2019, Zheng]

- **Solution:**



- **Equations:**

$$\mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} \quad \text{(Penalty Term)}$$

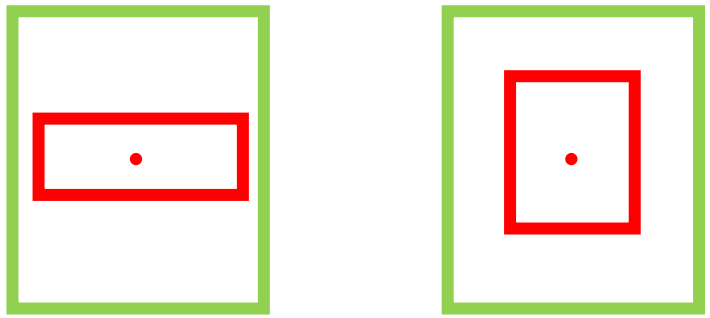
$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%

IV. Loss

M. DIoU Loss [2019, Zheng]

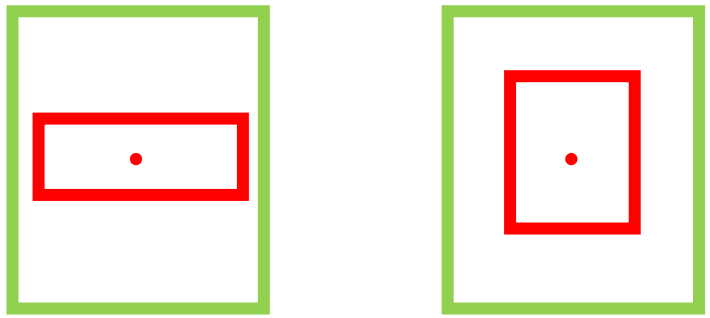
- **Problems:**
  - What if intersections have same area
    localized at same position (or share same radius)



IV. Loss

M. DIoU Loss [2019, Zheng]

- **Problems:**
  - What if intersections have same area localized at same position (or share same radius)



- **Principles:**
  - Overlapped area
  - Central point distances
  - Aspect ratio (shape)!

IV. Loss

N. CIoU Loss [2019, Zheng]

- **Equations:**

$$\mathcal{R}_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v$$

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

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2$$

$$\alpha = \frac{v}{(1 - IoU) + v}$$

The more IoU is, the more the α will be, the more important the v will be

IV. Loss

N. CIoU Loss [2019, Zheng]

- **Results:**

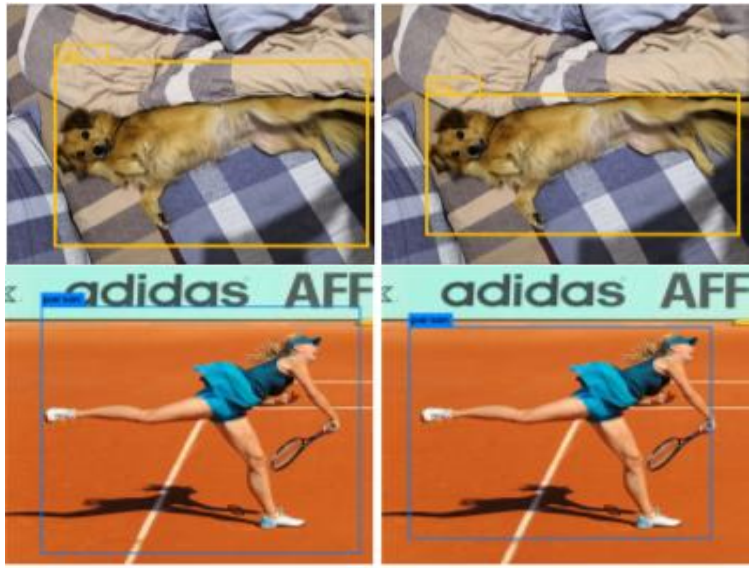
Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
$\mathcal{L}_{CIoU}$	**49.21**	**48.42**	**54.28**	**52.87**
Relative improv. %	**5.67%**	**5.67%**	**8.95%**	**8.43%**

IV. Loss

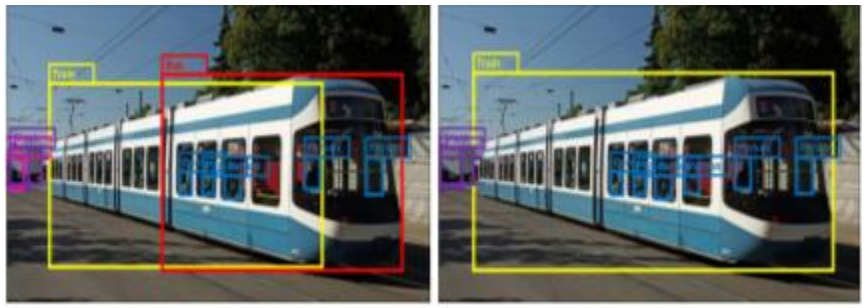
N. CIoU Loss [2019, Zheng]

- **DIoU-NMS:**

$$s_i = \begin{cases} s_i, & IoU - \mathcal{R}_{DIoU}(\mathcal{M}, B_i) < \varepsilon \\ 0, & IoU - \mathcal{R}_{DIoU}(\mathcal{M}, B_i) \geq \varepsilon \end{cases}$$



$\mathcal{L}_{GIoU}$ $\mathcal{L}_{CIoU}$



$\mathcal{L}_{GIoU}$ $\mathcal{L}_{CIoU}$

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
$\mathcal{L}_{CIoU}$	**49.21**	**48.42**	**54.28**	**52.87**
Relative improv. %	**5.67%**	**5.67%**	**8.95%**	**8.43%**
$\mathcal{L}_{CIoU}(D)$	**49.32**	**48.54**	**54.74**	**53.30**
Relative improv. %	**5.91%**	**5.94%**	**9.88%**	**9.31%**