TPH-YOLOv5++: Boosting Object Detection on Drone-Captured Scenarios with Cross-Layer Asymmetric Transformer

Qi Zhao[1], Binghao Liu[1], Shuchang Lyu[1], Chunlei Wang[1], and Hong Zhang[2]

[1]Department of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2]School of Astronautics, Beihang University, Beijing 100191, China

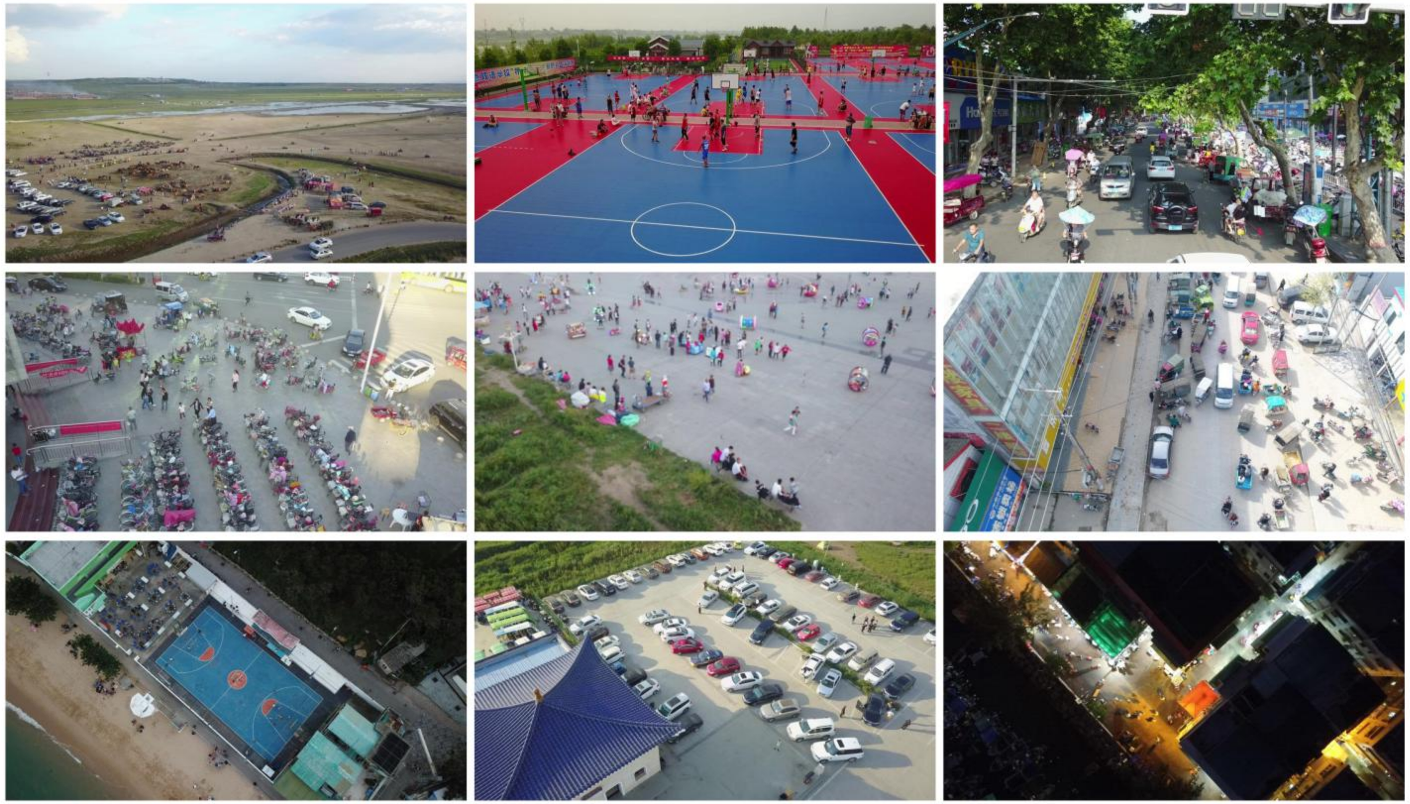
Presenter: Nguyen Duy Linh
ndlinh301@mail.ulsan.ac.kr
Nov. 18, 2023

** Published on Remote Sensing Journal*
** Published: 21 March 2023*

Motivation

- **The three main problems in object detection in drone-captured images:**
  - (a) Size variation
  - (b) High-density
  - (c) Large coverage of objects on drone-captured images

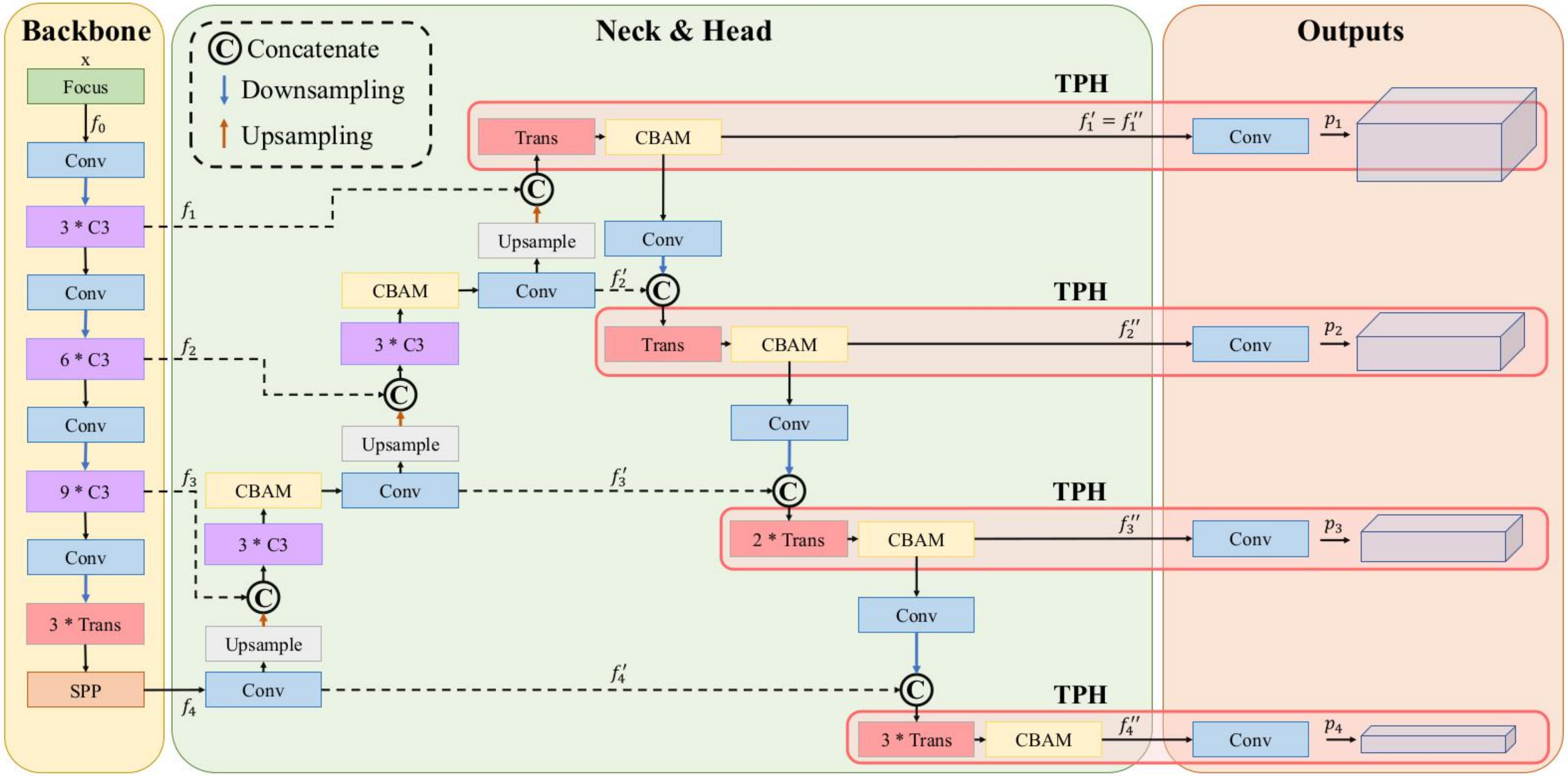
(a)
(b)
(c)



Motivation

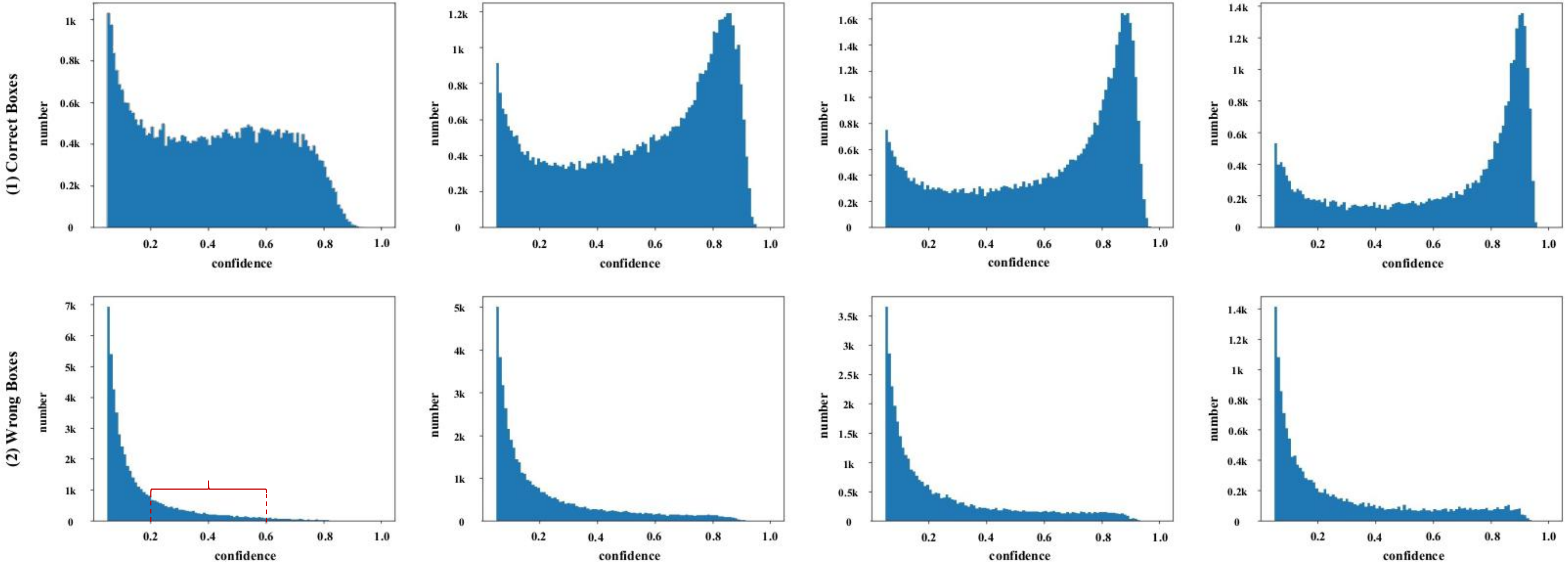
TPH-YOLOv5 architecture:

- TPH-YOLOv5 introduces an additional head, Transformer prediction head (TPH), and convolutional block attention module (CBAM).
- Four prediction heads are named tiny, small, medium, and large heads.



Problems of TPH-YOLOv5:

- **Problems at Tiny Prediction Head:** It produces plenty of wrong boxes with relatively large confidence, especially between 0.2 and 0.6.

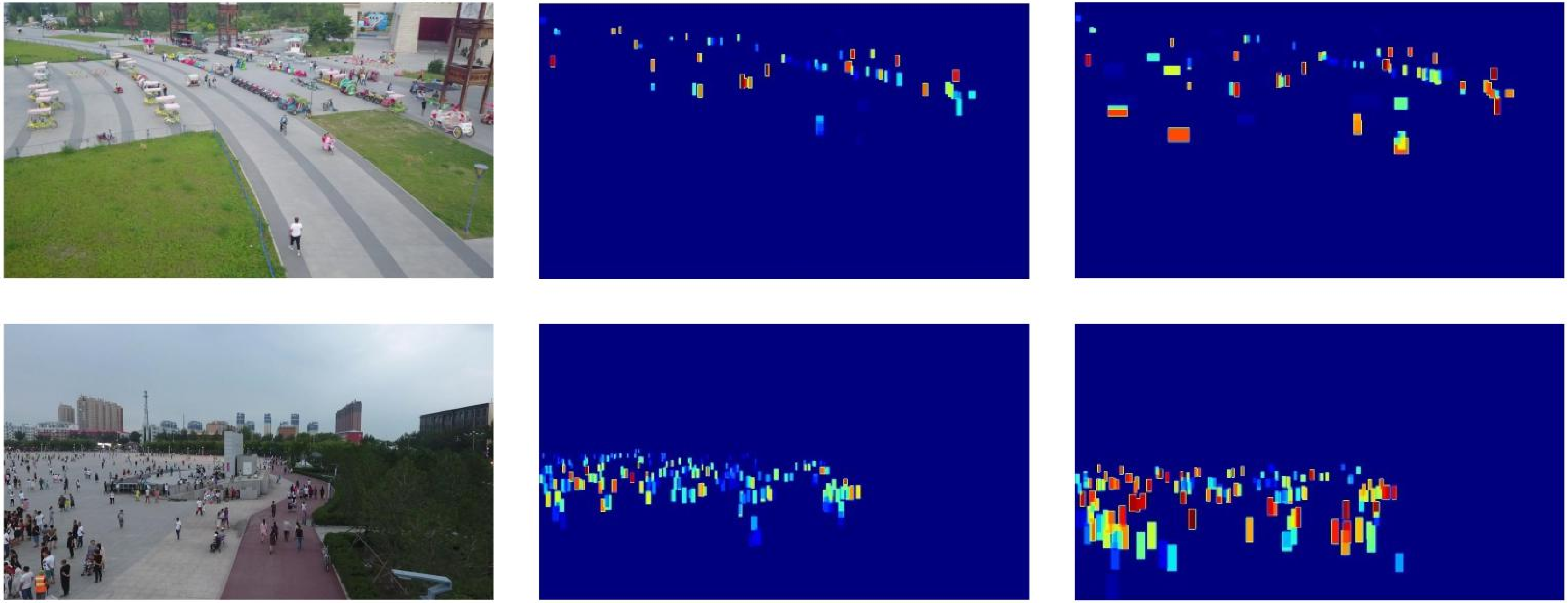


(a) Tiny Prediction Head (b) Small Prediction Head (c) Medium Prediction Head (d) Large Prediction Head

Motivation

Problems of TPH-YOLOv5:

- **Problems at Small Prediction Head:** the Small Prediction Head also captures objects that are contained by the results predicted by the additional Tiny Prediction Head

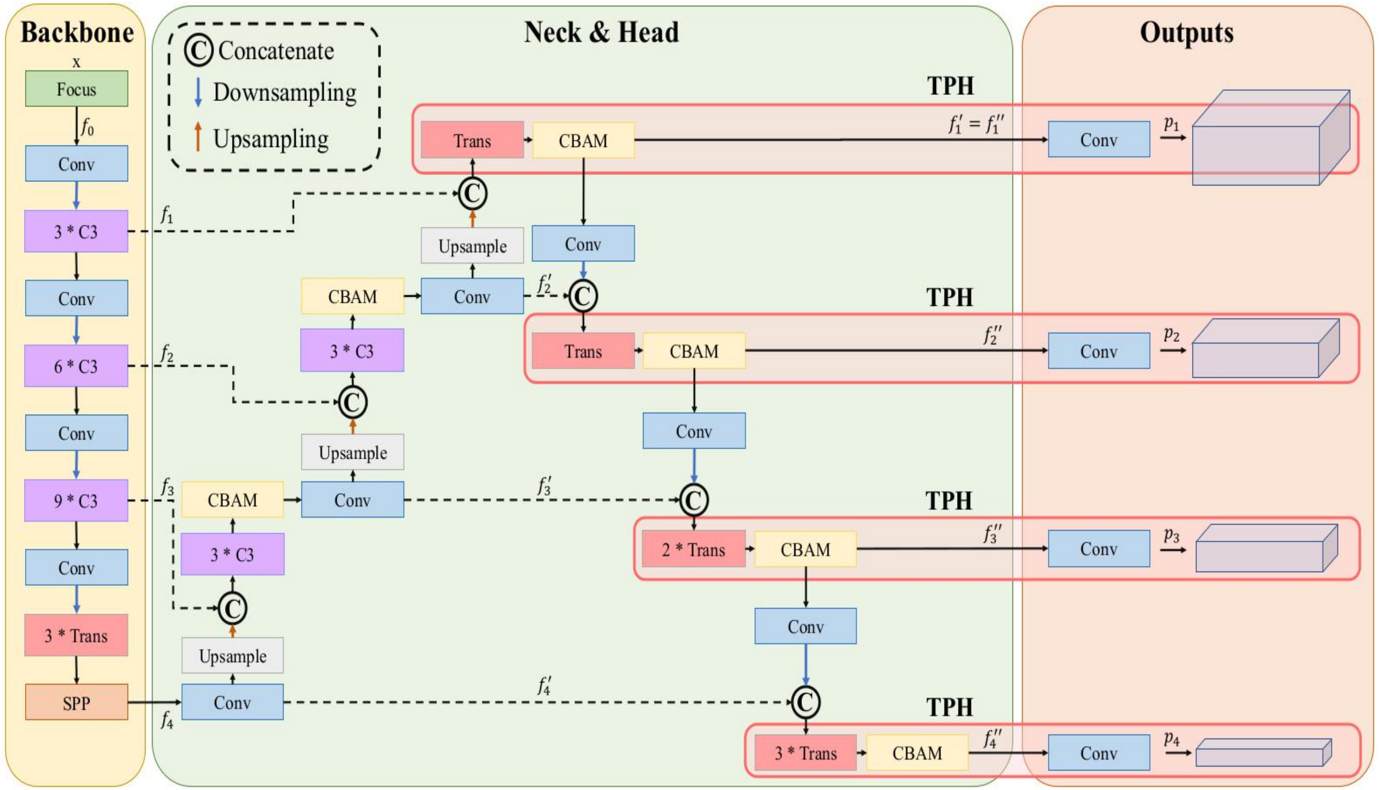


(a) Original Image (b) Tiny Prediction Head (c) Small Prediction Head

Contributions

- **The new contributions in the paper:**
  - TPH-YOLOv5++ is proposed to significantly **reduce the computational cost and improve the detection speed** of TPH-YOLOv5.
  - **Cross-layer Asymmetric Transformer (CA-Trans)** is designed to replace the additional prediction head while maintain the knowledge of this head.
  - By using a **Sparse Local Attention (SLA)** module, the asymmetric information between the additional head and other heads can be captured efficiently, enriching the features of other heads.

TPH-YOLOv5 Architecture



$$f_i = B_i(f_{i-1}), \quad i = 1, \cdots, 4$$

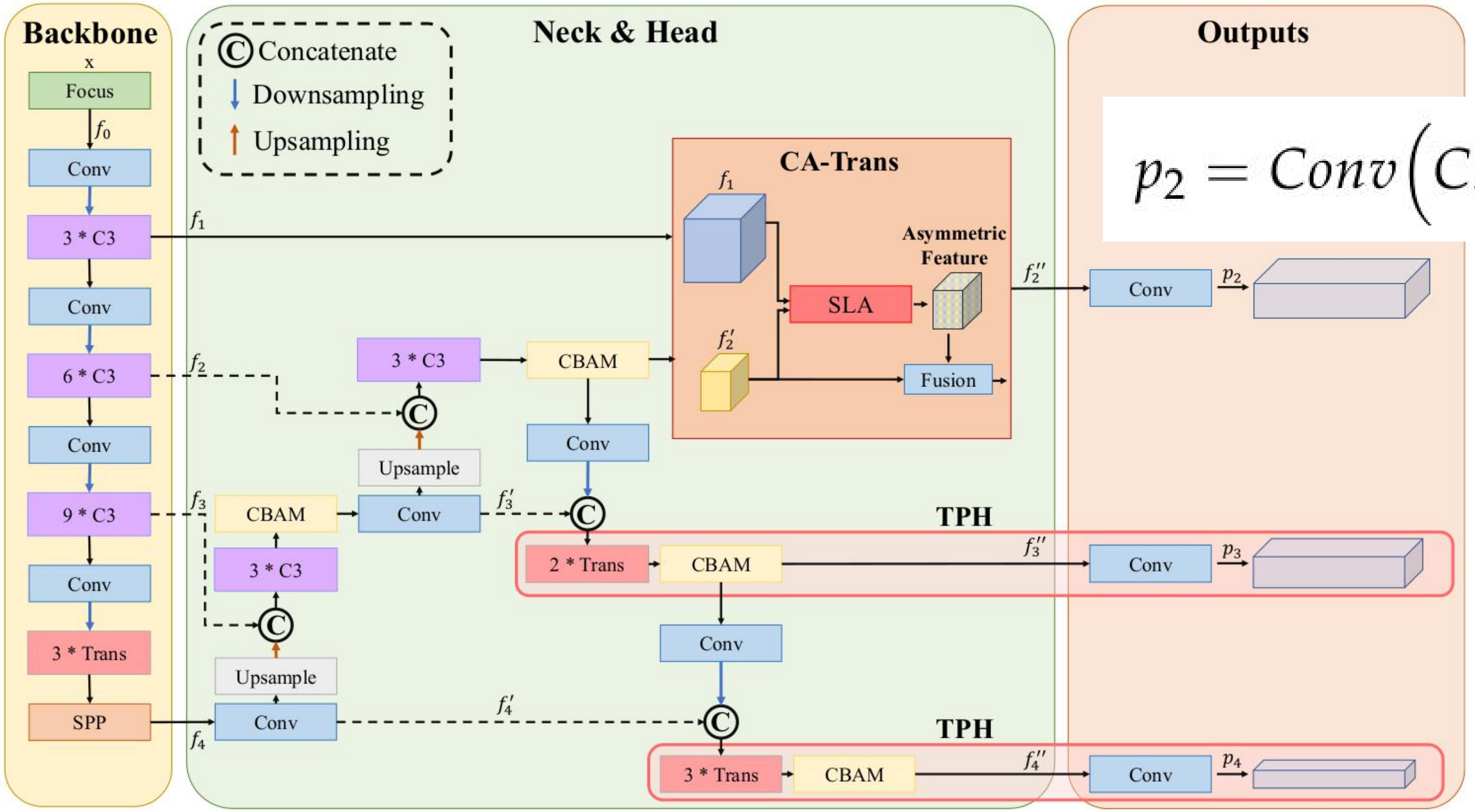
$$f'_i = \begin{cases} N_i\left(f_i, f'_{i+1}\right), & i = 1,2,3 \\ Conv(f_4), & i = 4 \end{cases}$$

$$N_i\left(f_i, f'_{i+1}\right) = UpBlock\left(Concat\left(f_i, Upsampling\left(f'_{i+1}\right)\right)\right)$$

$$f''_i = \begin{cases} f'_1, & i = 1 \\ H_i\left(f'_i, f''_{i-1}\right), & i = 2,3,4 \end{cases}$$

$$H_i\left(f'_i, f''_{i-1}\right) = DownBlock\left(Concat\left(f'_i, Conv\left(f''_{i-1}\right)\right)\right)$$

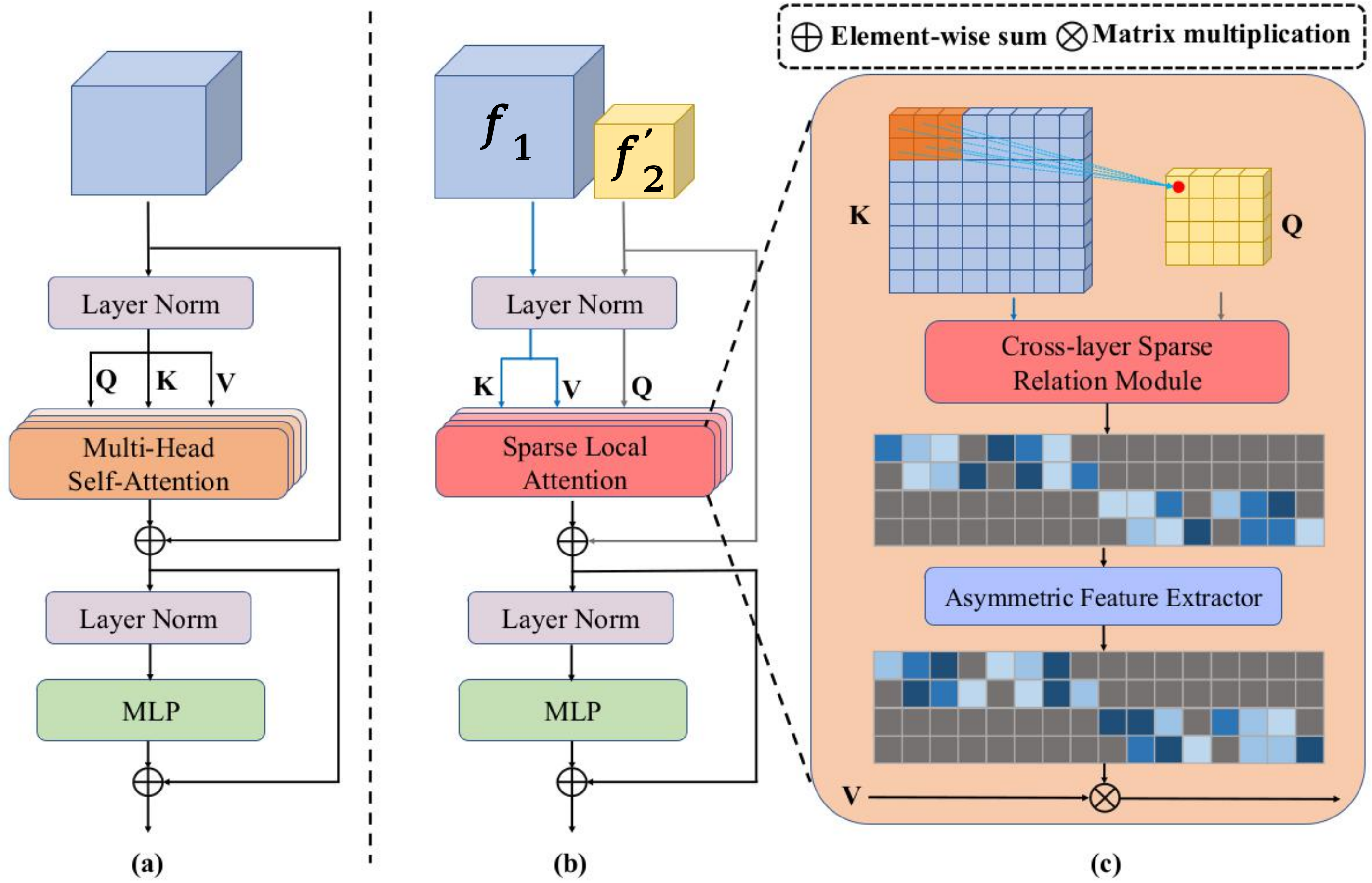
$$p_i = Conv\left(f''_i\right), \quad i = 1, \cdots, 4$$



$$p_2 = Conv\left(CA - Trans\left(f_1, f_2''\right)\right)$$

TPH-YOLOv5++ Architecture

Cross-layer Asymmetric Transformer (CA-Trans)



Algorithm 1: Sparse Local Attention

Input: input feature maps Q, K_p, V

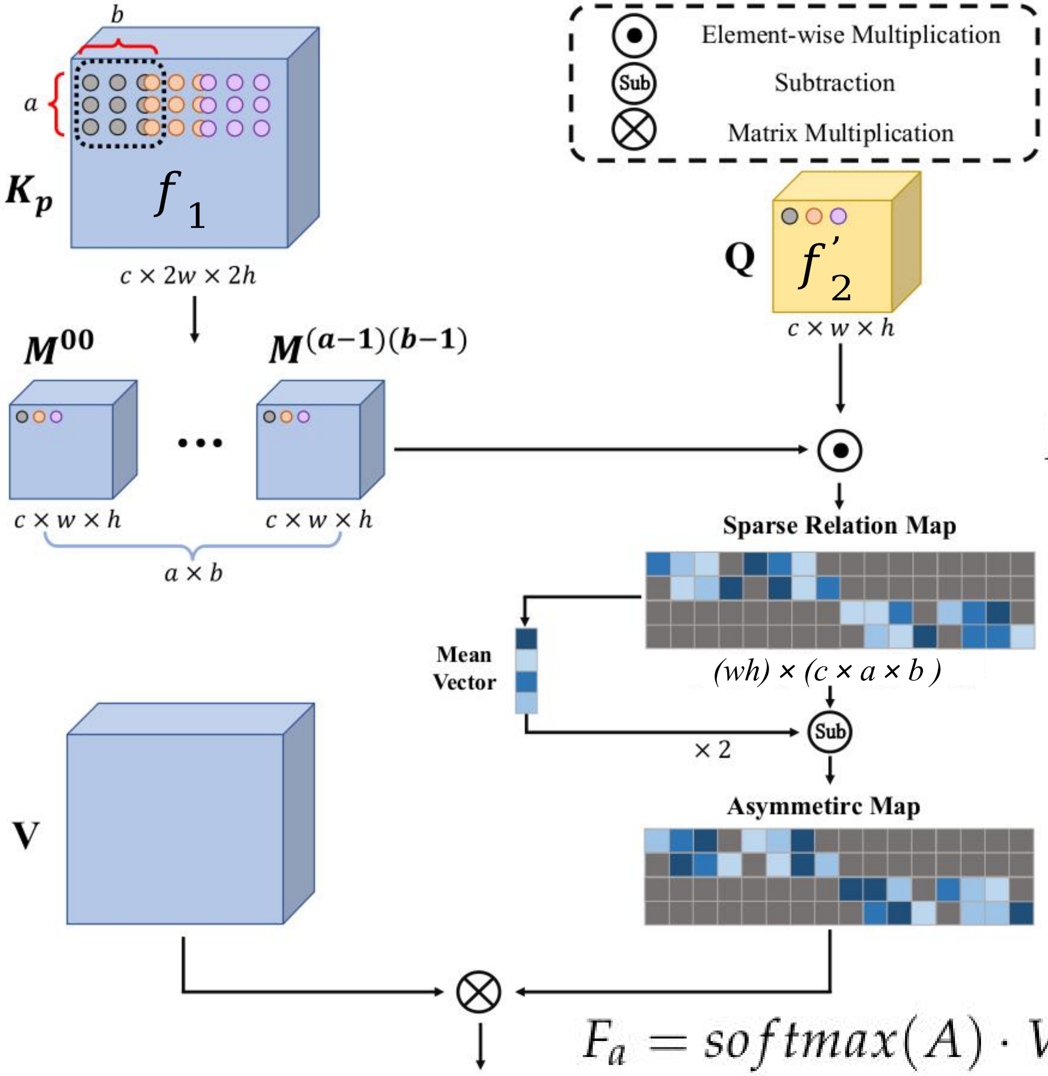
Output: output F_{out}

for i *from* 0 *to* $a-1$ **do**
  for j *from* 0 *to* $b-1$ **do**
    select the (i, j) point in neighborhood of each pixel in Q;
    build the neighborhood feature M^{ij};
  end
end
concatenate all neighborhood features to generate K_{sparse};
$R \longleftarrow \frac{K_{sparse} \cdot Q}{\sqrt{d_Q}}$;
for *each row* r *of* R **do**
  calculate the average $\bar{R}_r$;
  $A(r, l) \longleftarrow 2 \cdot \bar{R}_r - R(r, l)$;
end
build the asymmetric map A;
$F_a \longleftarrow softmax(A) \cdot V$;
return F_a;

Figure 7. Overview of the CA-Trans module. (**a**) is the vanilla ViT module that generates Q, K, and V from a single feature map and uses multi-head self-attention (MHSA) to obtain attentions. (**b**) shows the architecture of our CA-Trans, where K and V are generated from f_1, while Q is generated from f'_2. Otherwise, we replace the MHSA with the sparse local attention (SLA) to extract attentions between two different layers. (**c**) is the SLA. By introducing the cross-layer sparse relation module (CSRM) and asymmetric feature extractor (AFE), our CA-Trans can efficiently extract asymmetric information between two paths and enrich the features of small paths.

TPH-YOLOv5++ Architecture

Sparse Local Attention (SLA)



$$M = \left\{ M^{ij} \mid neighborhood\ feature\ of\ pixel\ (i,j),\ i = 0, \cdots, a-1,\ \ j = 0, \cdots, b-1 \right\}$$

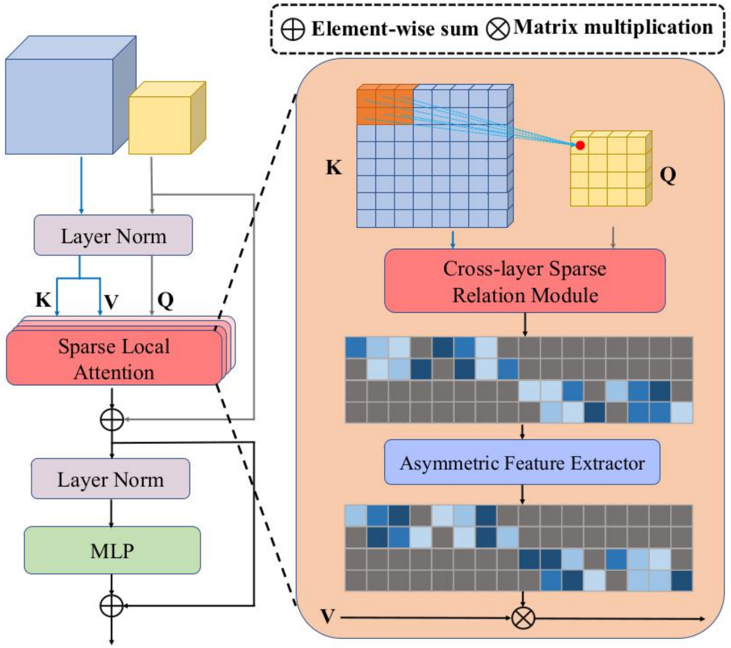
$$M_{ij}^{(u,v)} = K_p(2u + i, 2v + j)$$

$$K_{sparse} = Concat\left(M^{00}, M^{01}, \cdots, M^{(a-1)(b-1)} \right)$$

$$R = \frac{K_{sparse} \cdot Q}{\sqrt{d_Q}}$$

$$A(i,j) = 2 \cdot \bar{R}_i - R(i,j)$$

$$F_a = softmax(A) \cdot V$$



$$F_{out} = f_2' + F_a + MLP\left(LN\left(f_2' + F_a \right) \right)$$

- **Dataset**
  - **VisDrone2021 dataset**
    - Trainset: 6,471 images
    - Validation set: 548 images
    - Test-dev: 1,610 images
    - Test-challenge: 1,850 images
    - Evaluation: validation set and testset-dev

Experiments

- **Dataset**
  - **UAVDT (Unmanned Aerial Vehicle Benchmark Object Detection and Tracking) dataset**
    - 40,376 images (50 videos)
    - Trainset: 24,778 images (31 videos)
    - Testing set: 15,598 (19 videos)

Implementation Details

- PyTorch
- An NVIDIA RTX3090Ti GPU for training and testing
- Pre-trained model from YOLOv5x
- Adam optimizer
- Use 0.0003 as the initial learning rate with the cosine lr schedule
- **On VisDrone2021:**
  - Epoch: 65 epochs
  - Batch size: 2
  - Input images: 1536 x 1536
- **On UAVDT:**
  - Epoch: 30 epochs
  - Batch size: 4
  - Input images: 1024 x 1024

Experiments

Comparisons with the State-of-the-art

On VisDrone 2021 test-challenge dataset

Method	AP [%]	AP50 [%]	AP75 [%]	AR1 [%]	AR10 [%]	AR100 [%]	AR500 [%]
DBNet(A.1)	**39.43**	**65.34**	41.07	0.29	2.03	12.13	55.36
SOLOer(A.2)	39.42	63.91	40.87	1.75	10.94	44.69	55.91
Swin-T(A.3)	39.40	63.91	40.87	1.76	10.96	44.65	56.83
TPH-YOLOv5(A.4)	39.18	62.83	**41.34**	**2.61**	**13.63**	**45.62**	**56.88**
VistrongerDet(A.5)	38.77	64.28	40.24	0.77	8.10	43.23	55.12
cascade++(A.6)	38.72	62.92	41.05	1.04	6.69	43.36	43.36
DNEFS(A.7)	38.53	62.86	40.19	1.42	9.38	43.10	54.87
EfficientDet(A.8)	38.51	63.25	39.54	1.82	11.12	43.89	55.12
DPNet-ensemble	37.37	62.05	39.10	0.85	7.96	42.03	53.78
DroneEye2020	34.57	58.21	35.74	0.28	1.92	6.93	52.37
Cascade R-CNN	16.09	31.94	15.01	0.28	2.79	21.37	28.43

Comparisons with the State-of-the-art

On VisDrone 2021 validation dataset

Method	AP [%]	AP50 [%]	AP75 [%]
ClusDet [32]	28.4	53.2	26.4
Zhang et al. [33]	30.3	58.0	27.5
GLSAN [35]	32.5	55.8	33.0
DMNet [36]	29.4	49.3	30.6
DSHNet [37]	30.3	51.8	30.9
HawkNet [41]	25.6	44.3	25.8
CDMNet [42]	31.9	52.9	33.2
DCRFF [60]	35.0	57.0	29.5
UFPMP-Net [49]	39.2	**65.3**	40.2
TPH-YOLOv5	**42.1**	63.1	**45.7**
TPH-YOLOv5++	41.4	61.9	45.0

Comparisons with the State-of-the-art

On VisDrone 2021 test-dev dataset

Method	AP [%]	AP50 [%]	AP75 [%]
GDFNet [45]	18.7	31.7	19.4
VistrongerDet [62]	33.85	57.27	34.81
ViT-YOLO [61]	**38.5**	**63.2**	**40.5**
TPH-YOLOv5	34.4	54.5	36.5
TPH-YOLOv5++	33.5	52.5	35.7

Comparisons with the State-of-the-art

On UAVDT

Method	AP [%]	AP50 [%]	AP75 [%]
ClusDet [32]	13.7	26.5	12.5
Zhang et al. [33]	17.7	-	-
GDFNet [45]	15.4	26.1	17.0
GLSAN [35]	19.0	30.5	21.7
DMNet [36]	14.7	24.6	16.3
DSHNet [37]	17.8	30.4	19.7
CDMNet [42]	20.7	35.5	22.4
SODNet [48]	17.1	29.9	18.0
UFPMP-Net [49]	24.6	38.7	28.0
TPH-YOLOv5	26.9	41.3	32.7
TPH-YOLOv5++	**30.1**	**43.5**	**34.3**

Ablation Study on VisDrone2021 Test-Dev Set

Methods	AP [%]	AP50 [%]	AP75 [%]	GPU Memory	GFLOPs	FPS
YOLOv5x	28.9	45.4	30.8	4279 M	200.2	13.68
YOLOv5x+p2	31.0	48.7	32.9	4667 M	241.2	10.89
YOLOv5x+p2+ViT	32.8	52.0	34.8	5103 M	244.5	9.51
TPH-YOLOv5 (previous+CBAM)	33.6	53.2	35.8	5105 M	245.1	8.22
TPH-YOLOv5 (SwinTrans+CBAM)	34.0	53.2	35.8	4977 M	315.4	7.36
TPH-YOLOv5++	33.1	52.1	35.1	4715 M	207.0	11.86

Ablation Study on UAVDT

Methods	AP [%]	AP50 [%]	AP75 [%]	GPU Memory	GFLOPs	FPS
TPH-YOLOv5	26.9	41.3	32.7	3631 M	556.6	25.12
TPH-YOLOv5++	30.1	43.5	34.3	3361 M	293.2	42.19

Ablation Study for Each Category on VisDrone2021 Test-Dev Set

Methods	All	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	Awning-Tricycle	Bus	Motor
YOLOv5x	28.9	23.5	14.3	13.5	51.8	35.4	38.0	20.2	19.9	48.6	23.8
YOLOv5x+p2	31.0	25.6	14.9	14.3	56.2	37.4	40.1	22.0	21.5	52.5	25.3
YOLOv5x+p2+ViT	32.8	26.7	16.0	15.5	59.1	40.0	42.7	23.4	22.2	55.4	27.1
TPH-YOLOv5 (previous+CBAM)	33.6	27.4	16.3	15.9	61.4	41.9	43.3	23.9	21.5	56.9	27.8
TPH-YOLOv5 (SwinTrans+CBAM)	34.0	27.5	16.1	15.9	61.7	41.9	43.9	24.2	24.0	56.4	28.5
TPH-YOLOv5+ms-testing	34.9	28.8	16.3	15.0	65.9	44.3	43.8	25.7	22.8	59.0	27.1

Ablation studies

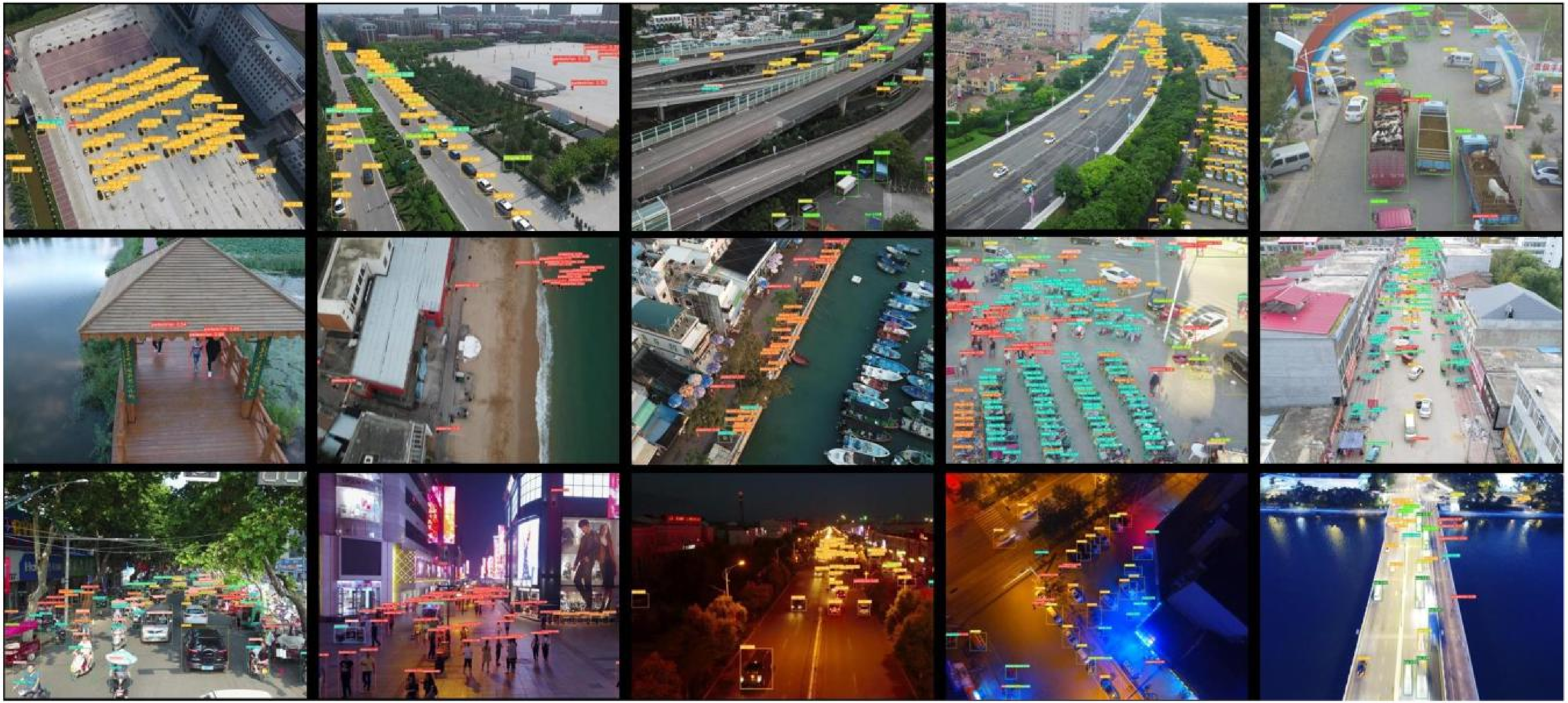
Comparisons with the State-of-the-art

Ablation Study of Neighborhood Size

	Neighborhood Size (a, b)			
	$a = 0, b = 0$	$a = 2, b = 2$	$a = 4, b = 4$	$a = 6, b = 6$
AP [%]	31.9	33.1	33.5	33.6
AP50 [%]	51.7	52.1	52.5	52.6
AP75 [%]	33.5	35.1	34.9	35.0
GPU Memory	4299 M	4715 M	7475 M	12,185 M
GFLOPs	204.5	207	214.9	228.1
FPS	12.01	11.86	10.14	7.67

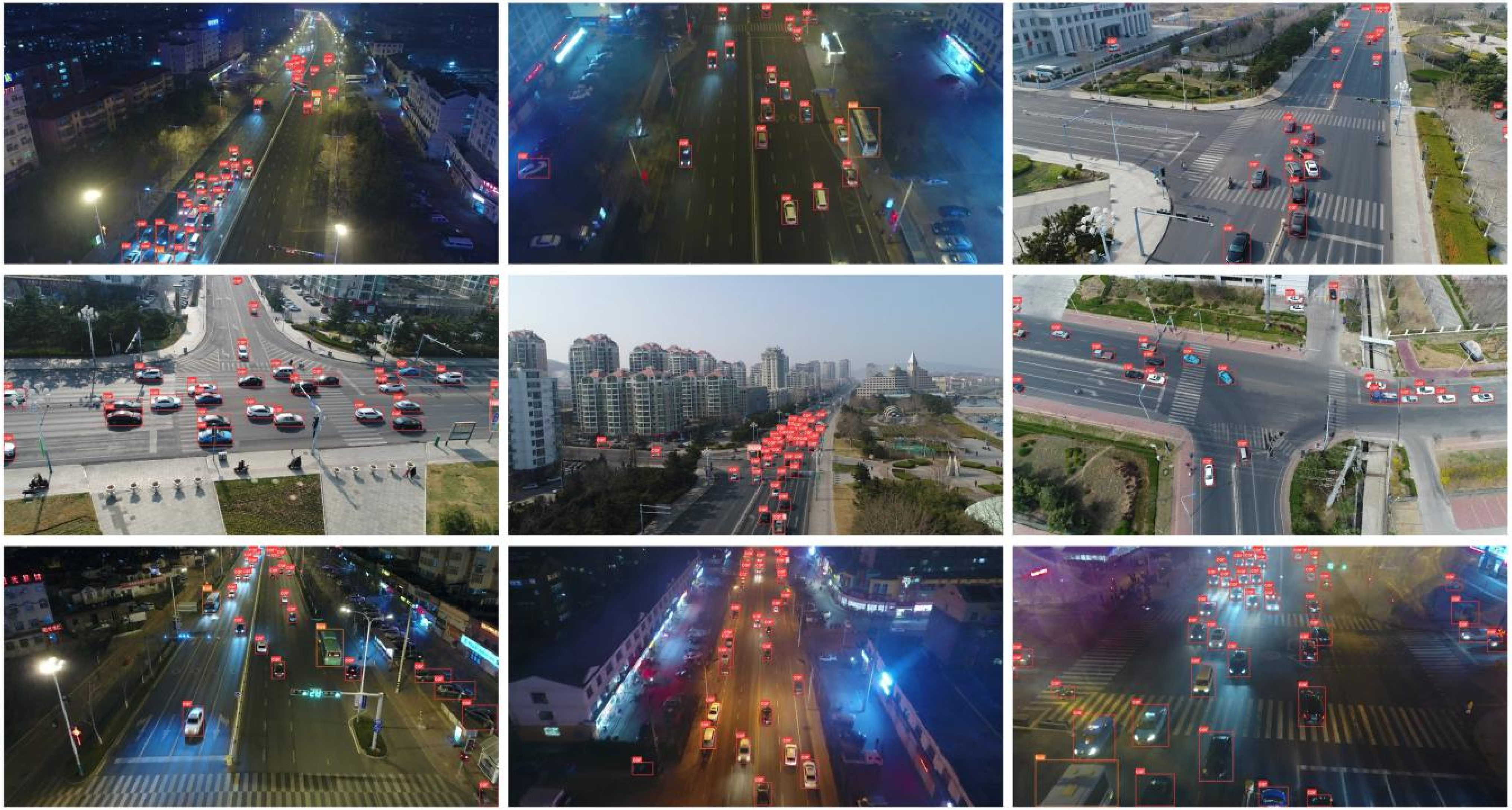
Qualitative Visualization of Detection Results

On the VisDrone2021 test-challenge set

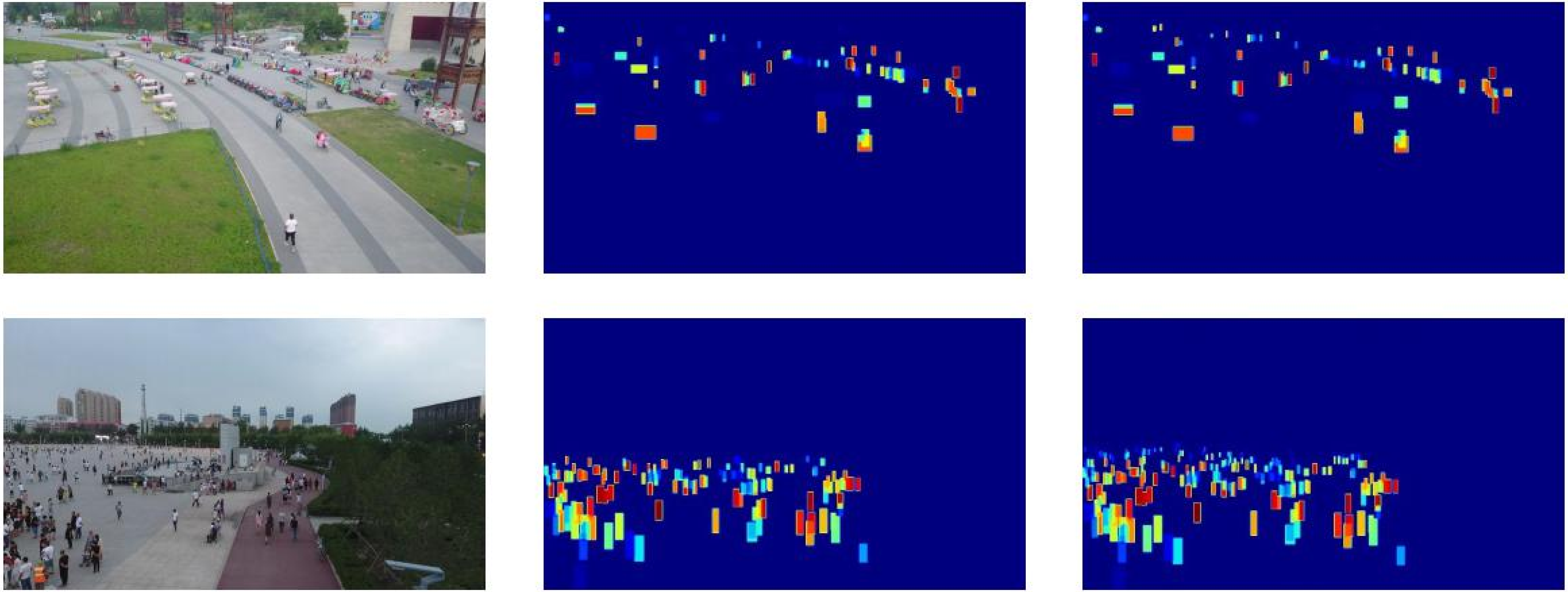


Qualitative Visualization of Detection Results

On the UAVDT dataset



Visualization of Correct Bounding Boxes



(a) Original Image (b) TPH-YOLOv5 (c) TPH-YOLOv5++

Qualitative Visualization of Detection Results

Visualization between TPH-YOLOv5 and TPH-YOLOv5++

(a) TPH-YOLOv5 (b) TPH-YOLOv5++

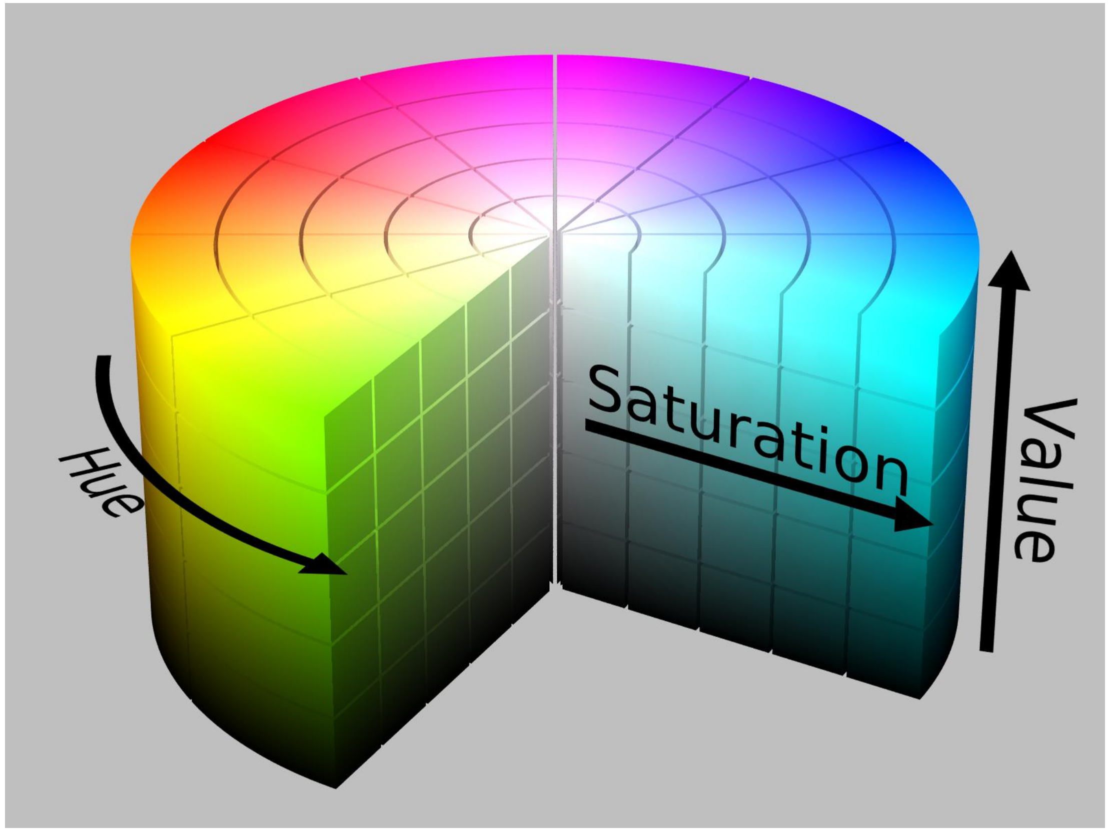
Conclusions

- This paper presents a novel Cross-layer Asymmetric Transformer module
- By replacing the original multi-head self-attention in Vision Transformer with Sparse Local Attention, the Cross-layer Asymmetric Transformer module can enrich the feature of small paths with the help of tiny paths.
- **Opinions:**
  - Explain the YOLOv5 architecture using mathematical formulas
  - Modify the ViT Transformer
  - Apply in my work with Drone datasest

Thank you for your attention!

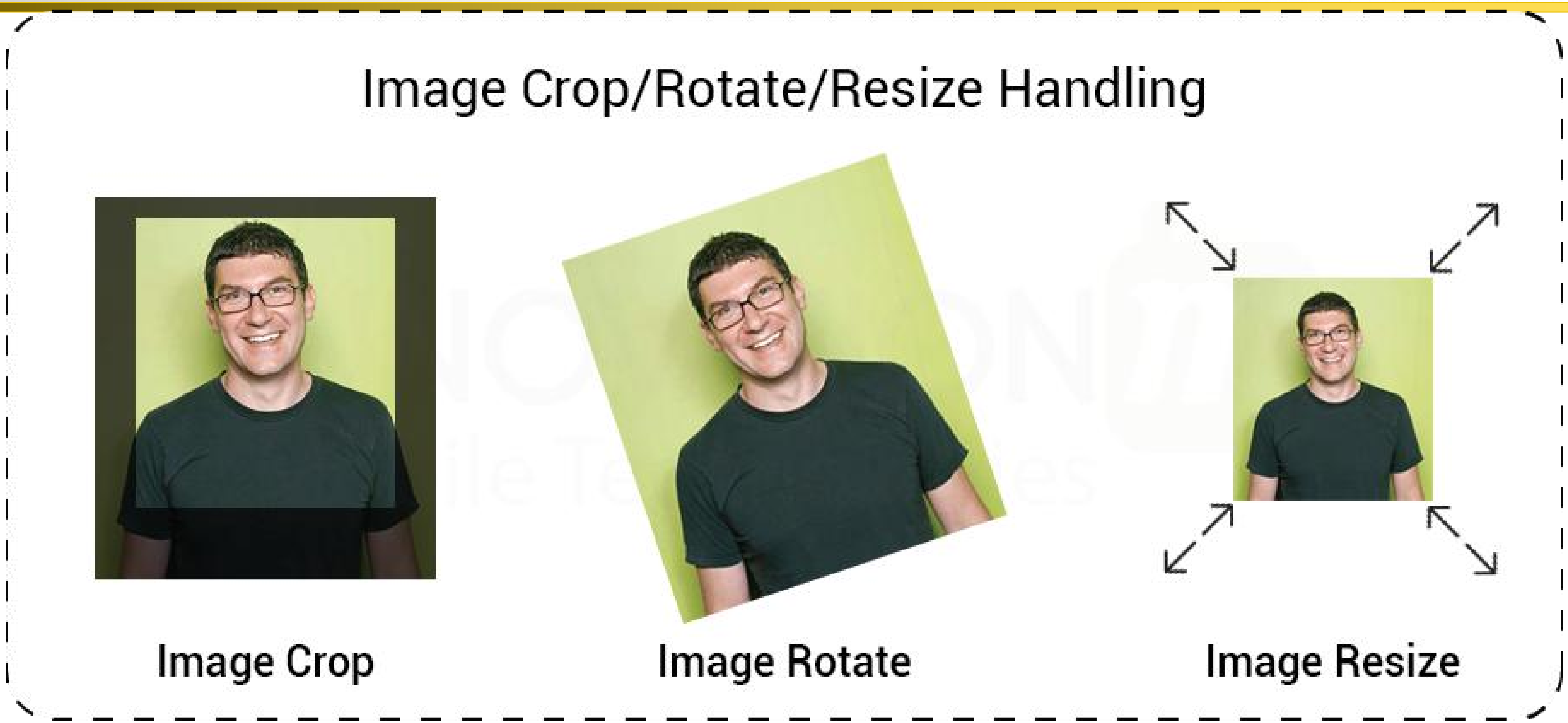
Apendix

Photometric Distortion



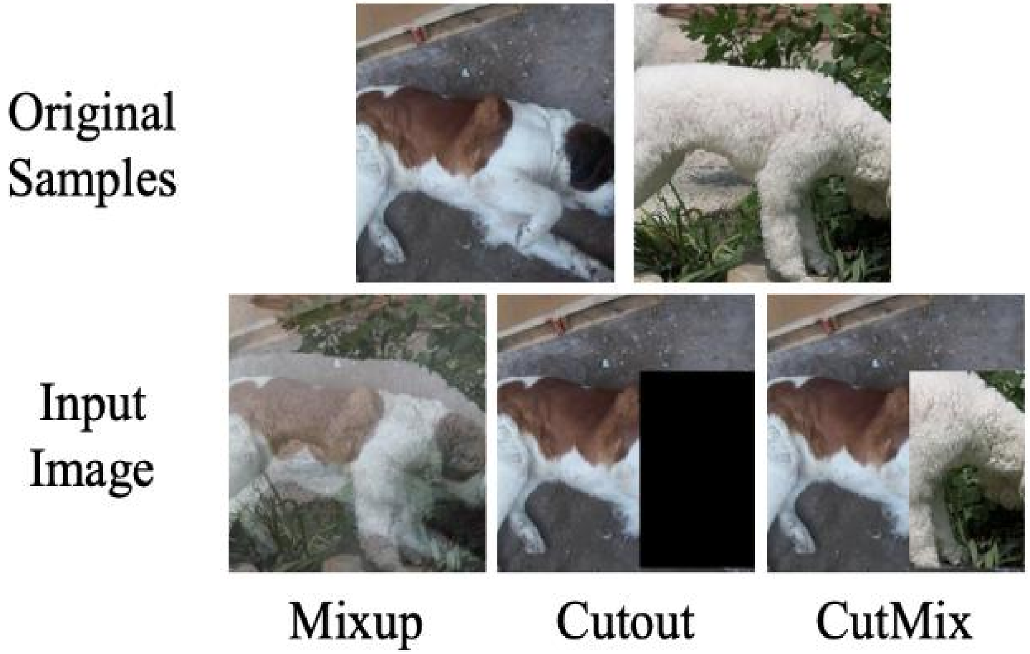
https://giggster.com/guide/basics/hue-saturation-lightness/

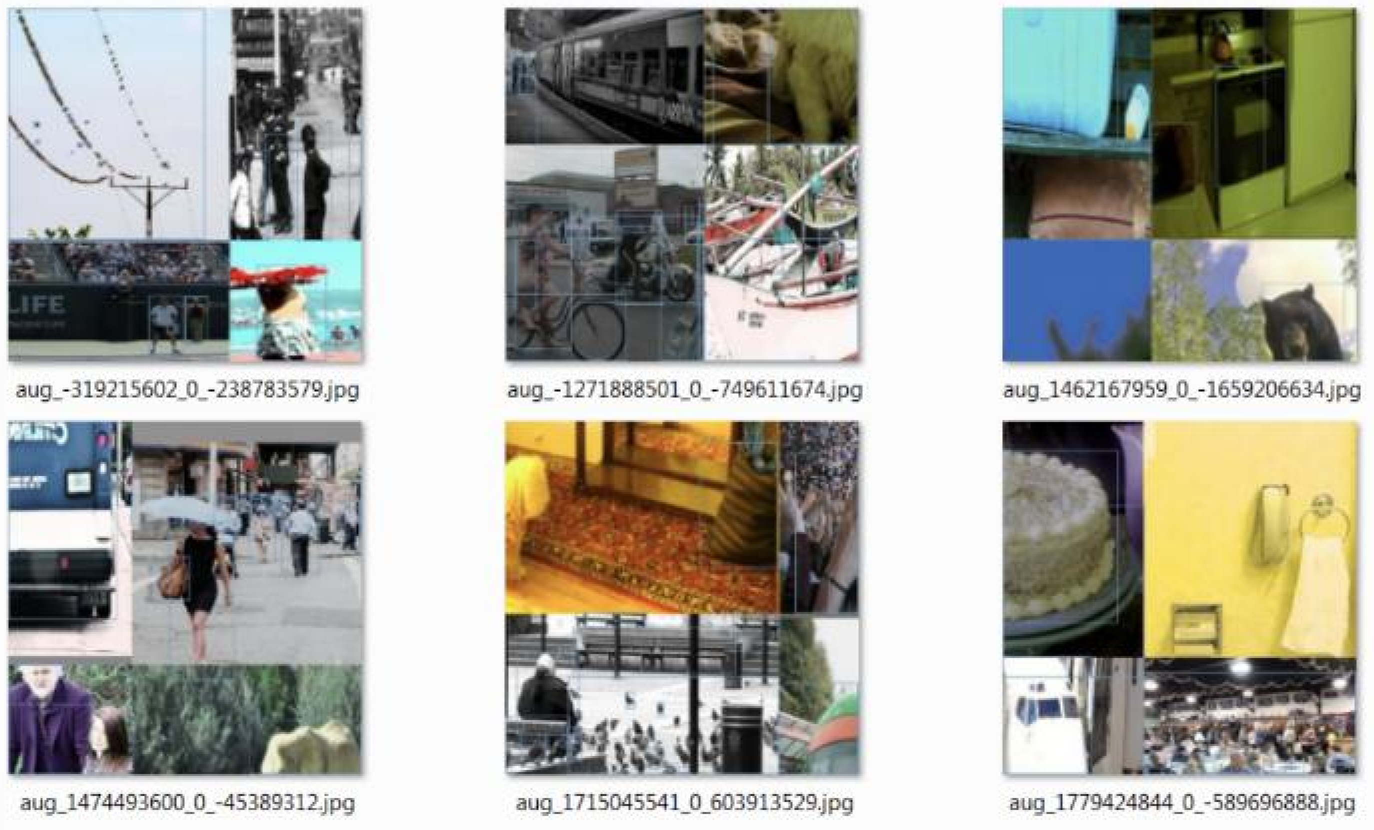
Geometric Distortion



https://innovationm.co/image-croprotateresize-handling-in-ios/

CV Data Augmentation





Mosaic

https://blog.roboflow.com/yolov4-data-augmentation/

Non-Max Suppression (NMS)

The following is the process of selecting the best bounding box using NMS-

Step 1: Select the box with highest objectiveness score

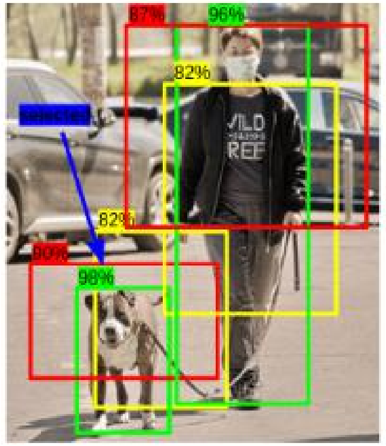
Step 2: Then, compare the overlap (intersection over union) of this box with other boxes

Step 3: Remove the bounding boxes with overlap (intersection over union) >50%

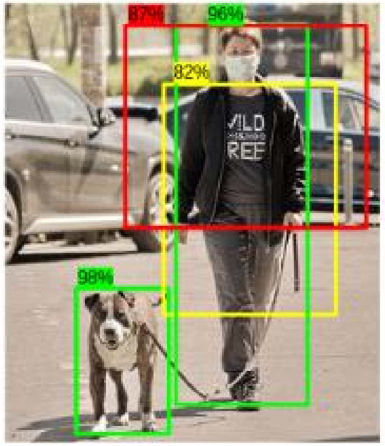
Step 4: Then, move to the next highest objectiveness score

Step 5: Finally, repeat steps 2-4

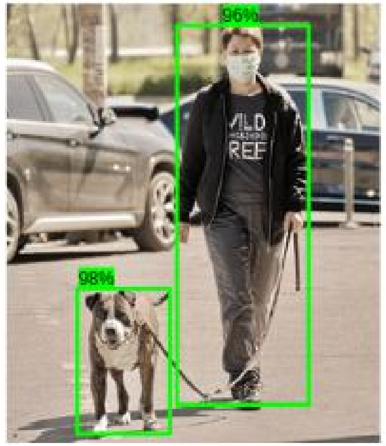
For our example, this loop will run twice. The below images show the output after different steps.



Step 1: Selecting Bounding box with highest score



Step 3: Delete Bounding box with high overlap

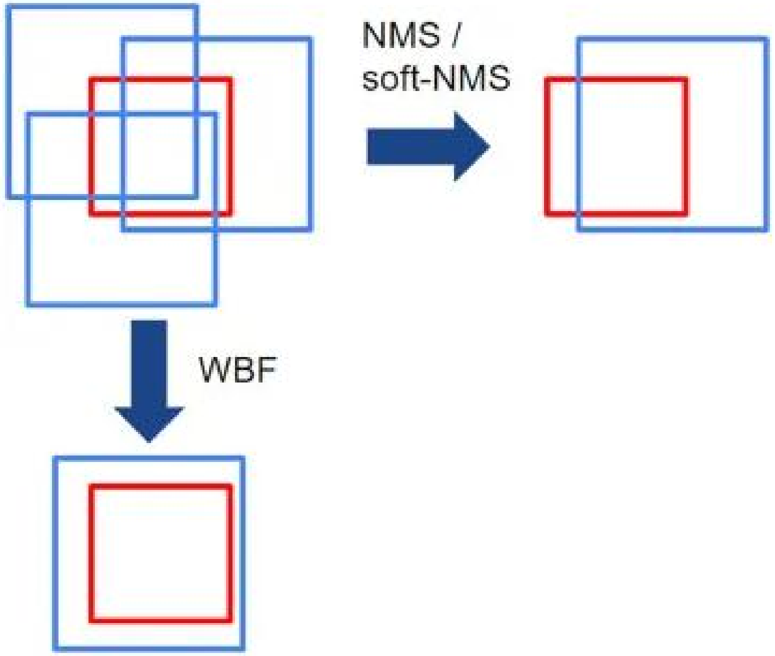


Step 5: Final Output

https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

WBF vs NMS

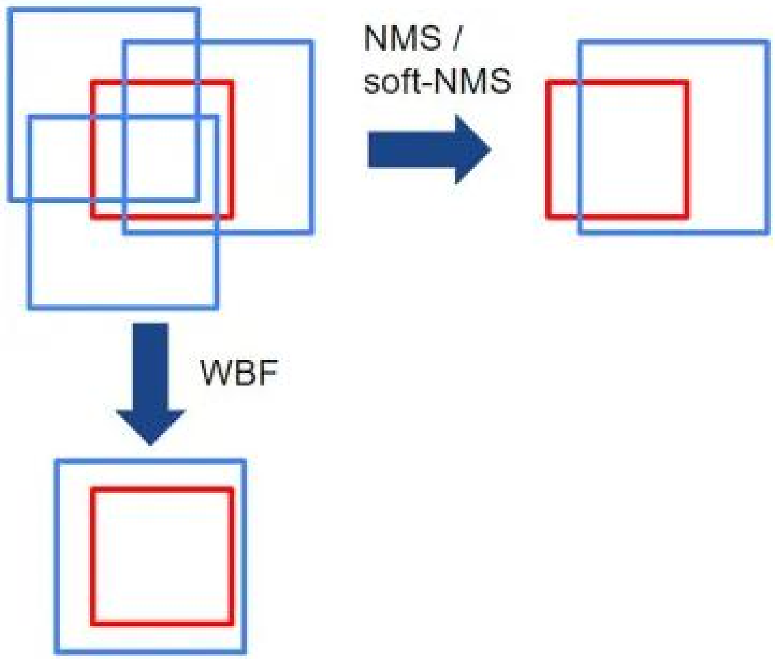
- Both NMS and Soft-NMS filters the boxes by discarding boxes with low confidence scores, but WBF uses information from all the boxes.
- WBF can boost the results where all the ensembled models predict inaccurate boxes, by taking an average of them. See the fig below for a better understanding.



https://blog.roboflow.com/yolov4-data-augmentation/

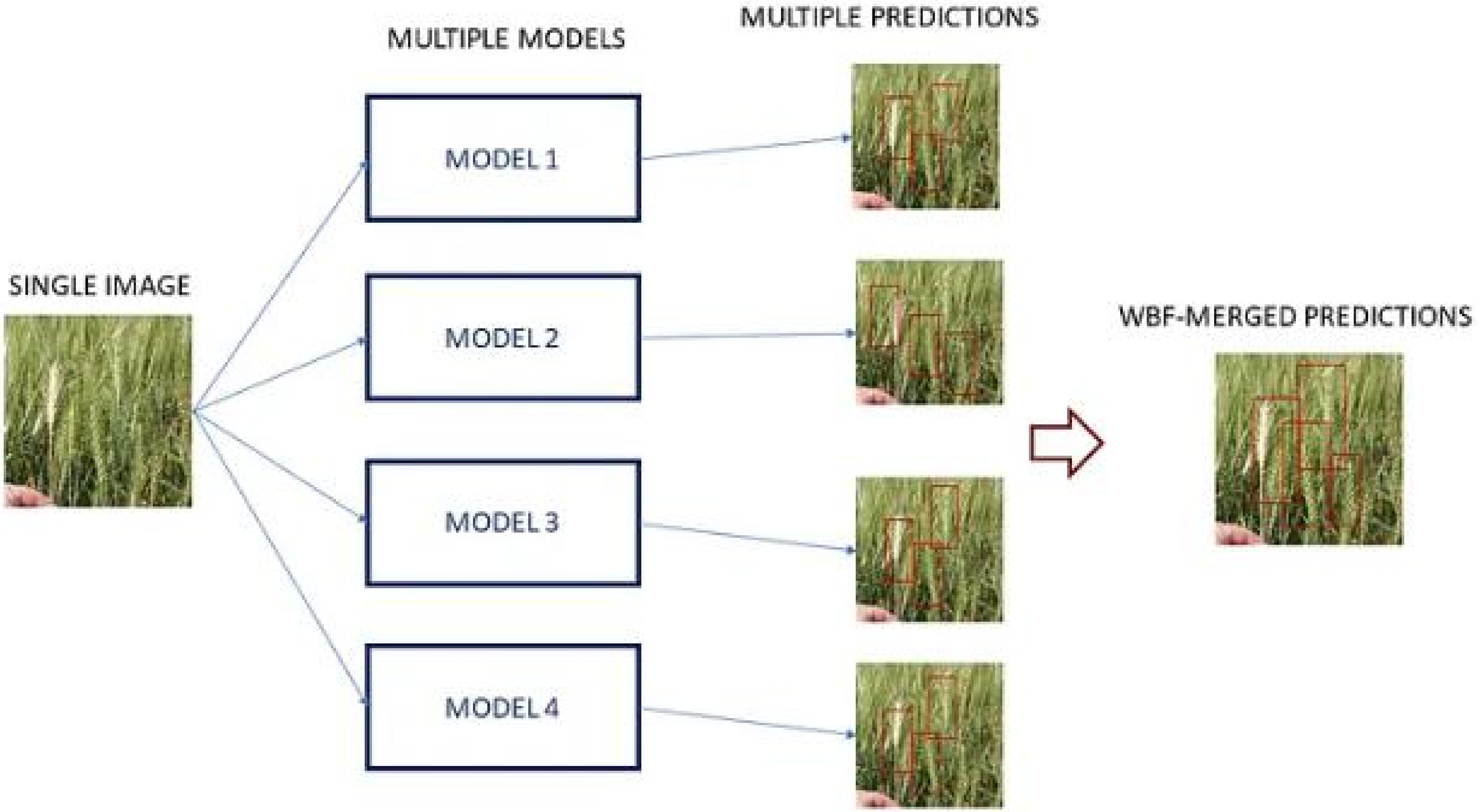
WBF

- Both NMS and Soft-NMS filters the boxes by discarding boxes with low confidence scores, but WBF uses information from all the boxes.
- WBF can boost the results where all the ensembled models predict inaccurate boxes, by taking an average of them. See the fig below for a better understanding.



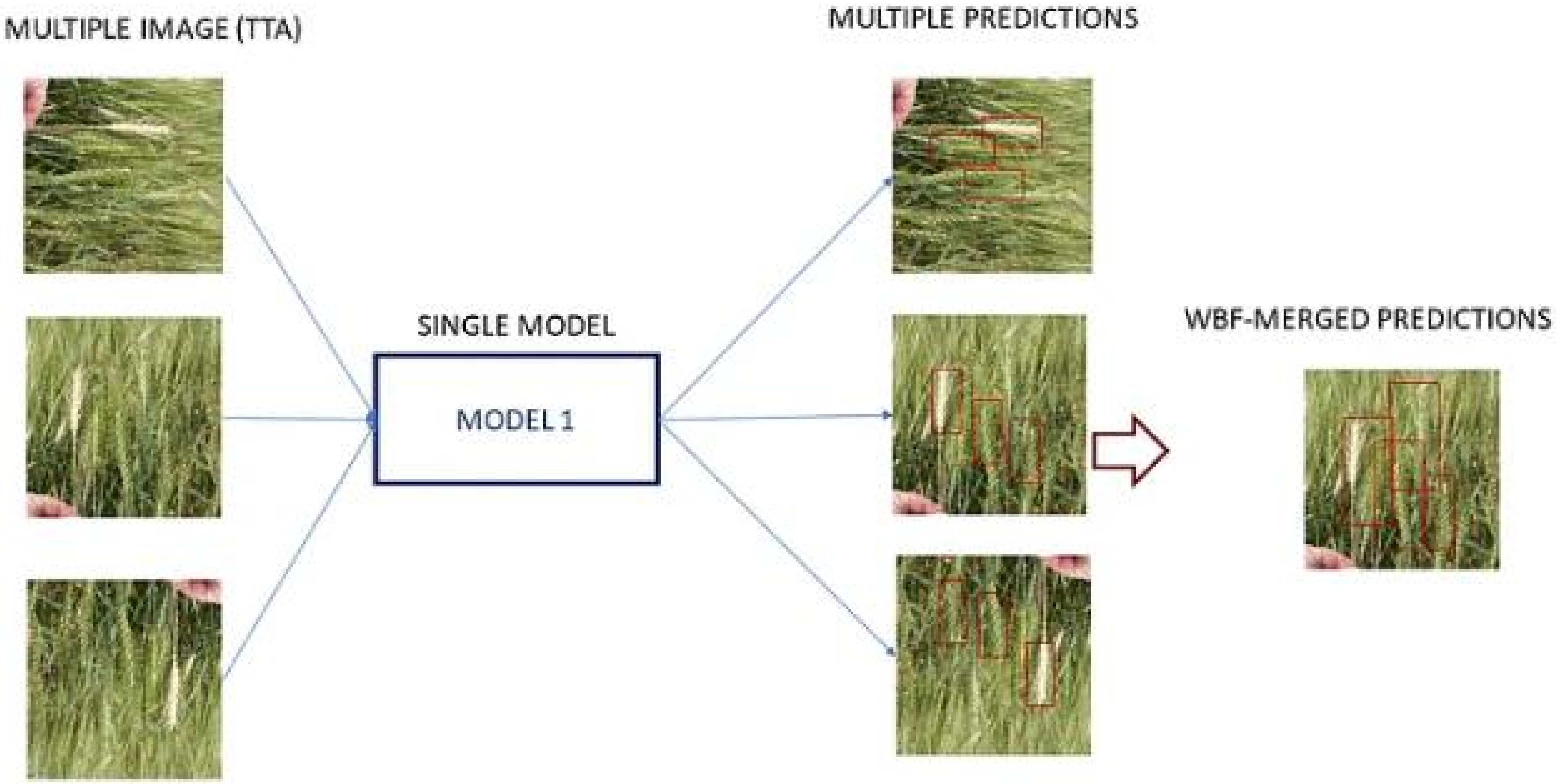
https://blog.roboflow.com/yolov4-data-augmentation/

WBF: Ensemble models —predictions from different models on the same data



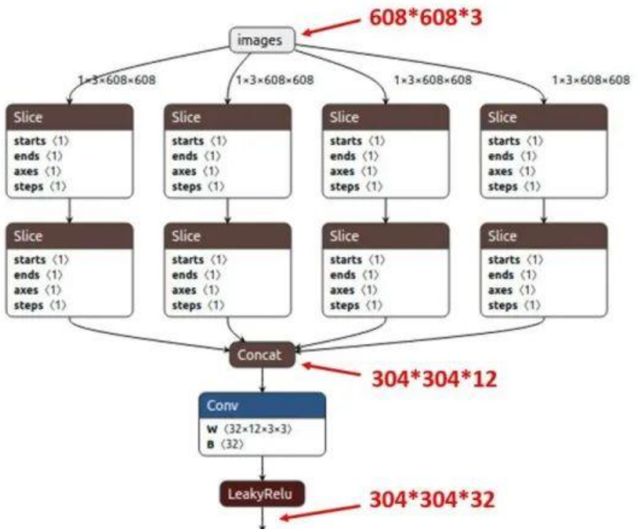
https://medium.com/

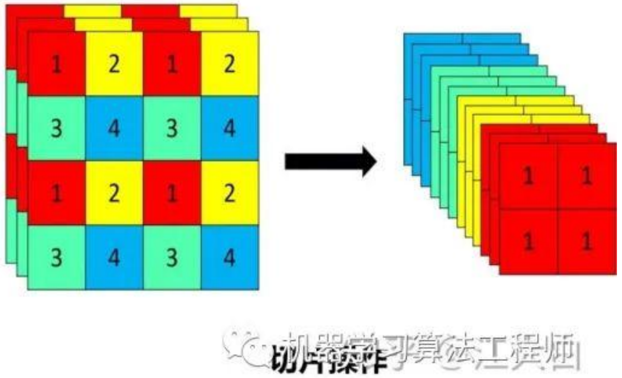
WBF: Predictions from Single model with Augmented data

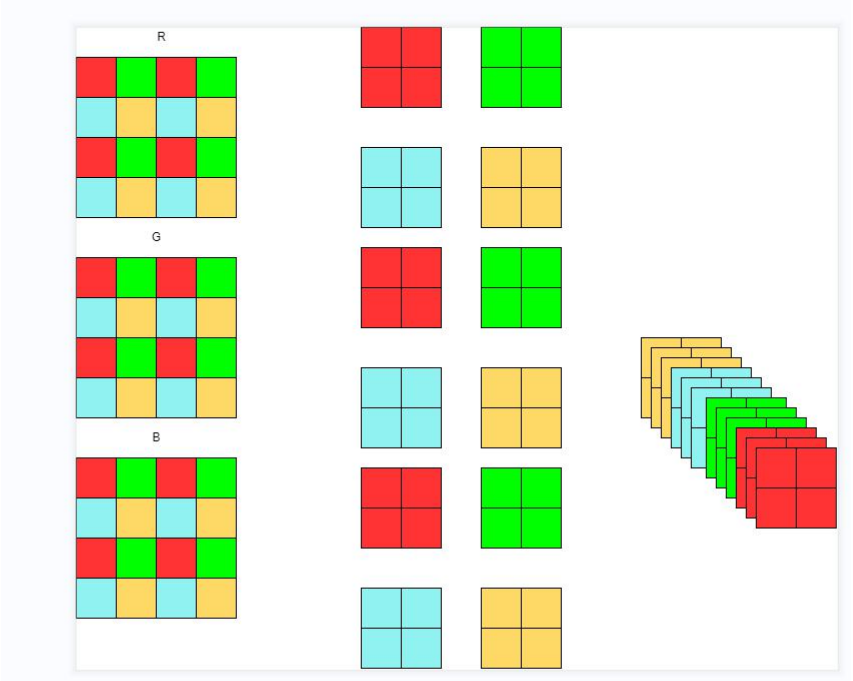


https://medium.com/

Focus module

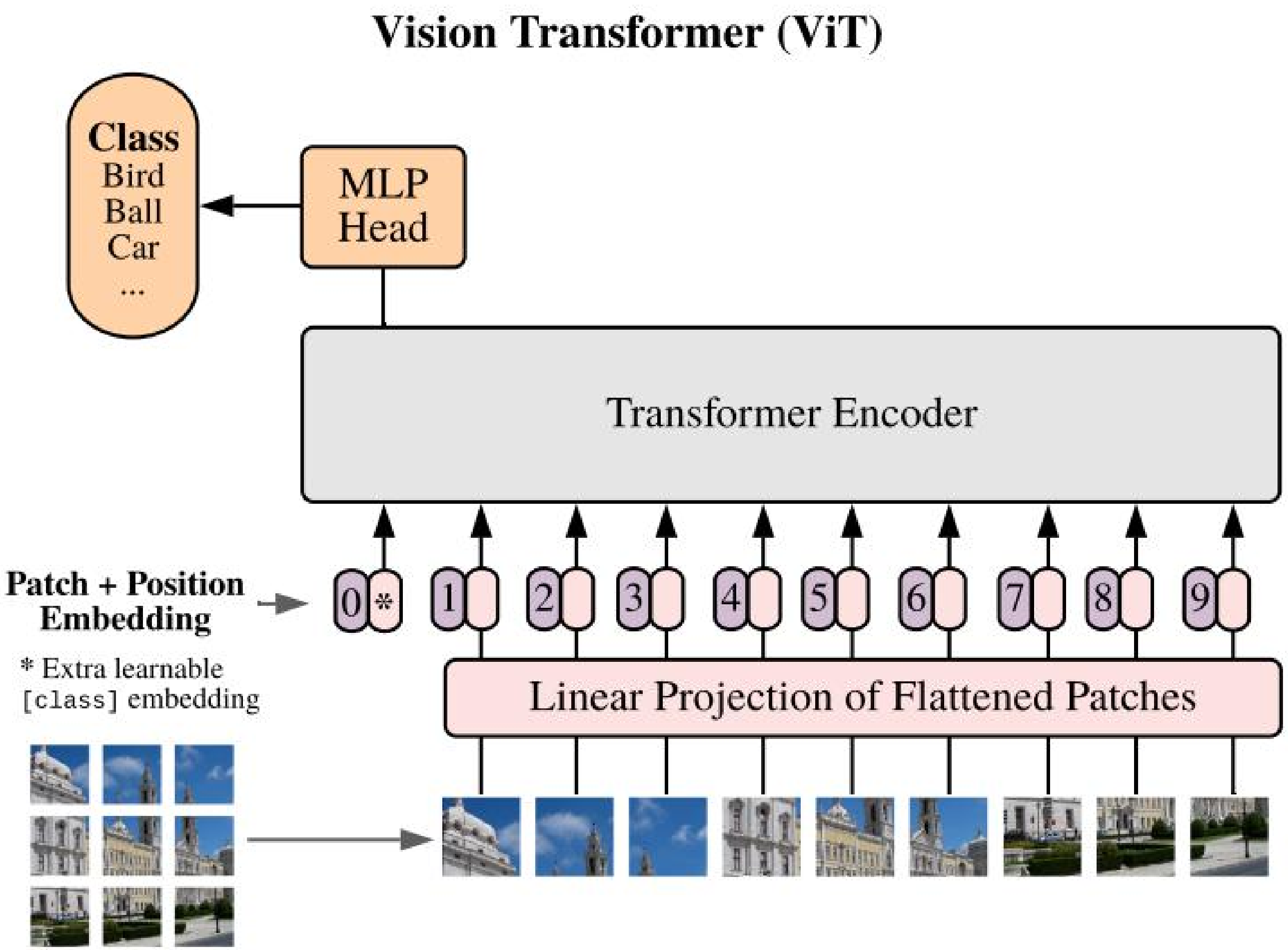




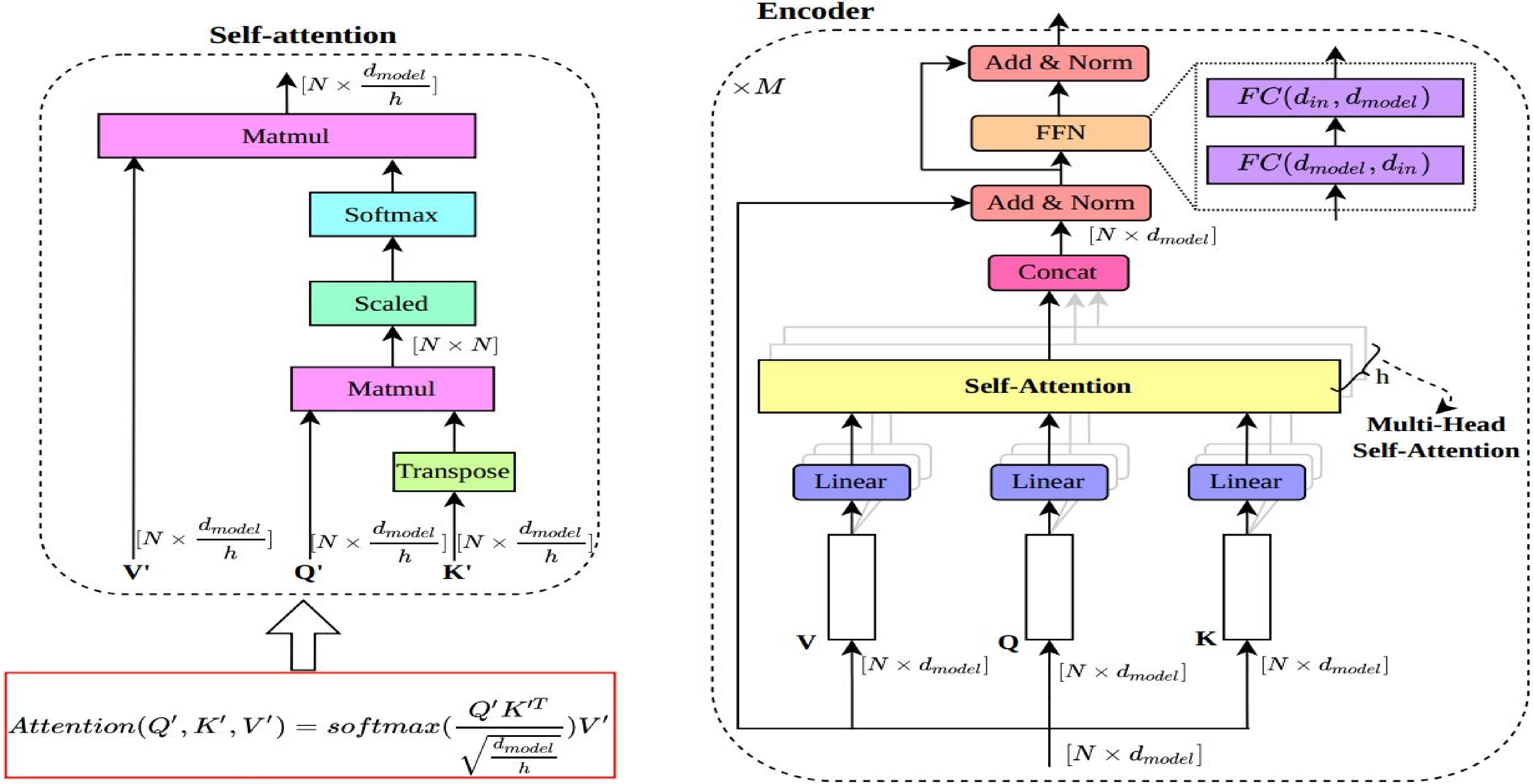


https://programmer.ink/think/analysis-of-yolov5-network-module.html

Vision Transformer (ViT)



Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." ICLR 2021



(a) Self-attention operation

(b) Transformer Encoder

Transformer Encoder

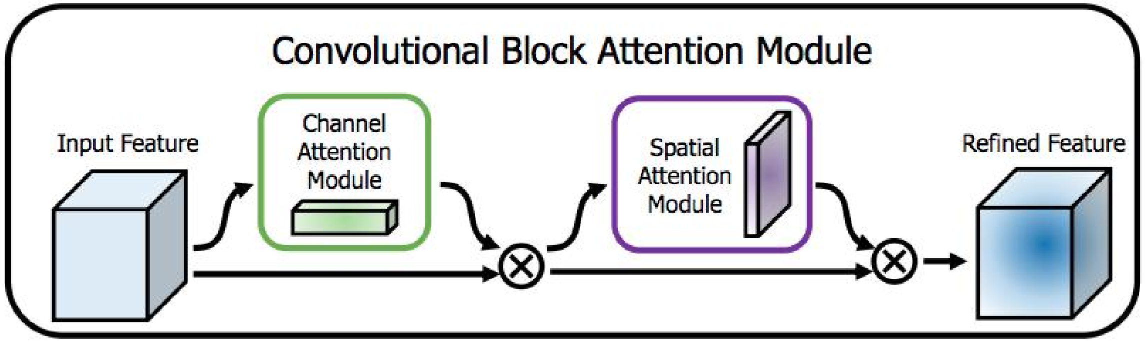
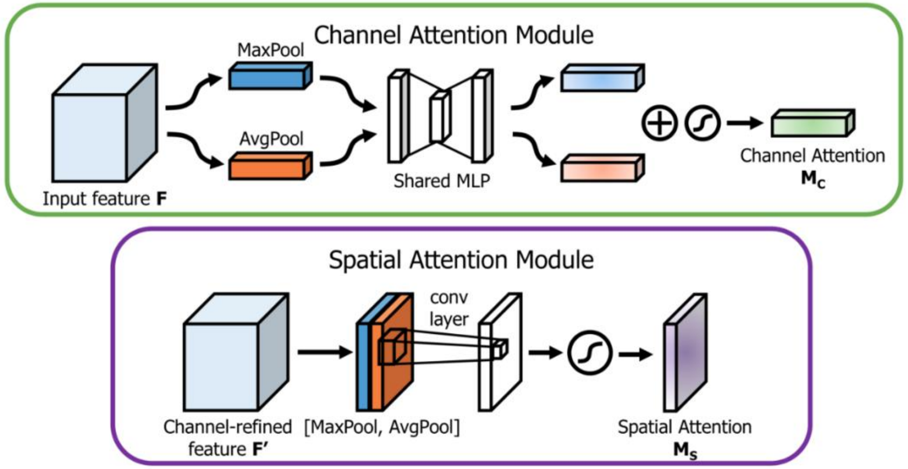


Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.



$$F' = M_c(F) \otimes F$$
$$F'' = M_s(F') \otimes F'$$

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F)))$$
$$M_s(F) = \sigma(f^{7\times7}([AvgPool(F); MaxPool(F)])) = \sigma(f^{7\times7}(F^S_{avg}; F^S_{max}))$$

Cosine lr schedule

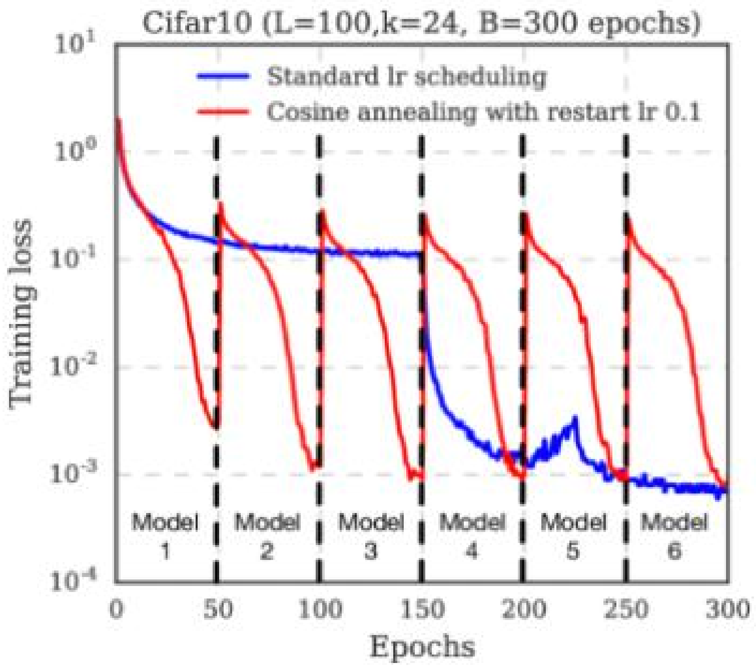
Cosine Annealing is a type of learning rate schedule that has the effect of starting with a large learning rate that is relatively rapidly decreased to a minimum value before being increased rapidly again. The resetting of the learning rate acts like a simulated restart of the learning process and the re-use of good weights as the starting point of the restart is referred to as a "warm restart" in contrast to a "cold restart" where a new set of small random numbers may be used as a starting point.

$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)\right)$$

Where where η^i_{min} and η^i_{max} are ranges for the learning rate, and T_{cur} account for how many epochs have been performed since the last restart.

Text Source: Jason Brownlee

Image Source: Gao Huang

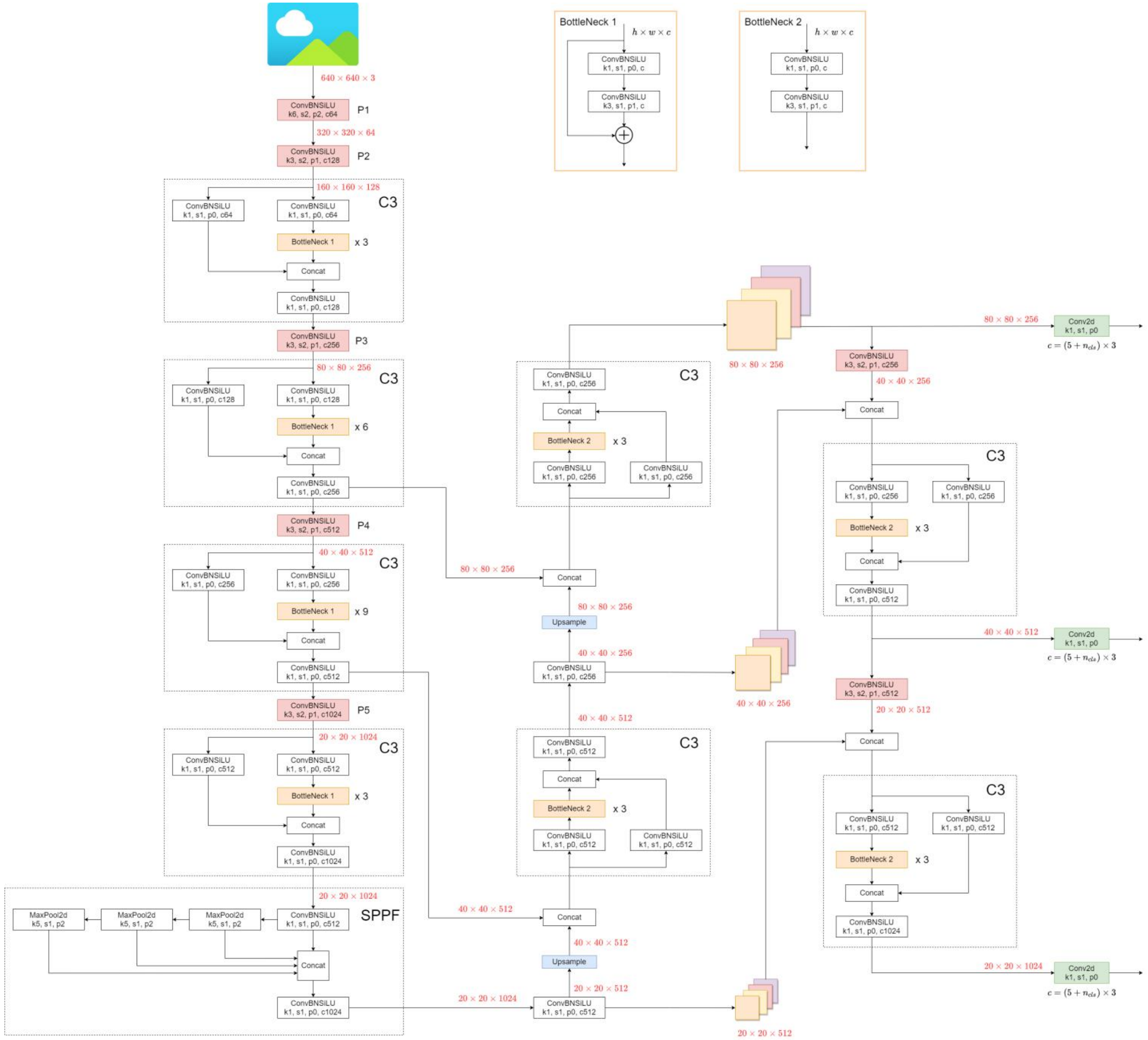


YOLOv5 v6.0

YOLOv5 (v6.0/6.1) consists of:

- **Backbone**: `New CSP-Darknet53`
- **Neck**: `SPPF`, `New CSP-PAN`
- **Head**: `YOLOv3 Head`

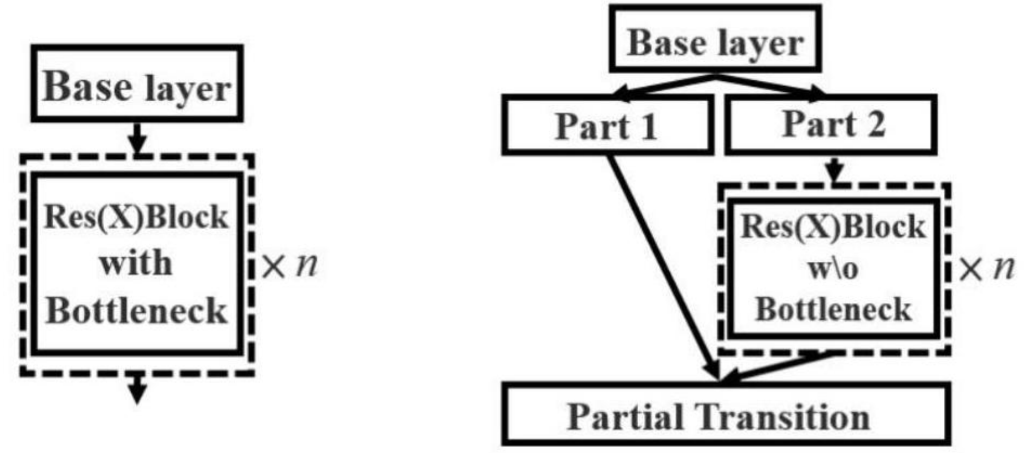
Model structure (`yolov5l.yaml`):



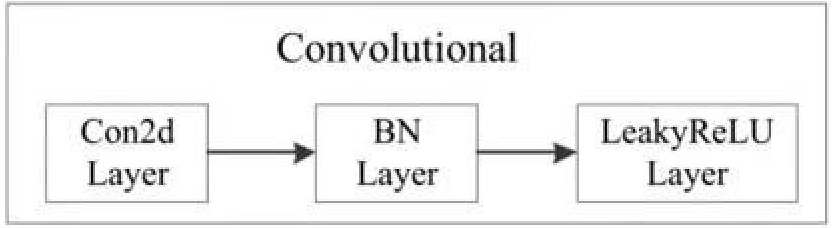
https://github.com/ultralytics/yolov5/issues/6998#41

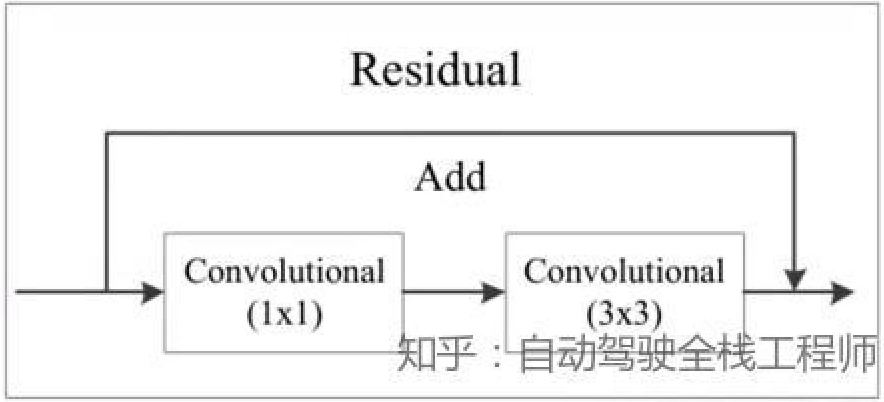
CSP module - Bottleneck

CSP （Cross Stage Partial Network） 跨阶段局部网络

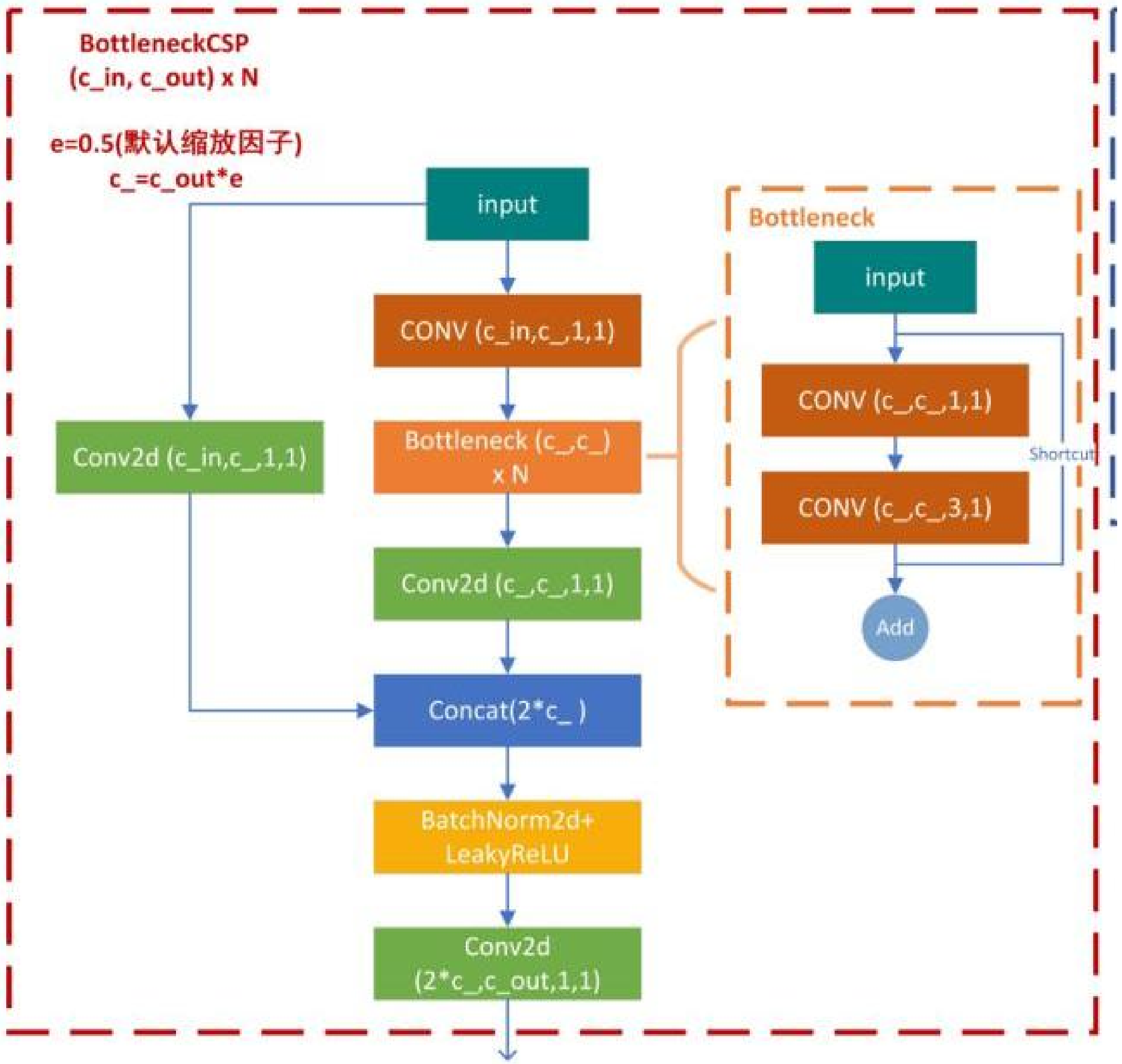


(a) ResNe(X)t (b) CSPResNe(X)t

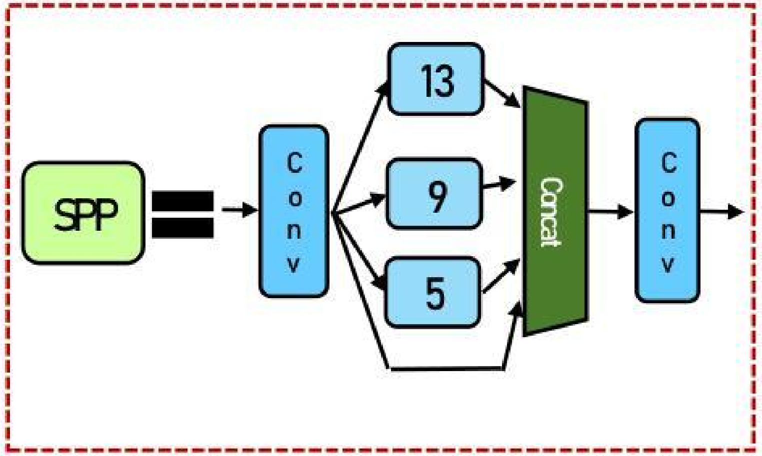




CSP module - Bottleneck

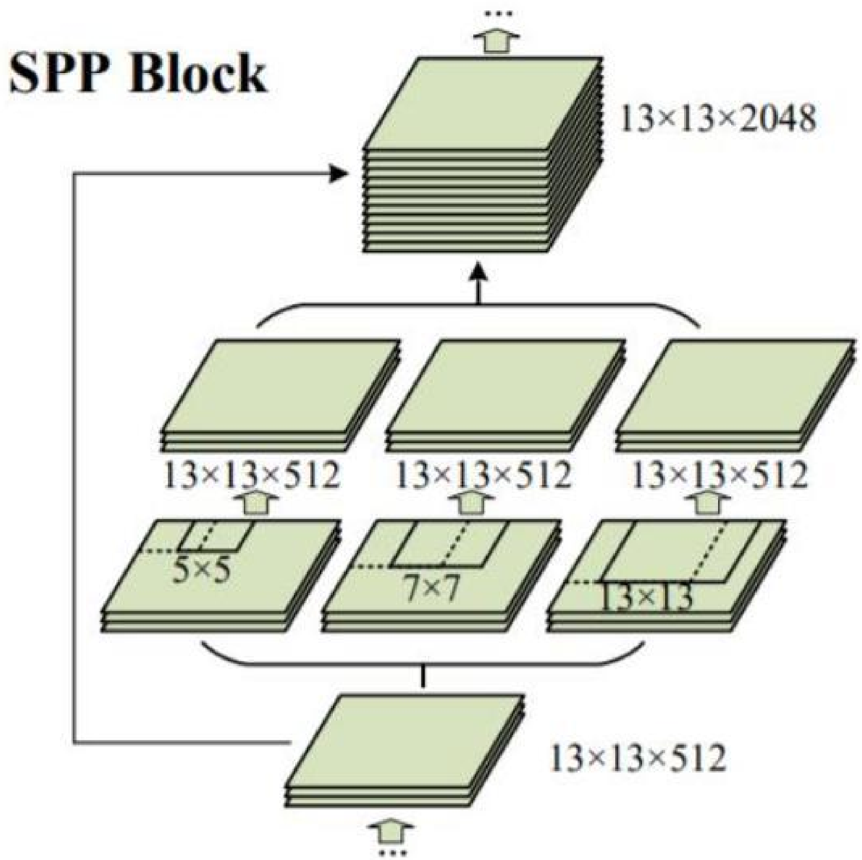


SPP – Spatial Pyramid Pooling



Add the SPP block over the CSP, since it significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed.

416×416输入



Neck – Path Aggregation Network

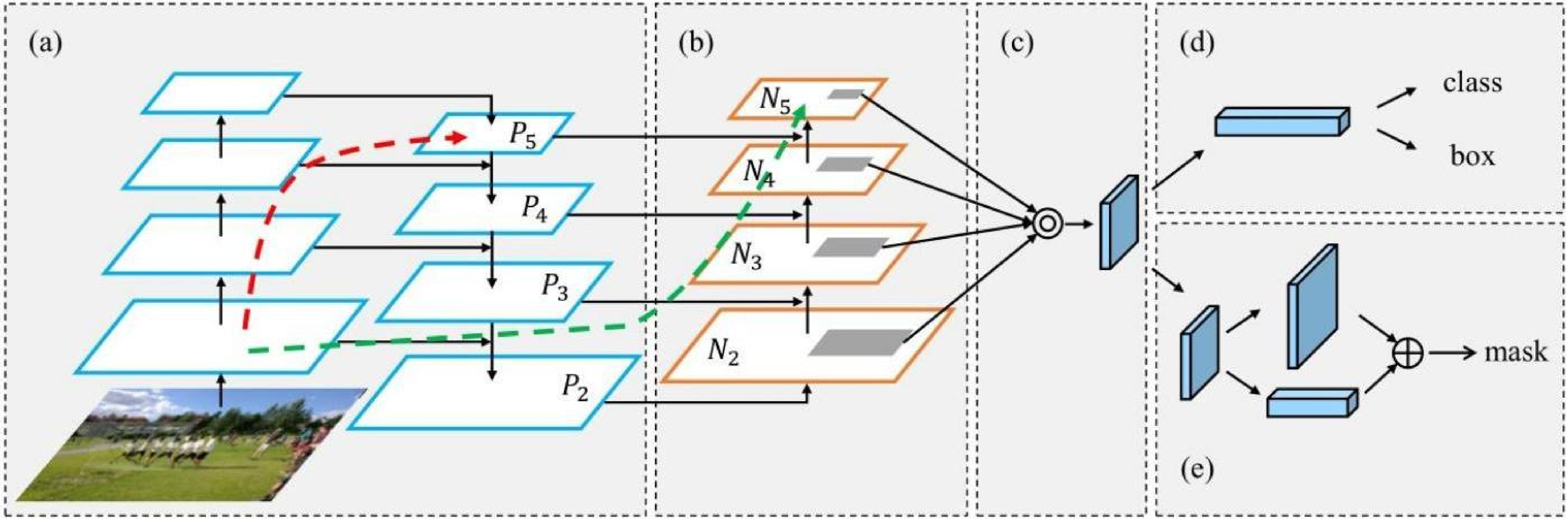
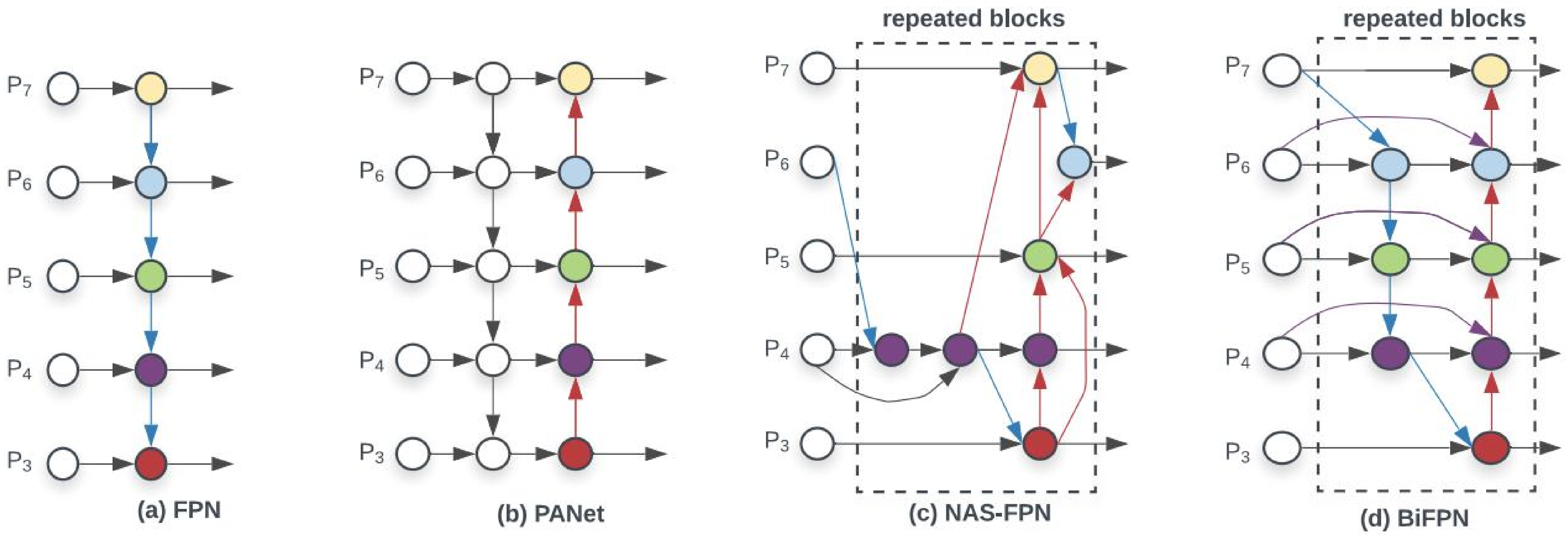
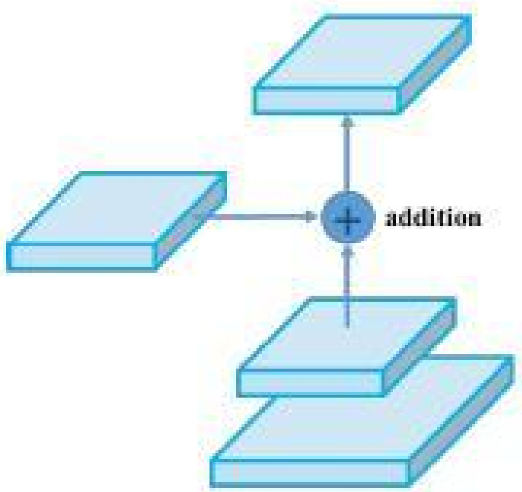


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

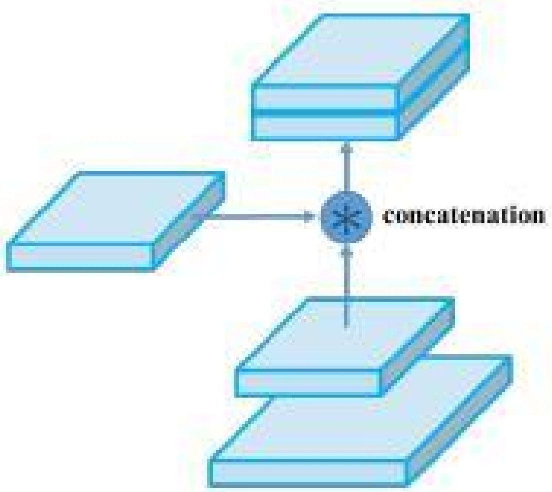
Neck – Path Aggregation Network



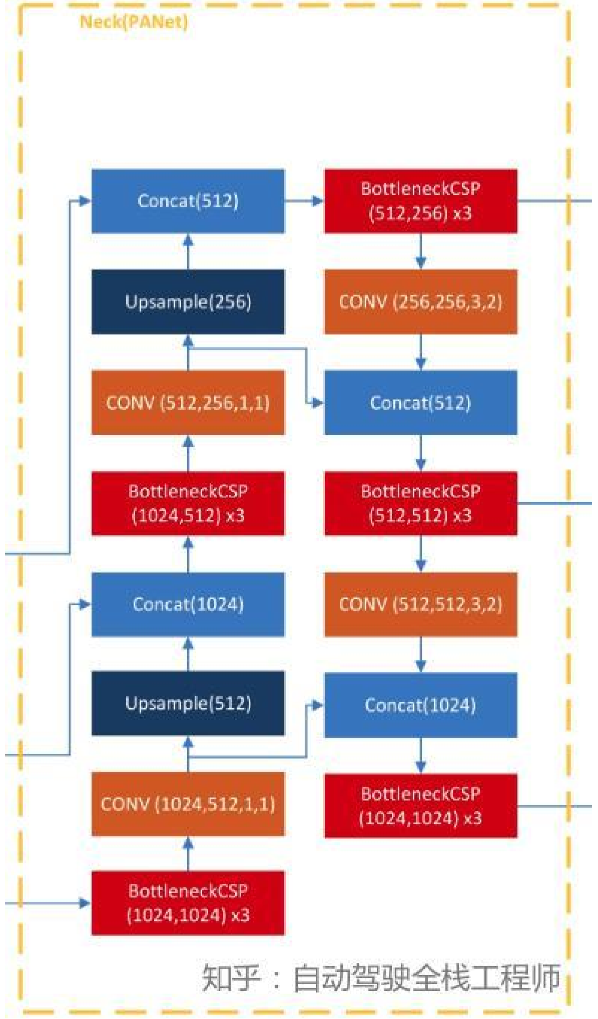
Neck – Path Aggregation Network



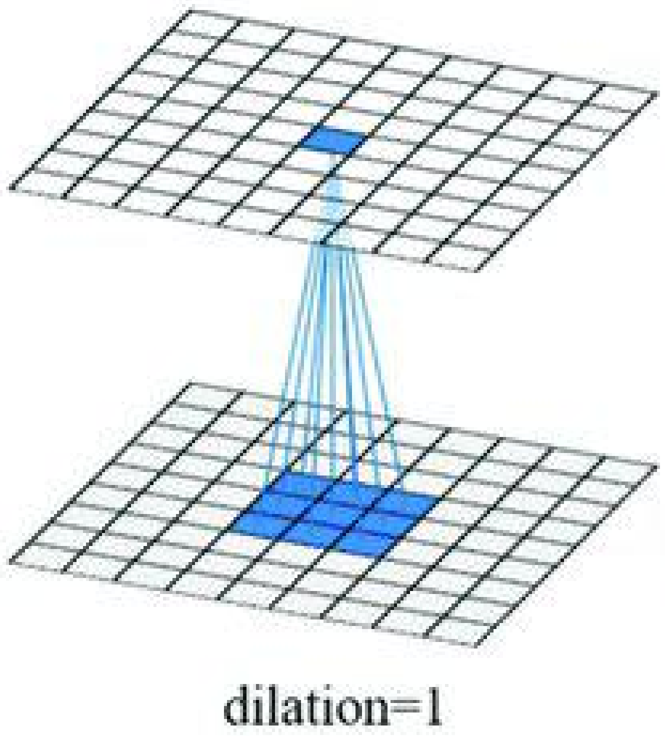
(a) PAN [49]



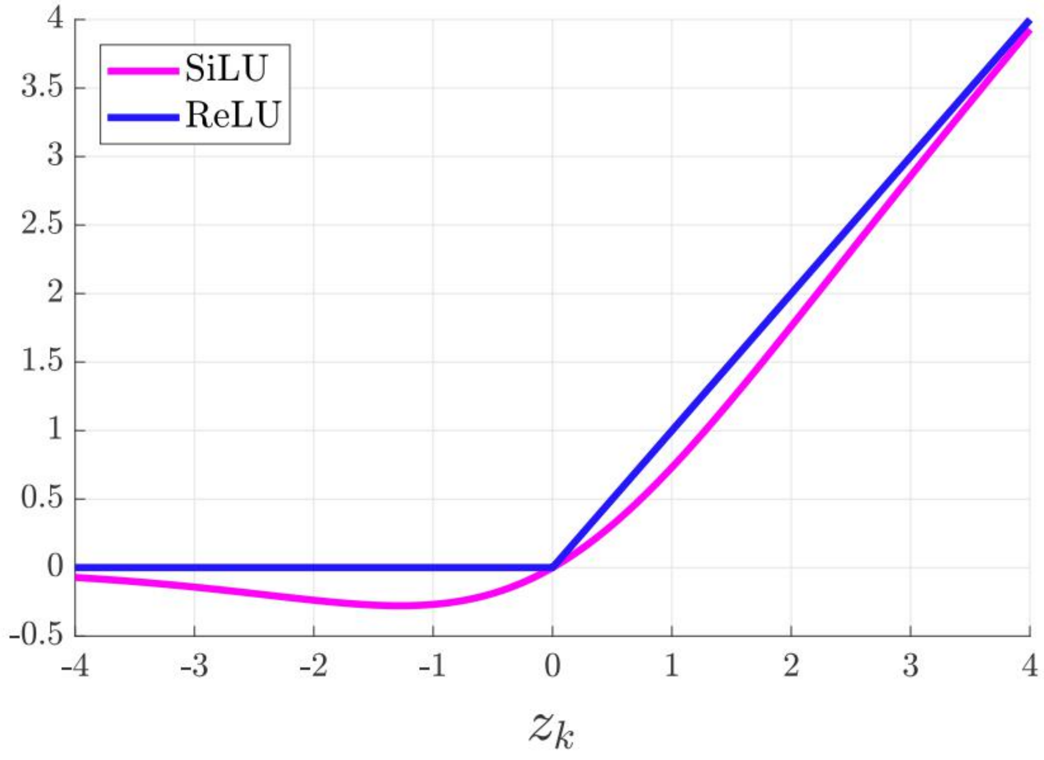
(a) Our modified PAN



Dilated convolution

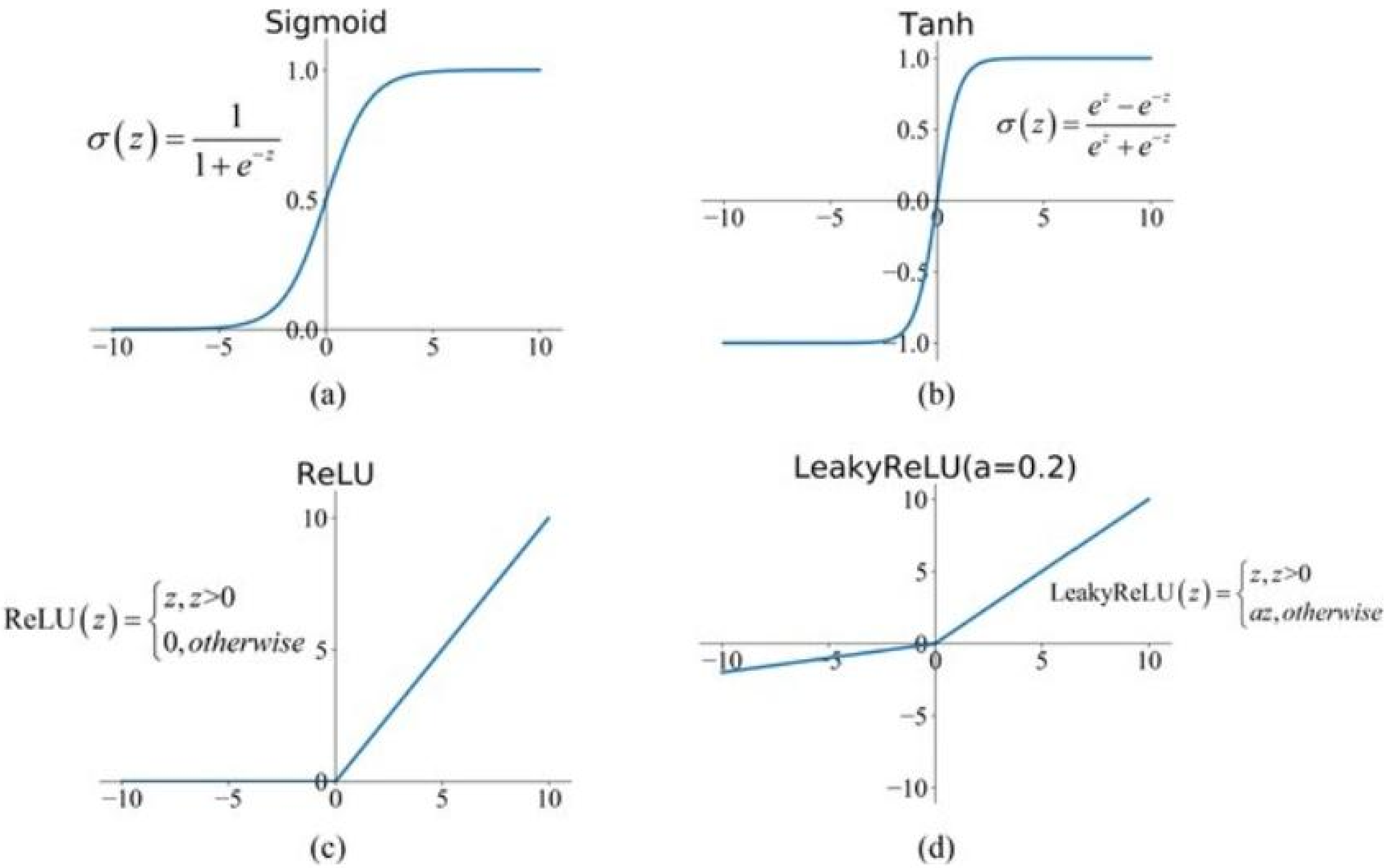


SiLU Activation



$$SiLU(x) = x * \sigma(x) = x * \frac{1}{1 + e^{-x}}$$

Activation function



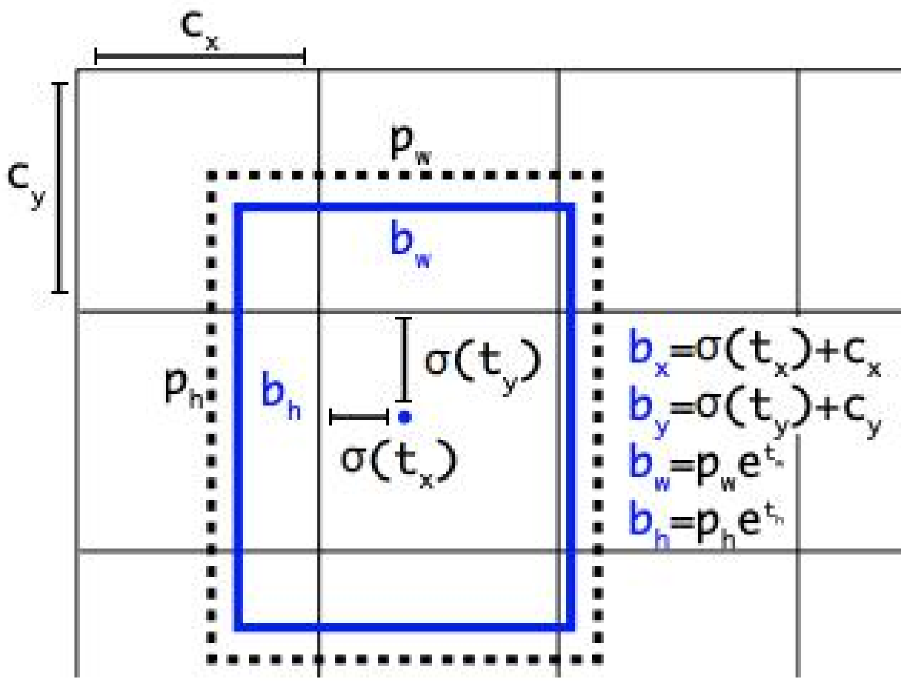
Parameters of the four structures

	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
depth_multiple	0.33	0.67	1.0	1.33
width_multiple	0.50	0.75	1.0	1.25
BottleneckCSP数 BCSPn(True)	1,3,3	2,6,6	3,9,9	4,12,12
BottleneckCSP数 BCSPn(Flase)	1	2	3	4
Conv卷积核数量	32,64,128,256,512	48,96,192,384,768	64,128,256,512,1024	80,160,320,640,1280

Anchor

- There will be Anchor frame with initial length and width
- the anchor frame initially set by Yolov5 on the Coco data set:

```
anchors:
  - [116,90, 156,198, 373,326]  # P5/32
  - [30,61, 62,45, 59,119]  # P4/16
  - [10,13, 16,30, 33,23]  # P3/8
```

Total Loss

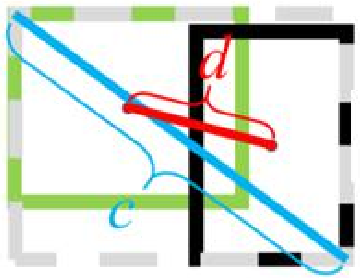
The YOLOv5 loss consists of three parts:

- Classes loss(BCE loss)
- Objectness loss(BCE loss)
- Location loss(CIoU loss)

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

Total Loss

$$Total\ Loss = \lambda_{box}L_{box} + \lambda_{obj}L_{obj} + \lambda_{cls}L_{cls}$$



$\lambda_{box} = balancing\ parameter\ loss\ regre$
$\lambda_{obj} = balancing\ parameter\ loss\ object$
$\lambda_{cls} = balancing\ parameter\ loss\ classi$

$$L_{box} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} L_{CIoU}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b,\ b^{gt})}{c^2} + \frac{v^2}{(1 - IoU) + v}$$

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

$\rho = distance\ of\ center\ point$

$c = distance\ of\ border\ points\ (top - left,\ right -$

$w = width\ prediction\ box$

$h = height\ prediction\ box$

$w^{gt} = width\ ground\ truth\ box$

$h^{gt} = height\ ground\ truth\ box$

$s^2 = grid\ size$

$B = number\ of\ anchors$

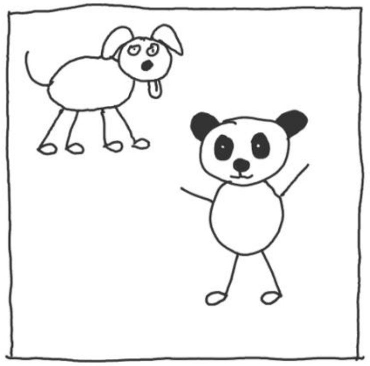
$i = grid$

$j = anchor$

*$\mathbb{1}$ **obj** is equal to one when there is an object in the cell, and 0 otherwise.*

Total Loss

$$\text{Binary Cross-entropy} = -\left(\underbrace{p(x)\cdot\log q(x)}_{\substack{\text{This cancels out}\\ \text{if the target is 0}}} + \underbrace{(1-p(x))\cdot\log(1-q(x))}_{\substack{\text{This cancels out}\\ \text{if the target is 1}}}\right)$$

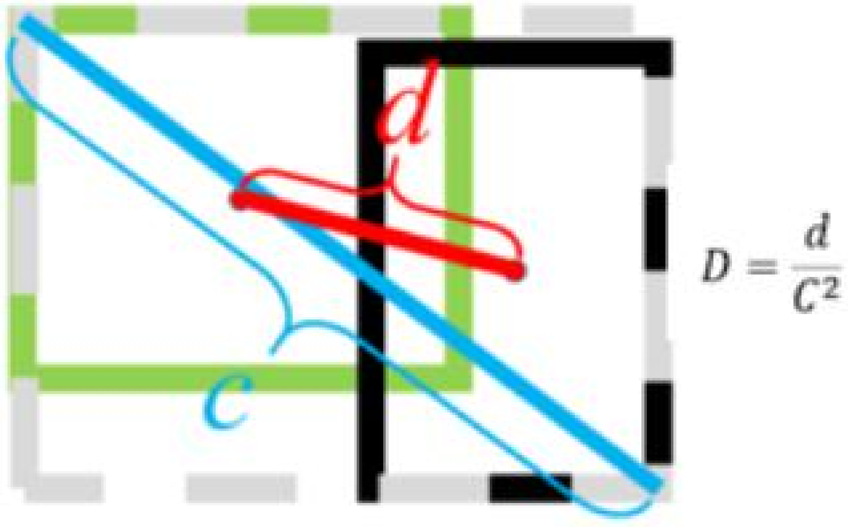


$$\begin{array}{cc} \text{TARGET} & \text{PREDICTION} \\ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0.6 \\ 0.7 \\ 0.4 \end{bmatrix} \end{array}$$

https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451

Distance IoU Loss

$$DIoU = 1 - \text{IoU} + \frac{d}{c^2}$$



DIoU loss is invariant to the scale of regression problem, and like GIoU loss, DIoU loss also provides the moving directions for predicted bounding boxes for non-overlapping cases.

Efficient sub-pixel convolutional neural network (ESPCN)

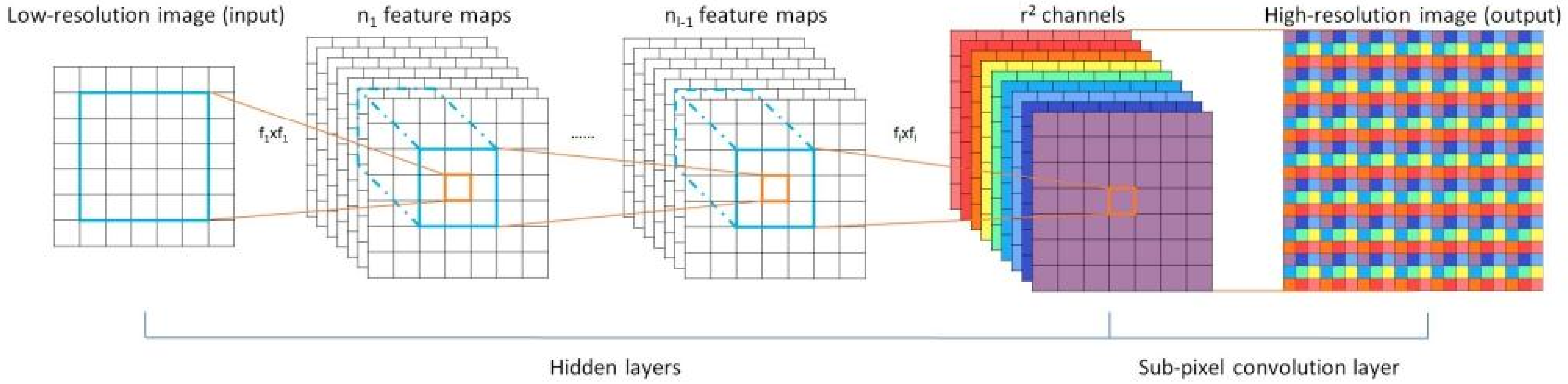
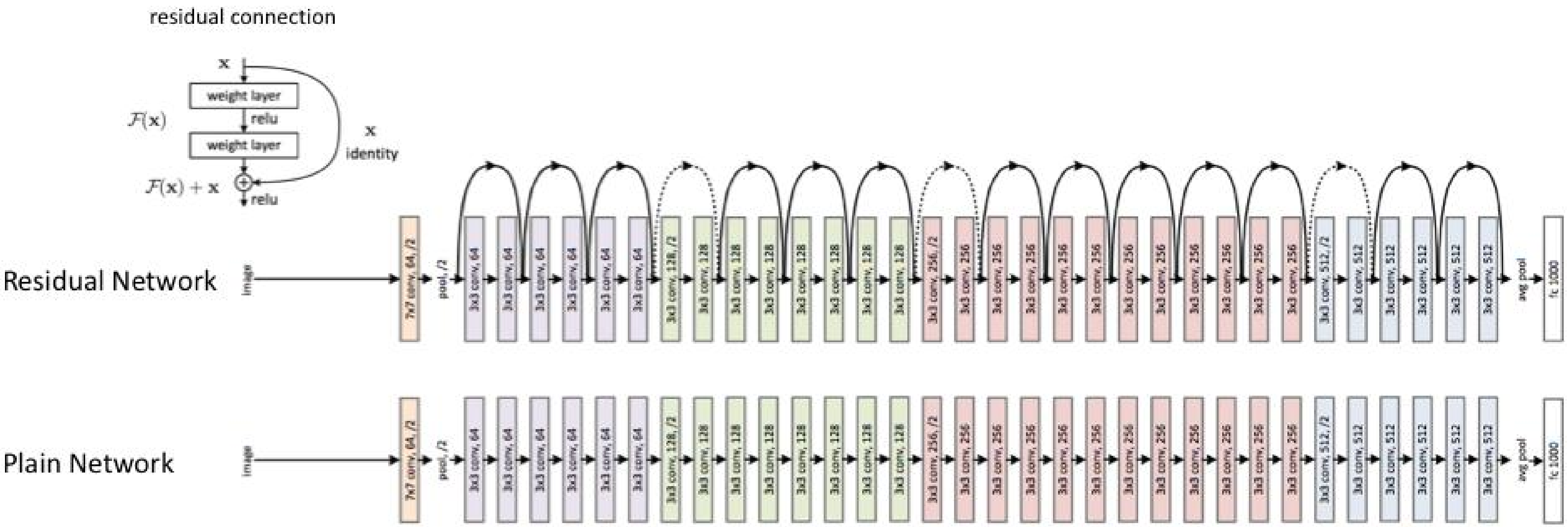


Figure 1. The proposed efficient sub-pixel convolutional neural network (ESPCN), with two convolution layers for feature maps extraction, and a sub-pixel convolution layer that aggregates the feature maps from LR space and builds the SR image in a single step.

ResNet



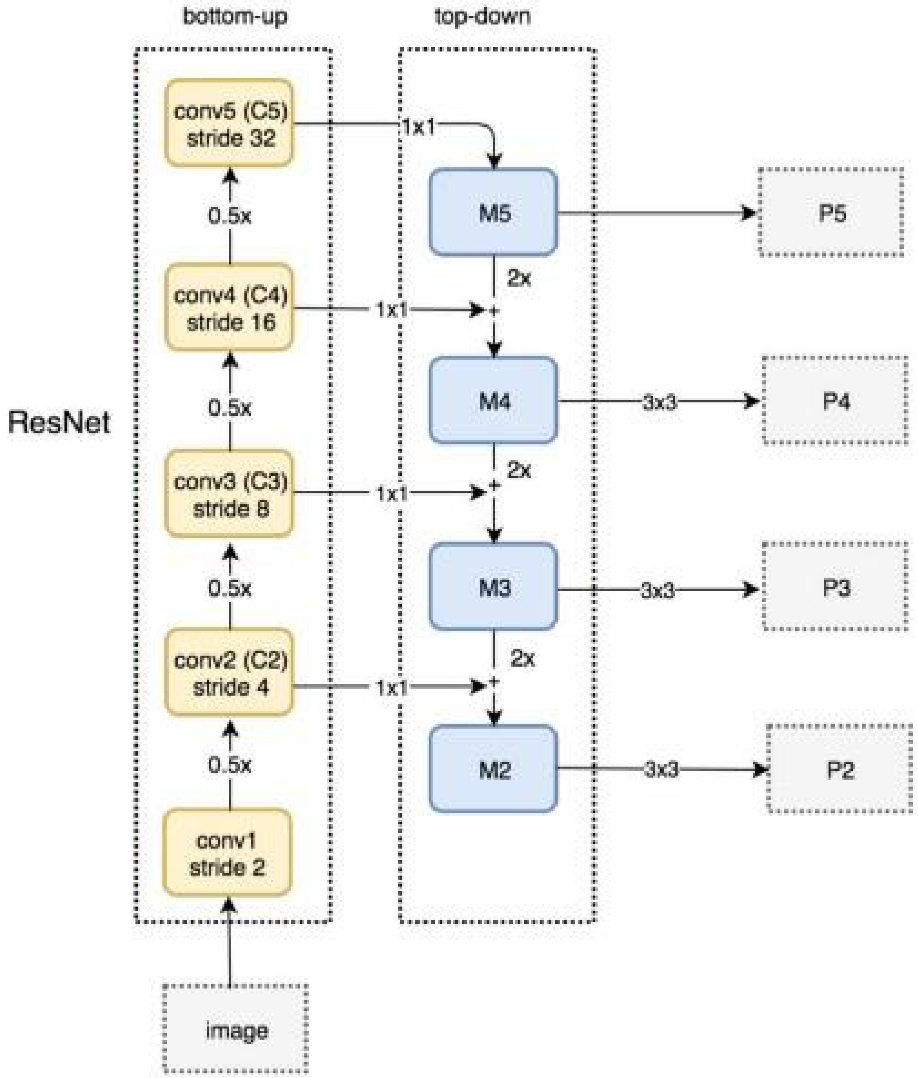
residual connection
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
Residual Network
Plain Network
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

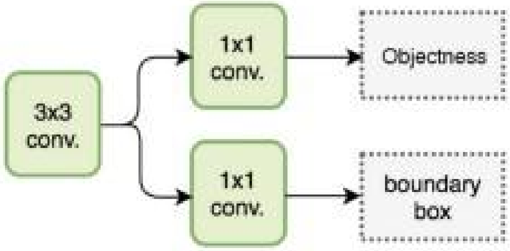
ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

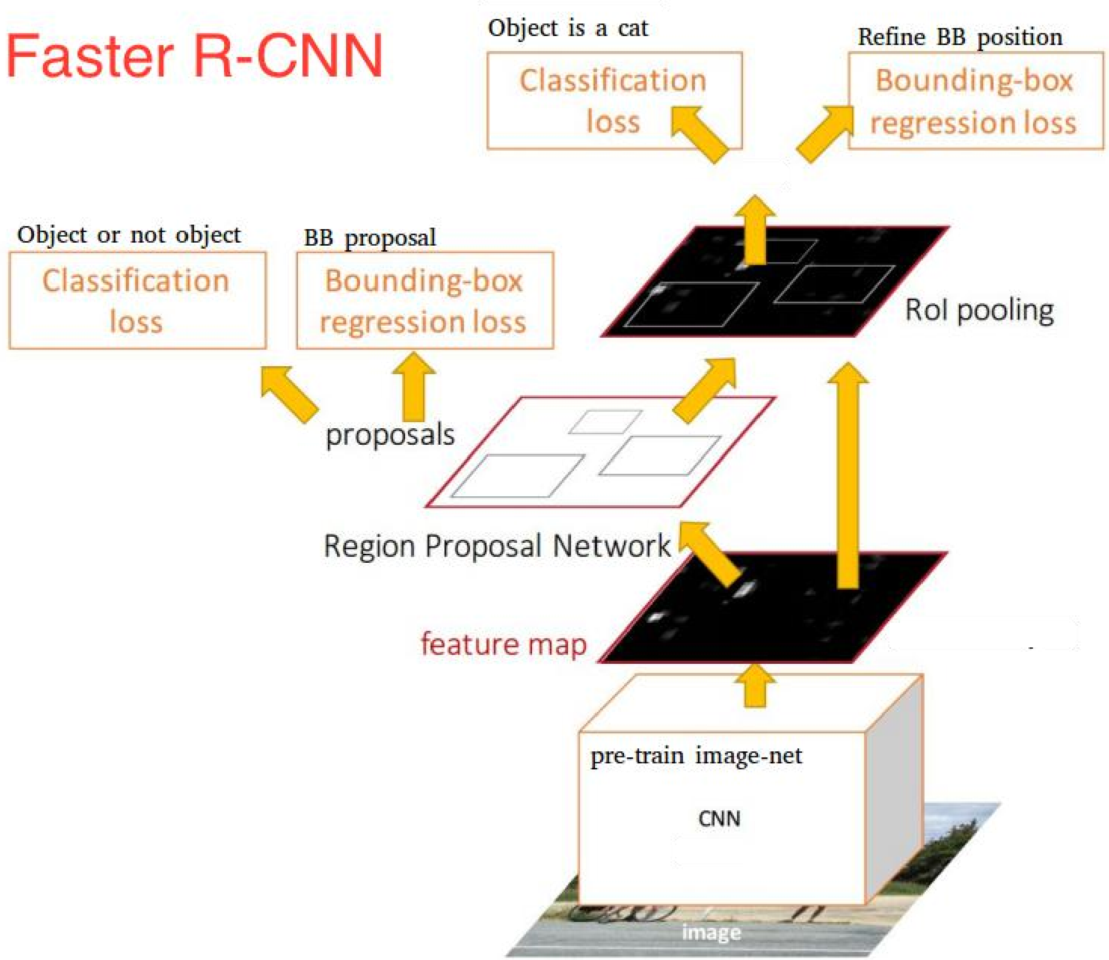
ResNet in FPN





predictor head

Faster-RCNN



Faster R-CNN
Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image

Faster-RCNN

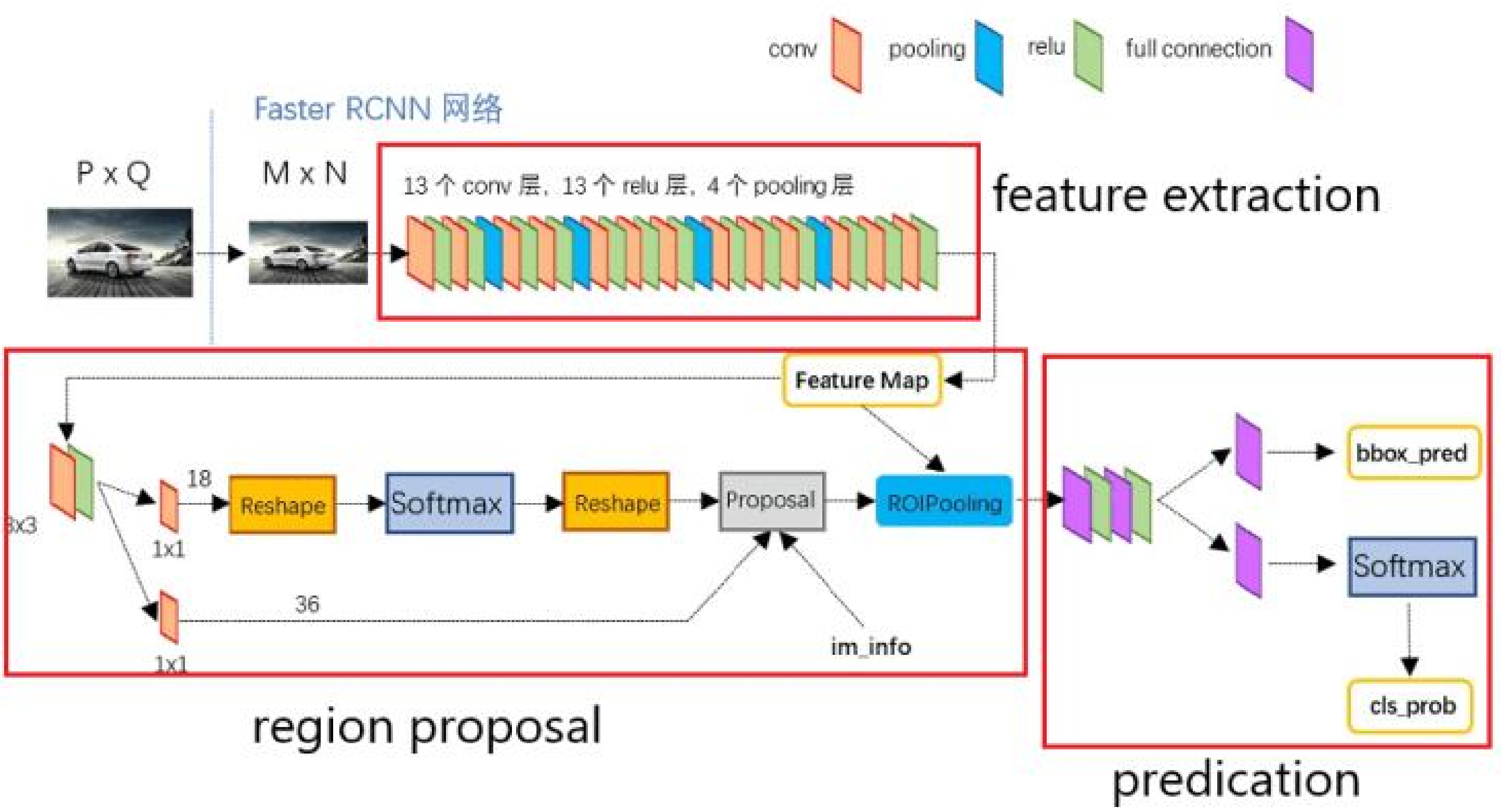


Image source: https://www.yfworld.com/?p=6049

Faster-RCNN with FPN

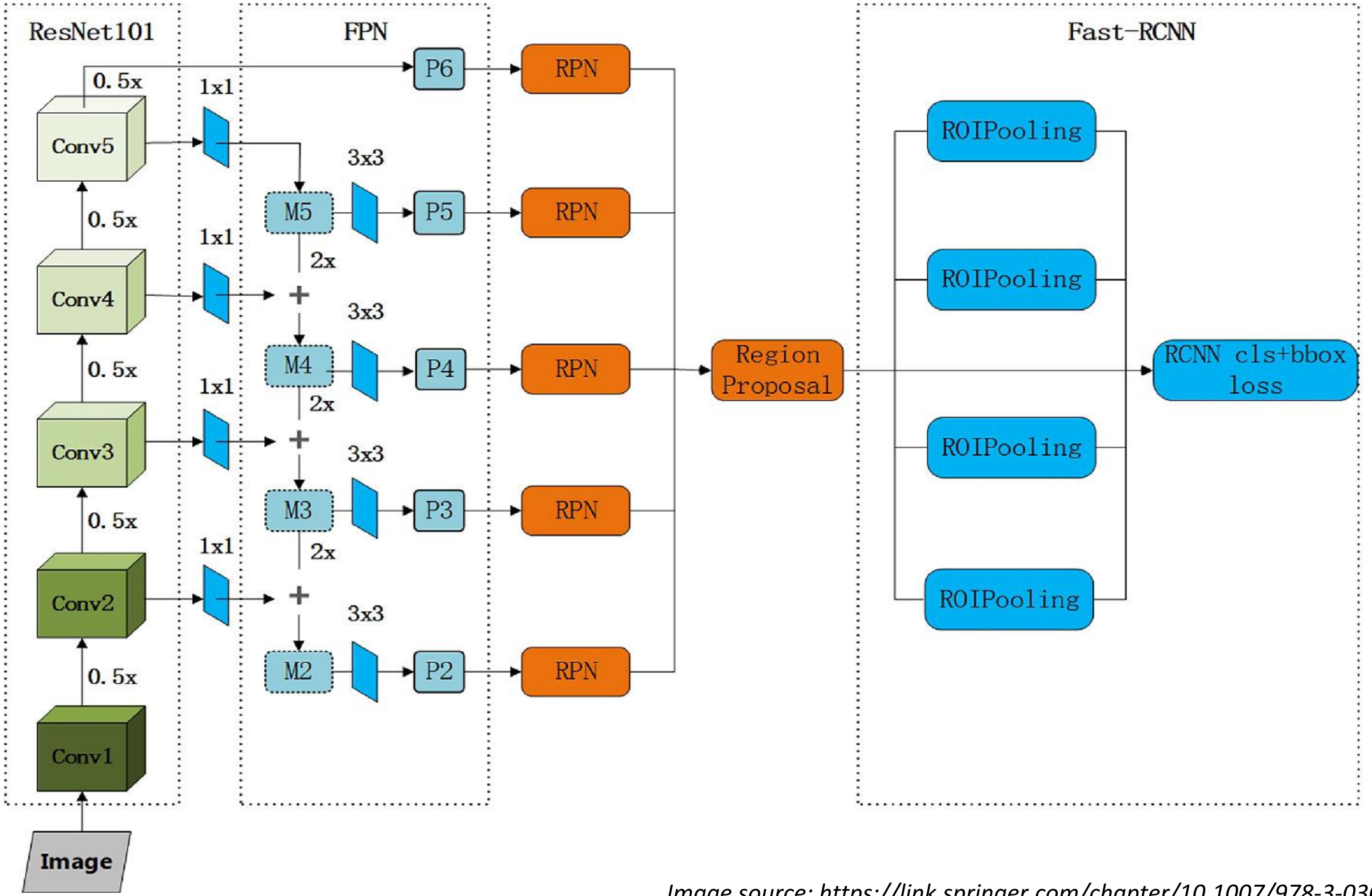
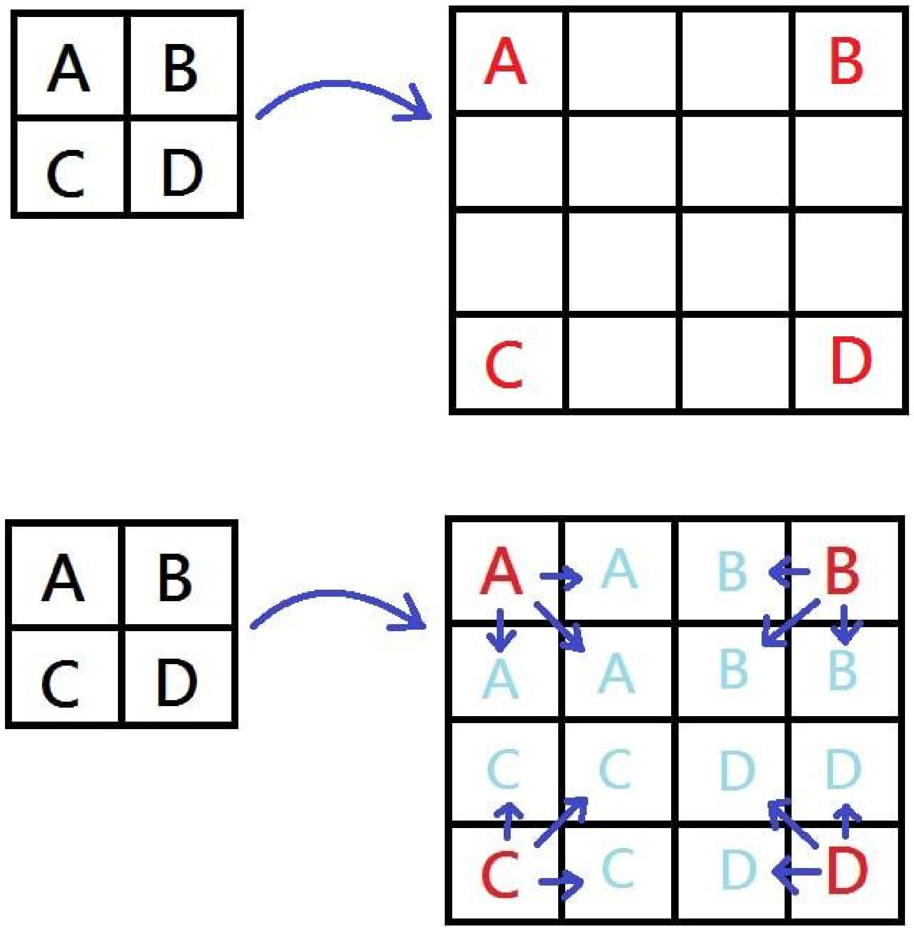
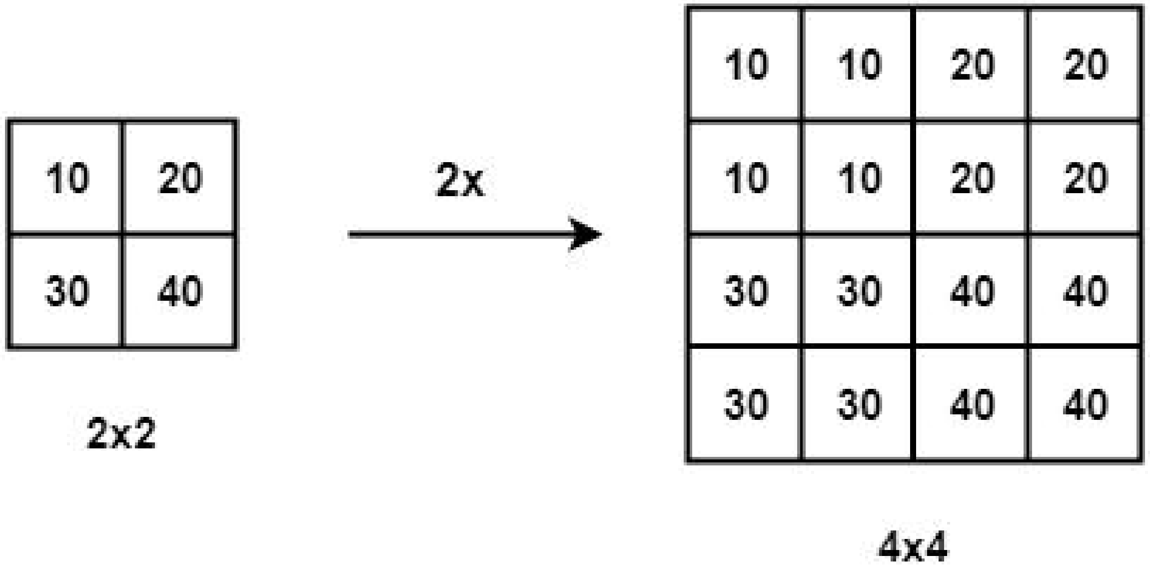


Image source: https://link.springer.com/chapter/10.1007/978-3-030-57884-8_38

Nearest Neighbor Interpolation





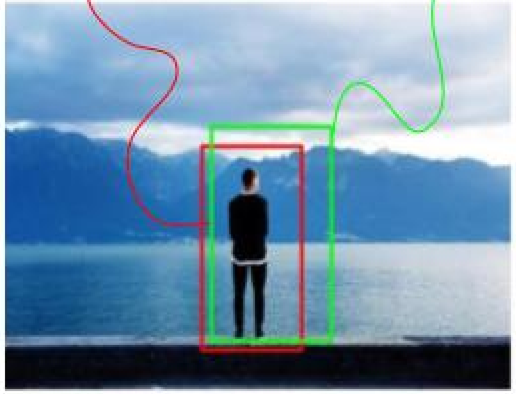
https://jason-chen-1992.weebly.com/home/nearest-neighbor-and-bilinear-interpolation
https://theailearner.com/2018/12/29/image-processing-nearest-neighbour-interpolation/

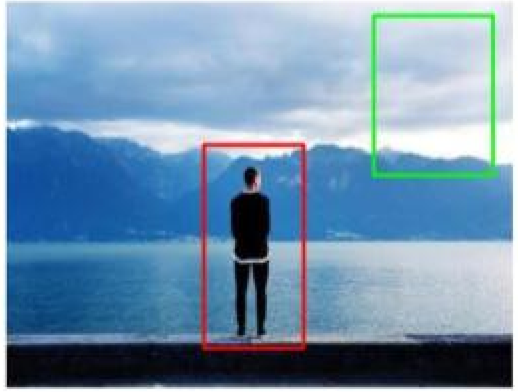
TP, FP and FN

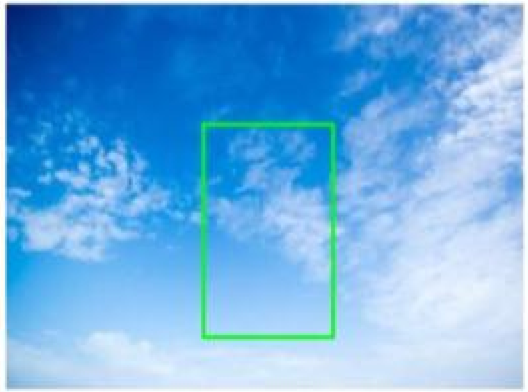
True Positive - TP	False Positive - FP	False Negative - FN

Ground truth box

Predicted box







True Positive - TP: The object **is there**, and the model **detects** it, with an IoU against ground truth box **above** the **threshold**.

False Positive - FP:

Left: The object **is there**, but the predicted box has an IoU against ground truth box **less** than **threshold**.

Right: The object is **not there**, and the model **detects** one.

False Negative - FN: The object **is there**, and the model **doesn't** detect it. The ground truth object has **no** prediction.

Image source: https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html

Metric

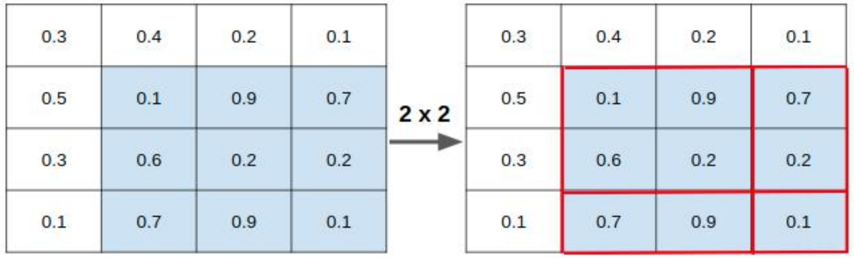
$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 &= \frac{2 \times precision \times recall}{precision + recall} \\
accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \\
specificity &= \frac{TN}{TN + FP}
\end{aligned}
$$

Source: https://www.researchgate.net/post/What_is_the_best_metric_precision_recall_f1_and_accuracy_to_evaluate_the_machine_learning_model_for_imbalanced_data

ROI Pooling

0.3	0.4	0.2	0.1
0.5	0.1	0.9	0.7
0.3	0.6	0.2	0.2
0.1	0.7	0.9	0.1

Input feature map for ROI pooling.



Divide taken region into fixed size grid using proposals (updated coordinates).

max pool ROI

0.9	0.7
0.9	0.1

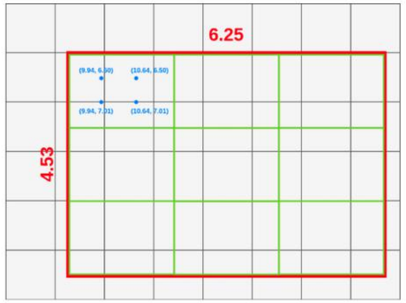
Output from ROI pool based in max pool operation.

https://firiuza.medium.com/roi-pooling-vs-roi-align-65293ab741db

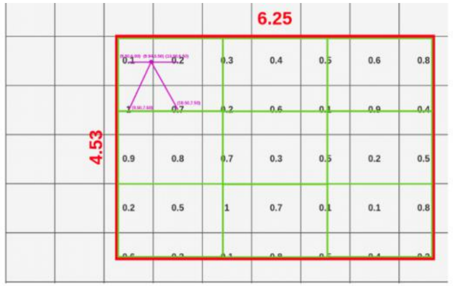
ROI Align

Rol Box size

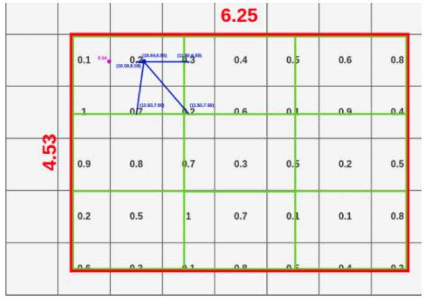
Rol divided into boxes



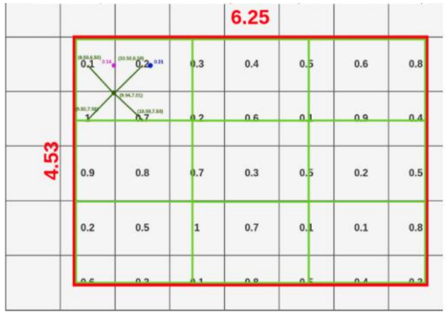
Sampling points distribution



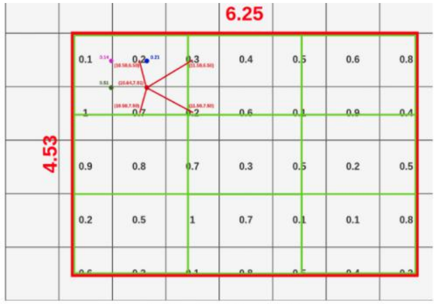
Bilinear Interpolation for the first point



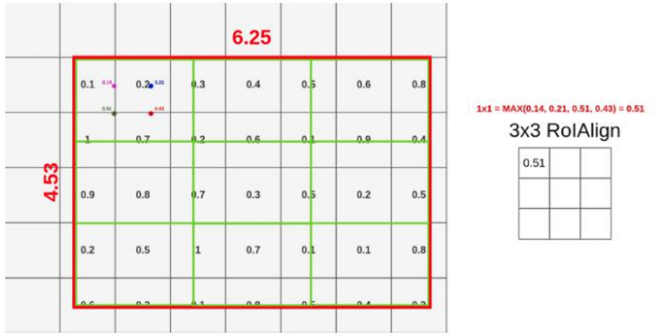
Bilinear Interpolation for the second point



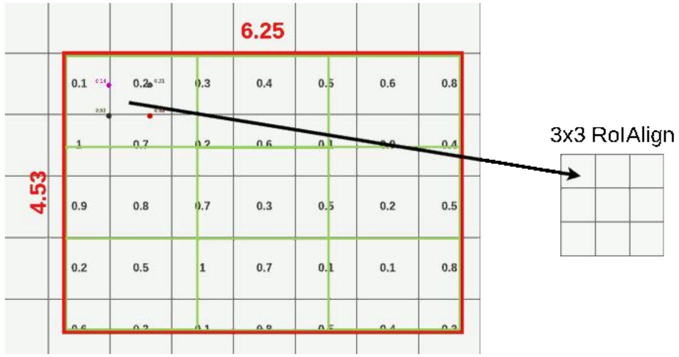
Bilinear Interpolation for the third point



Bilinear Interpolation for the fourth point

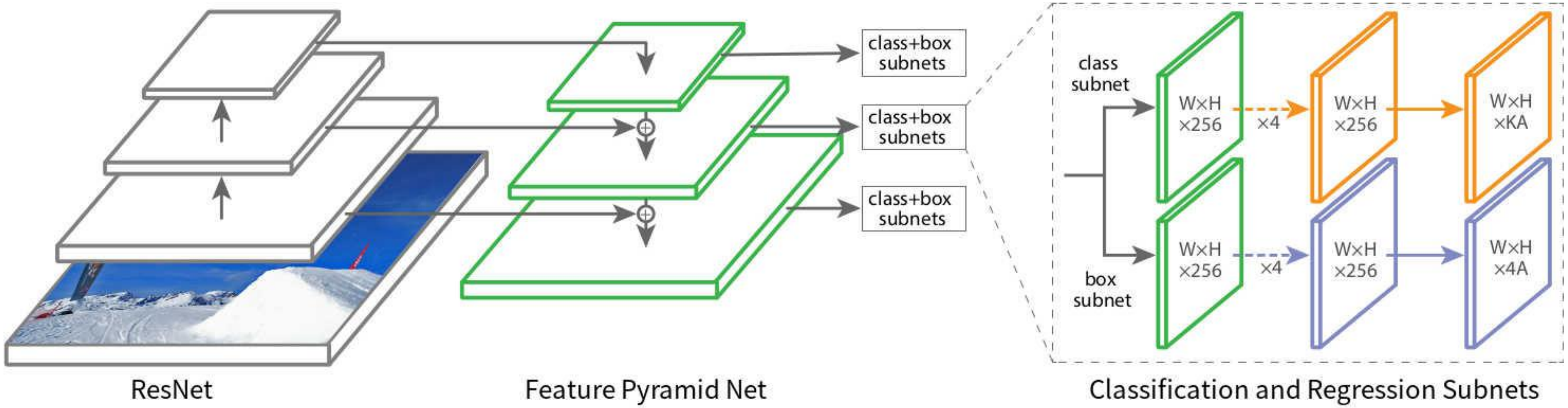


First Box Pooling



https://towardsdatascience.com/understanding-region-of-interest-part-2-roi-align-and-roi-warp-f795196fc193

RetinaNet


class+box subnets
class+box subnets
class+box subnets
class subnet
box subnet
W×H ×256
×4
W×H ×256
W×H ×KA
W×H ×256
×4
W×H ×256
W×H ×4A
ResNet
Feature Pyramid Net
Classification and Regression Subnets

FCOS

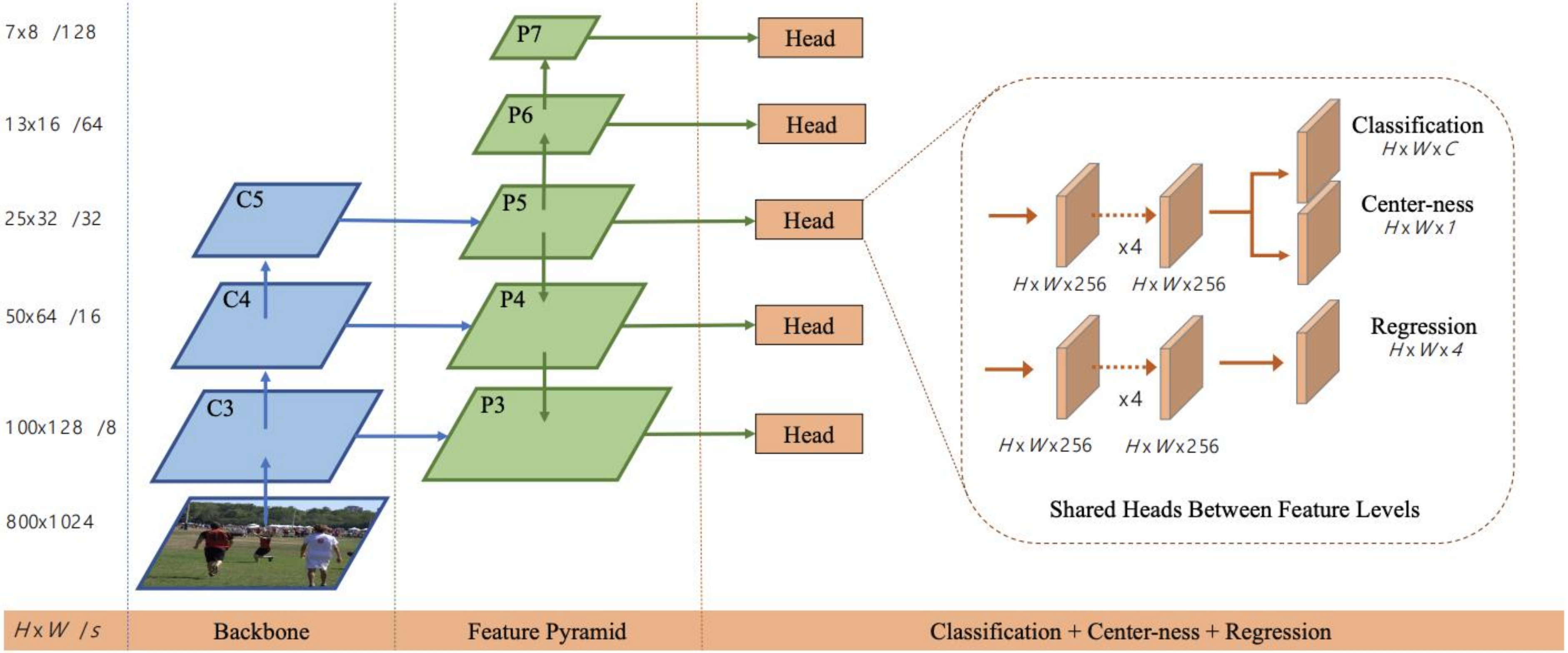
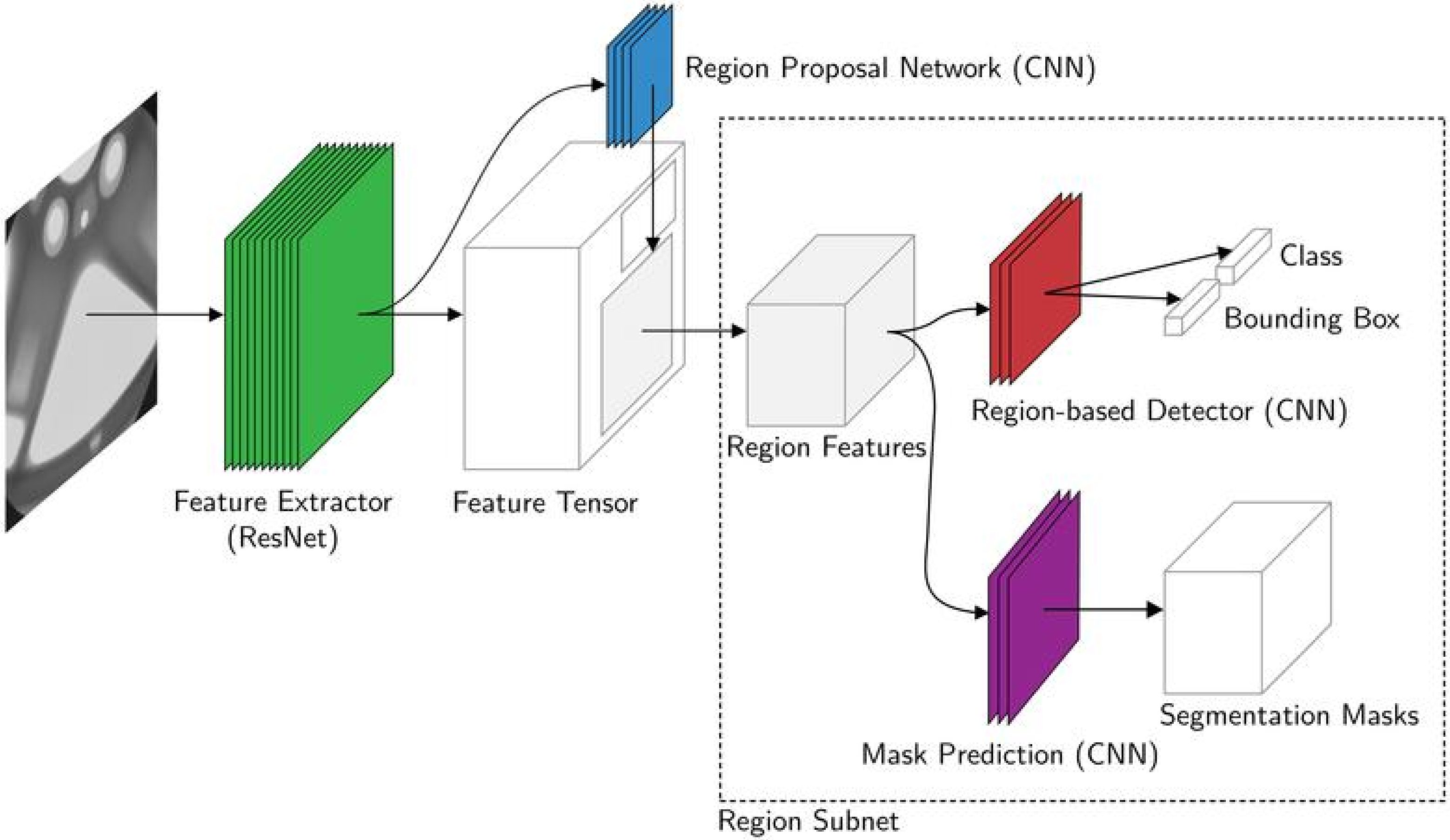


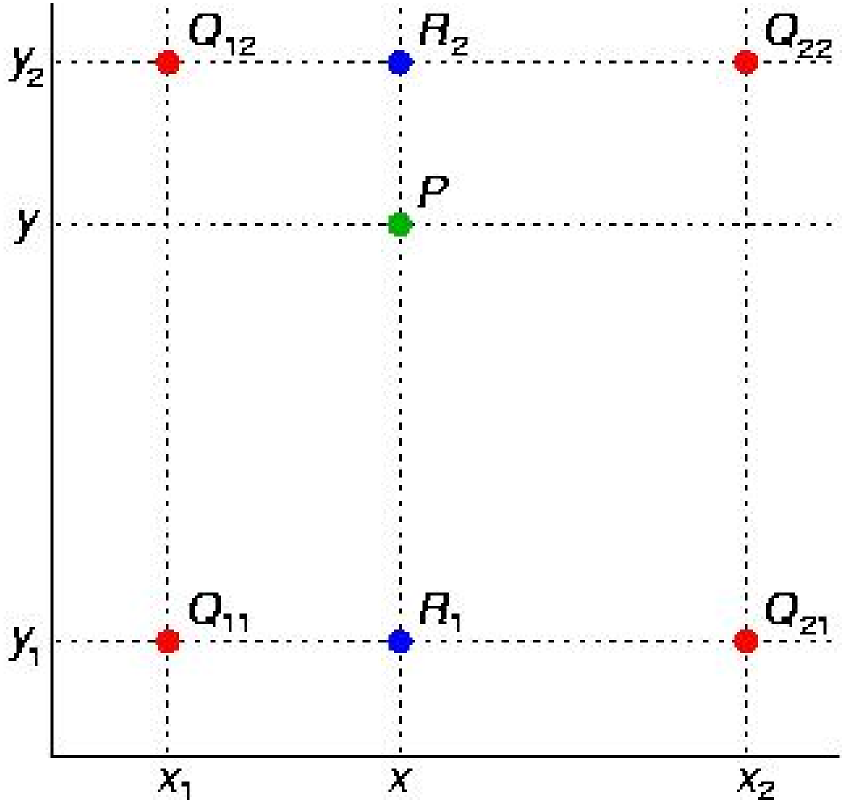
Figure 2 – The network architecture of FCOS, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. '/s' ($s = 8, 16, ..., 128$) is the down-sampling ratio of the feature maps at the level to the input image. As an example, all the numbers are computed with an 800×1024 input.

Mask-RCNN



Region Proposal Network (CNN)
Class
Bounding Box
Region-based Detector (CNN)
Region Features
Feature Extractor
(ResNet)
Feature Tensor
Segmentation Masks
Mask Prediction (CNN)
Region Subnet

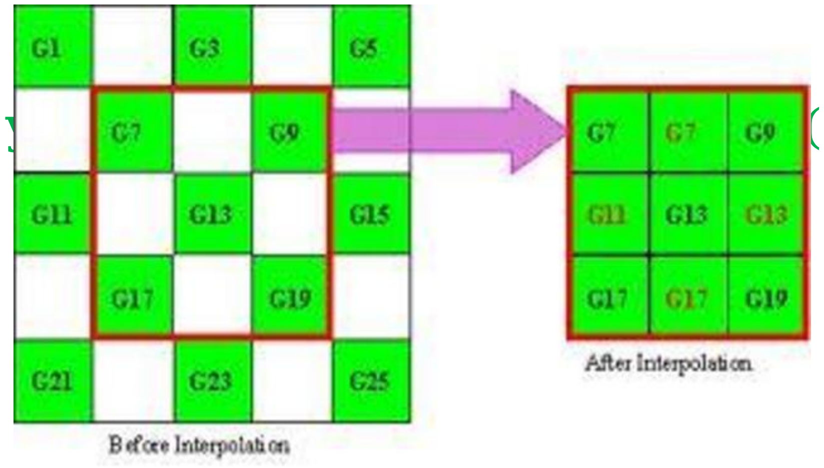
Bilinear interpolation



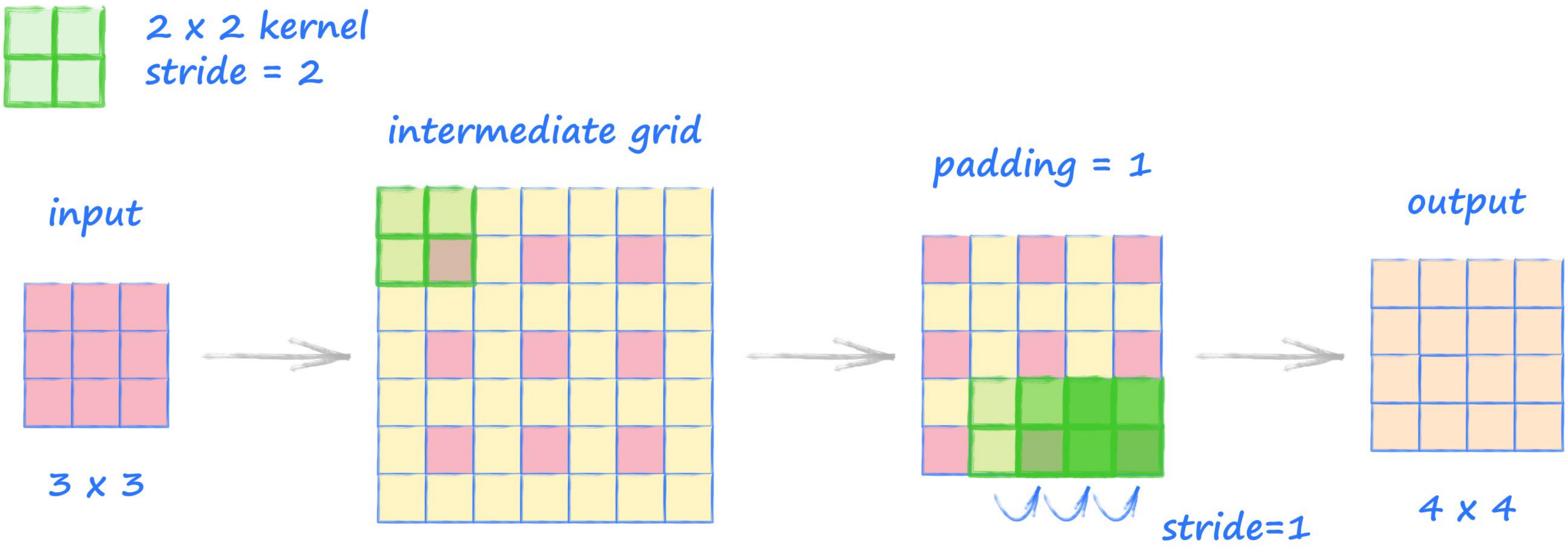
$$R1 = ((x2 - x)/(x2 - x1)) * Q11 + ((x - x1)/(x2 - x1)) * Q21$$

$$R2 = ((x2 - x)/(x2 - x1)) * Q12 + ((x - x1)/(x2 - x1)) * Q22$$

$$P = ((y2 - y \quad \ldots \quad (y2 - y1)) * R2$$

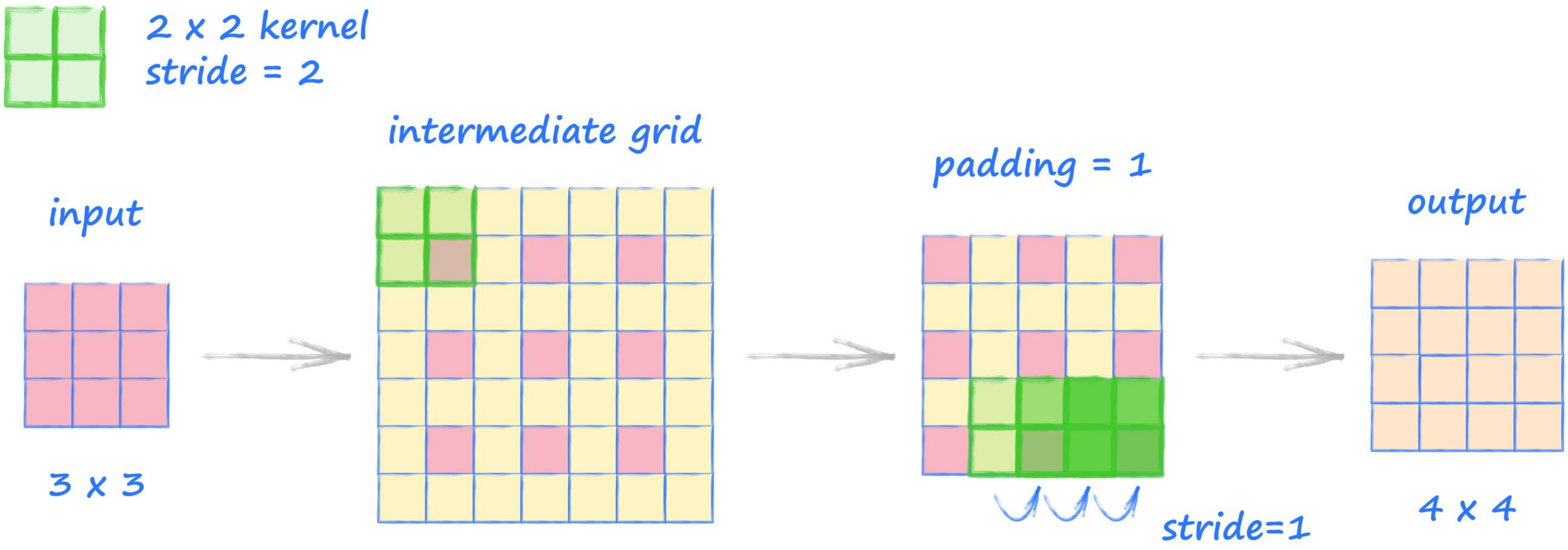


Deconvolution



https://i.stack.imgur.com/GlqLM.png

Deconvolution



https://i.stack.imgur.com/GlqLM.png

Softmax

Input pixels, $\mathbf{x}$

Forward propagation →

Shape: (3, 32, 32)

Feedforward output, $\mathbf{y}_i$

cat	dog	horse
5	4	2
4	2	8
4	4	1

Shape: (3,)

Softmax function →

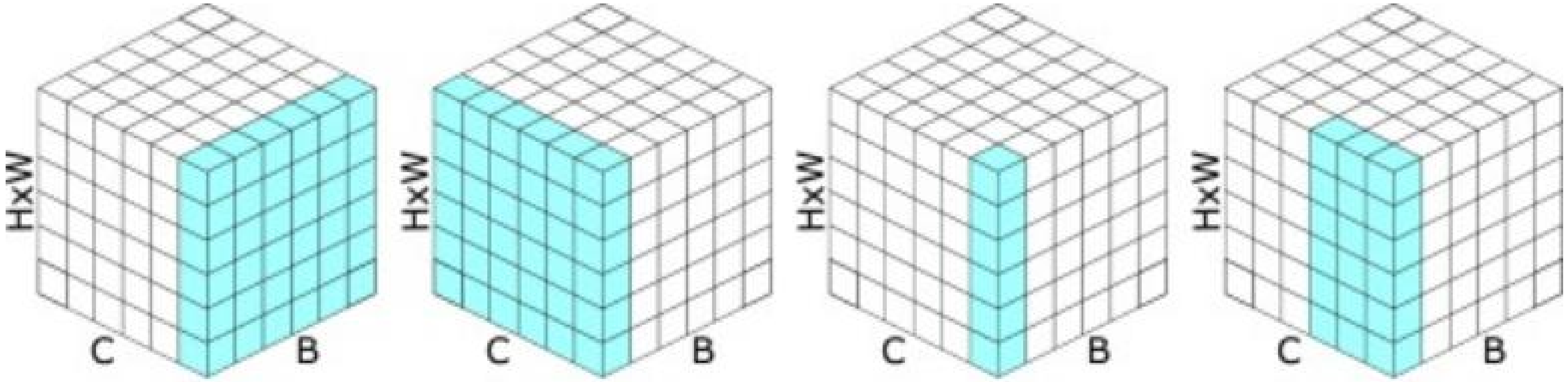
Softmax output, $\mathbf{S}(\mathbf{y}_i)$

cat	dog	horse
0.71	0.26	0.04
0.02	0.00	0.98
0.49	0.49	0.02

Shape: (3,)

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{j} e^{y_i}}$$

Normalization



(a) Batch normalization (b) Layer normalization (c) Instance normalization (d) Group normalization

Adam Optimizer

$$For\ each\ Parameter\ w^j$$

$$(j\ subscript\ dropped\ for\ clarity)$$

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*
g_t : *Gradient at time t along* ω^j
ν_t : *Exponential Average of gradients along* ω_j
s_t : *Exponential Average of squares of gradients along* ω_j
β_1, β_2 : *Hyperparameters*

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

Here, we compute the exponential average of the gradient as well as the squares of the gradient for each parameters (Eq 1, and Eq 2). To decide our learning step, we multiply our learning rate by average of the gradient (as was the case with momentum) and divide it by the root mean square of the exponential average of square of gradients (as was the case with momentum) in equation 3. Then, we add the update.

The hyperparameter *beta1* is generally kept around 0.9 while *beta_2* is kept at 0.99. Epsilon is chosen to be 1e-10 generally.