Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

AI Training Course Series

Object Detection

Lecture 8

Presenter: Yung-Han Chen
Advisor: Juinn-Dar Huang, Ph.D.

August 8, 2024

Outline

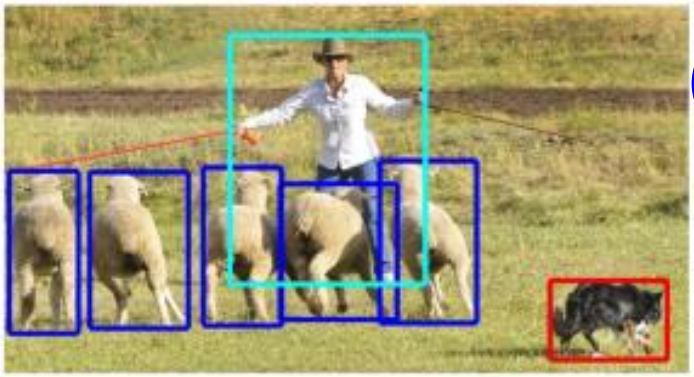
- Introduction
  - What? Why? How?
  - **Objective** and **Metrics** ➔ Always **Market-oriented**
- Data preparation
- Models
- Training and Testing
- Discussion
- Homework
- Reference

What is Objection Detection?

- Find out the **objects we are interested** in the given images

Why Object Detection?

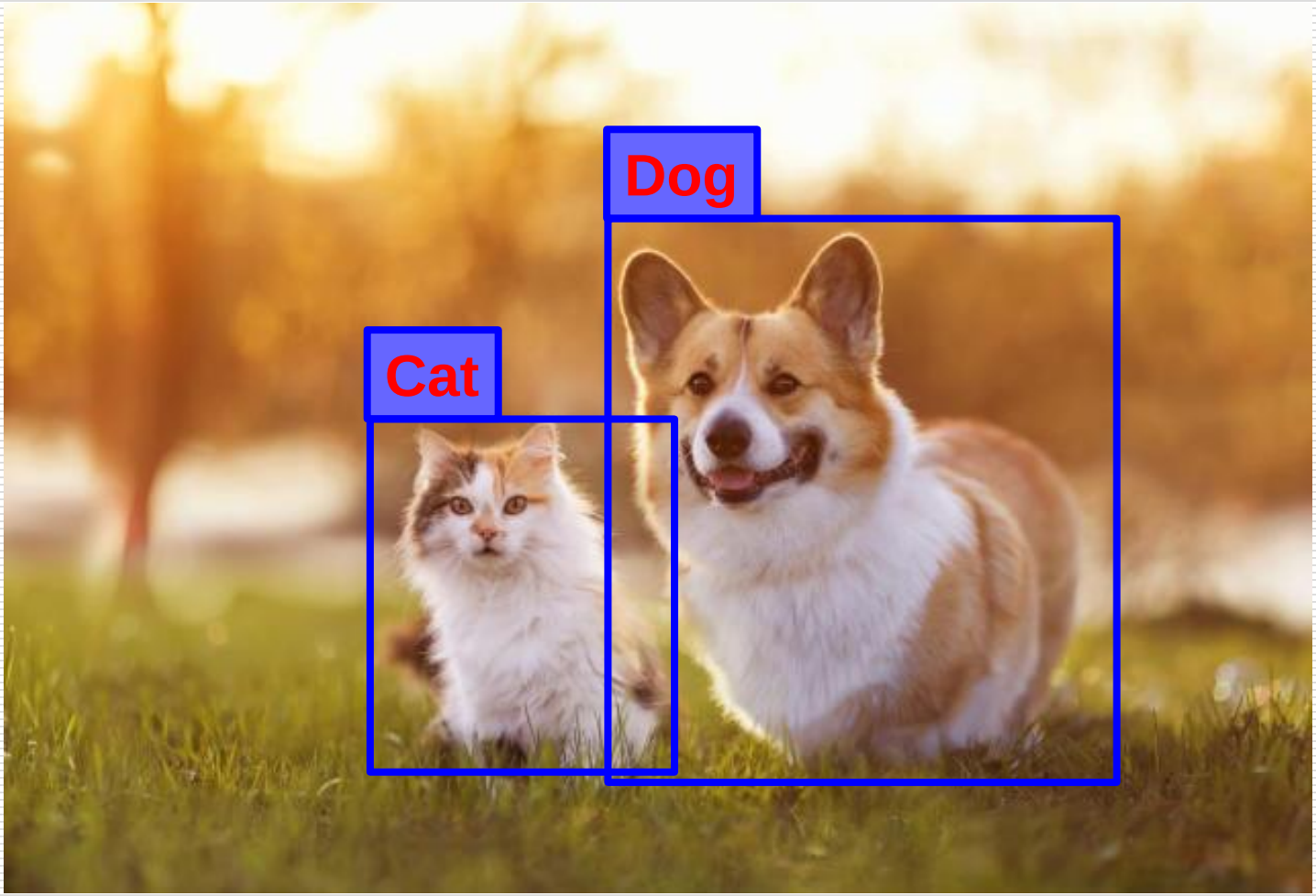
- **Multiple objects** to detect
- **Number of instances** are needed



Hard and Redundant!

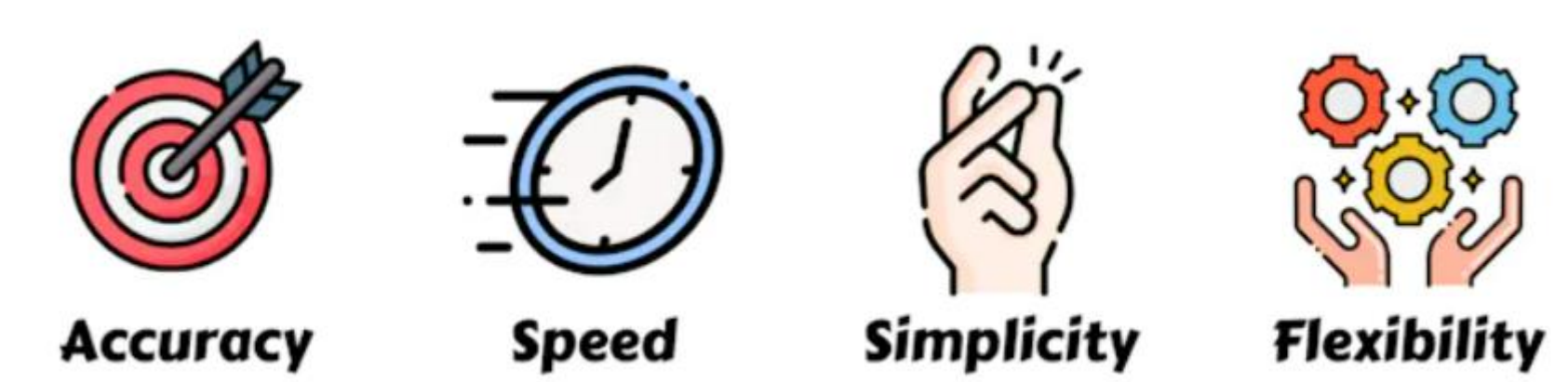
How to do?

- **Localization**



Objective

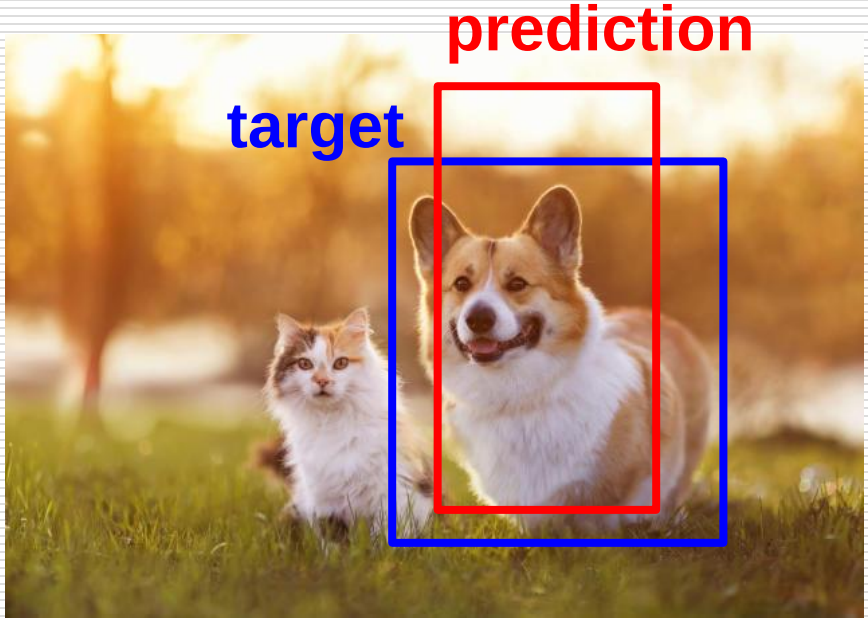
- Common application: drones, self-driving cars, medical images, self checkout system, instances tracking, … ➔ **Fast, Light, Precise**



Metrics (1/3)

- Intersection over Union (IoU)

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Metrics (2/3)

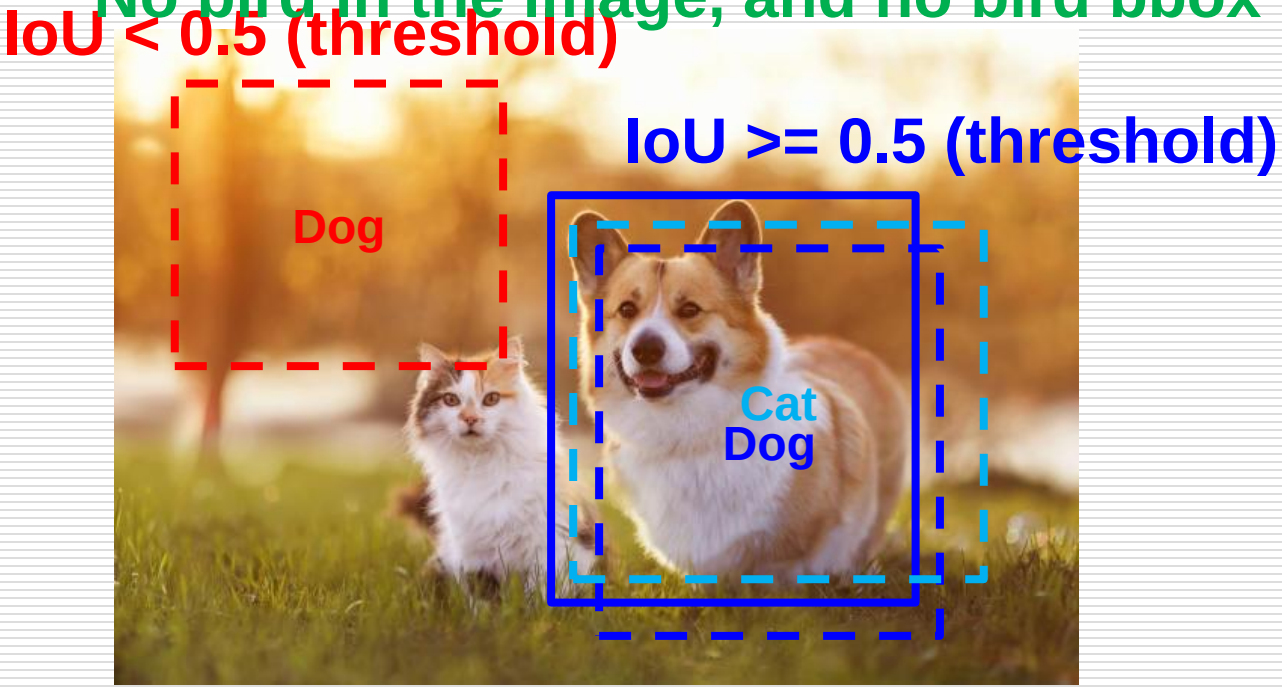
- True Positive (**TP**), False Positive (**FP**)
- True Negative (**TN**), False Negative (**FN**)
- What if …?

For "Cat" ➔ FP

For "Dog" ➔ FN

A dog in the image, but no dog bbox

No bird in the image, and no bird bbox



Metrics (3/3)

- Precision and Recall

$$\text{Precision} = \frac{tp}{tp + fp}$$

➔ # of hits

➔ # of predictions

$$\text{Recall} = \frac{tp}{tp + fn}$$

➔ # of hits

➔ # of targets

Outline

- Introduction
- Data preparation
  - Dataset
  - Data Augmentation
- Models
- Training and Testing
- Discussion
- Homework
- Reference

Dataset

- COCO
  - Object detection, keypoint detection, segmentation, panoptic segmentation, image captioning
  - 0.2 million images with labels
  - 1.5 million object instances
  - Instance: 80 classes

COCO Format

- COCO[1]
  - **annotations**
    - Instances.json
  - images
    - Train / Val
      - <id>.jpg
  - **labels**
    - Train / Val
      - <id>.txt
  - **.cache ➔ Make training faster**

COCO Caption

- Images with captions

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

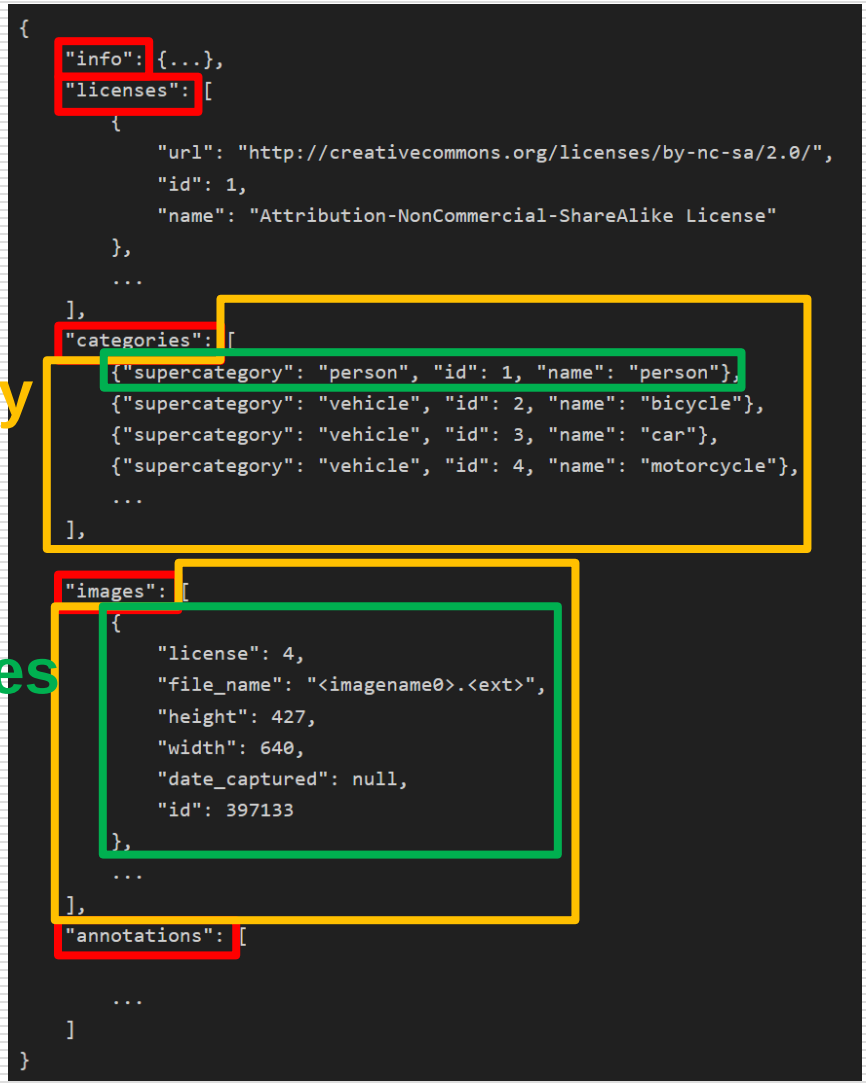
Annotations

- Data structure: **dict** of **lists** of **dicts**

Keys of a dict

Value to the key

Dict of Instances



Instance Information

```
image{
   "id"                : int,
   "width"             : int,
   "height"            : int,
   "file_name"         : str,
   "license"           : int,
   "flickr_url"        : str,
   "coco_url"          : str,
   "date_captured"     : datetime,
}
```

→ id of an image

```
categories[{
   "id"                : int,
   "name"              : str,
   "supercategory"     : str,
}]
```

→ id of a category

```
annotation{
   "id"                : int,
   "image_id"          : int,
   "category_id"       : int,
   "segmentation"      : RLE or [polygon],
   "area"              : float,
   "bbox"              : [x,y,width,height],
   "iscrowd"           : 0 or 1,
}
```

→ x, y, x, y, x, y, …

Label

- Two formats in <id>.txt
  - <id_categories> x, y, x, y, x, y, …
  - <id_categories> x, y, w, h

x = x_coord / width
y = y_coord / height

000000000776.txt

```
1 77 0.00672897 0.202703 0.0370561 0.171172 0.0606308 0.159906 0.0740888 0.146391 0.0942991 0.128375 0.121238 0.123875 0.181869 0
  .0900938 0.276168 0.0878437 0.346893 0.0923438 0.400771 0.103609 0.42771 0.148656 0.404136 0.168922 0.461402 0.234234 0.464766 0
  .290547 0.464766 0.319812 0.464766 0.358109 0.511916 0.376125 0.58264 0.373875 0.65 0.367109 0.687033 0.351344 0.707243 0.378375 0
  .757757 0.448203 0.808294 0.558563 0.804907 0.662156 0.774603 0.731984 0.683668 0.786031 0.596121 0.824328 0.535491 0.801797 0
  .518645 0.75225 0.572547 0.709453 0.609579 0.673422 0.612944 0.644141 0.599486 0.581078 0.575911 0.554047 0.569159 0.513516 0
  .559065 0.490984 0.525374 0.463969 0.498435 0.445953 0.468131 0.434688 0.458037 0.434688 0.431075 0.425672 0.404136 0.441437 0
  .346893 0.452703 0.319953 0.457203 0.303107 0.459453 0.255958 0.443687 0.208808 0.432437 0.151542 0.427922 0.121238 0.427922 0
  .0673598 0.430188 0.00336449 0.450453
2 77 0.0235748 0.434688 0.239112 0.434688 0.319953 0.463969 0.481612 0.441437 0.548949 0.531531 0.49507 0.646391 0.481612 0.682438 0
  .606215 0.675672 0.511916 0.779281 0.569159 0.835578 0.656729 0.828828 0.707243 0.797297 0.747664 0.981984 0.0269393 0.988734 0
  .00672897 0.463969
3 77 0.329299 0.094375 0.221776 0.0516875 0.231869 0.00898437 0.366262 0.0134844 0.887126 0.0112344 0.907266 0.0651719 0.940888 0
  .094375 0.998014 0.098875 0.998014 0.669656 0.964393 0.638203 0.944229 0.739328 0.789673 0.79775 0.729182 0.860672 0.719112 0
  .786516 0.809836 0.680906 0.809836 0.582016 0.779579 0.474156 0.688855 0.352813 0.598131 0.370781 0.497313 0.370781 0.473808 0
  .334828 0.473808 0.251688 0.443551 0.2 0.423388 0.177531 0.430117 0.152812
4 59 0.980164 0.655703 1 0.675828 0.996893 0.997984 0.749346 1 0.74264 0.852563 0.799509 0.812297 0.896519 0.774266 0.940023 0.738469
  0.990187 0.68925 0.980164 0.655703 0.314463 0.0897031 0.899883 0.0717969 0.899883 0.0203438 0.889836 0.0046875 0.996893 0.0046875
  1 0.0986563 0.960093 0.103125 0.926636 0.0919375 0.899883 0.0717969 0.314463 0.0897031 0.19736 0.0919375 0.160561 0.100891 0
  .120421 0.134438 0.0200701 0.181422 0.00334112 0.192609 0.00334112 0.00245313 0.270958 0.00021875 0.260935 0.0203438 0.234159 0
  .0337656 0.260935 0.060625 0.294393 0.0717969 0.314463 0.0897031
5
```

Code (YOLOv7)

- In utils/datasets.py

```
class LoadImagesAndLabels(Dataset):  # for training/testing
    def __init__(self, path, img_size=640, batch_size=16, augment=False, hyp=None, rect=False, image_we
                 cache_images=False, single_cls=False, stride=32, pad=0.0, prefix=''):
        def cache_labels(self, path=Path('./labels.cache'), prefix='')

    def __getitem__(self, index):
        if mosaic:
            load_mosaic(self, index)
        else:
            load_image(self, index)
        random_perspective(img, labels,
                           degrees=hyp['degrees'],
                           translate=hyp['translate'],
                           scale=hyp['scale'],
                           shear=hyp['shear'],
                           perspective=hyp['perspective'])
        augment_hsv(img, hgain=hyp['hsv_h'], sgain=hyp['hsv_s'], vgain=hyp['hsv_v'])
        pastein(img, labels, sample_labels, sample_images, sample_masks)

    @staticmethod
    def collate_fn(batch):
```

Load paths to cache

Read images

Data Augmentation

Get mini-batch

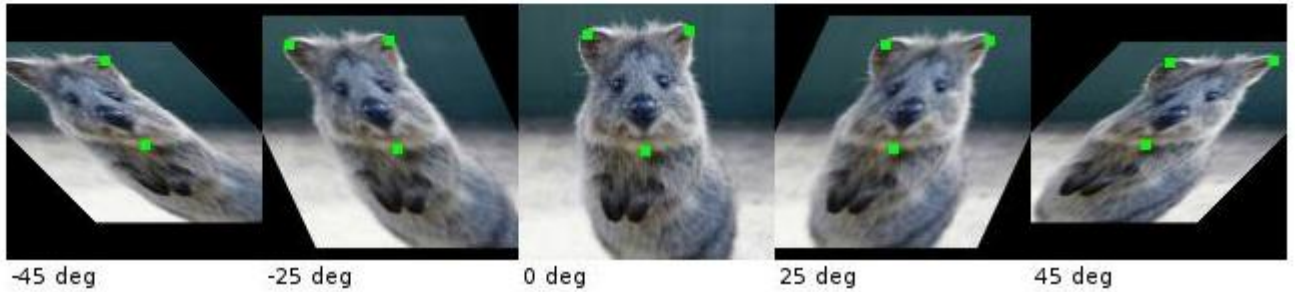
Data Augmentation (1/3)

- Affine (random perspective)

Affine: Translate

Affine: Rotate

Affine: Shear

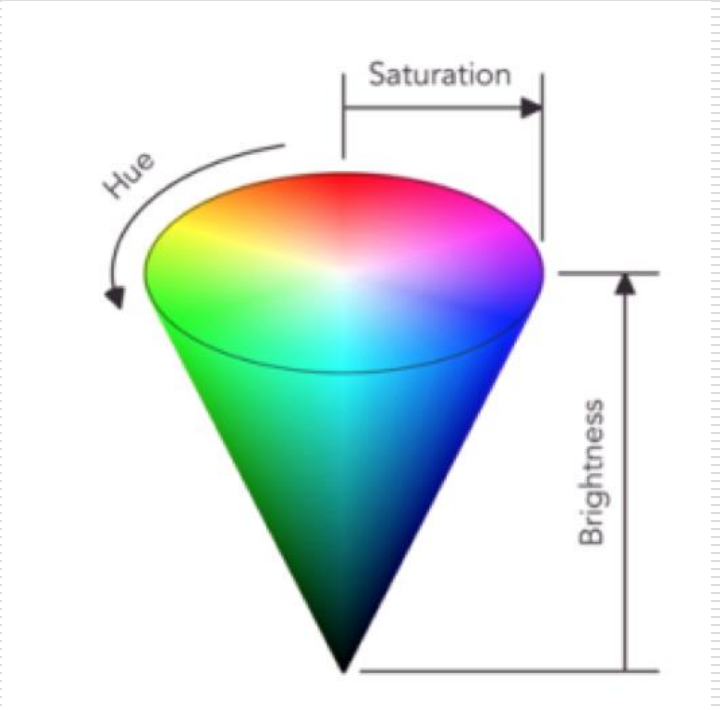


Data Augmentation (2/3)

- HSV (Hue, Saturation, Value)

Original

Hue=120, Saturation=2, Value=1

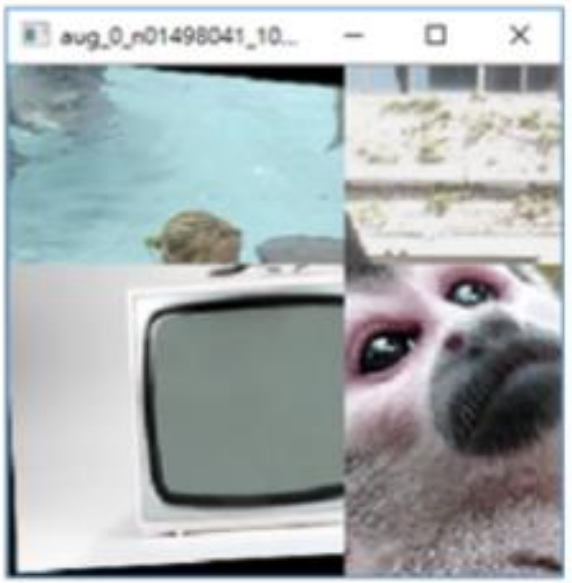


Data Augmentation (3/3)

- Mix

MixUp

CutMix



Mosaic

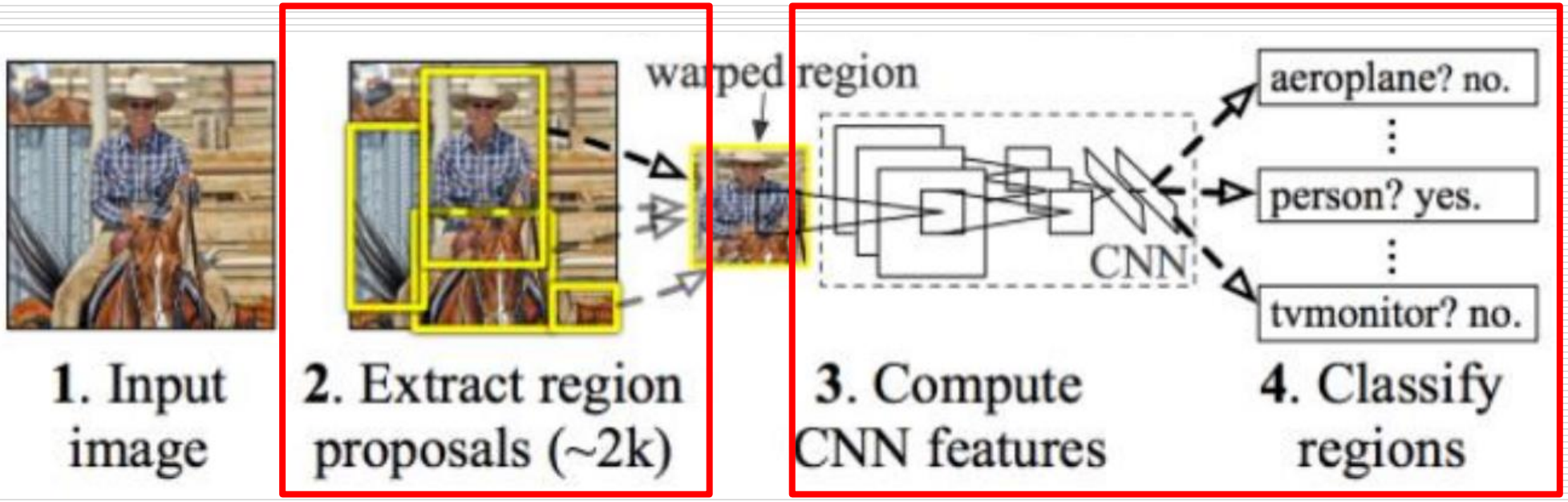
Outline

- Introduction
- Data preparation
- Models
  - **CNN-based: RCNN Family, YOLO Family, CenterNet**
  - **Transformer-based: DETR**
- Training and Testing
- Discussion
- Homework
- Reference

Really trivial.
Remember the points.

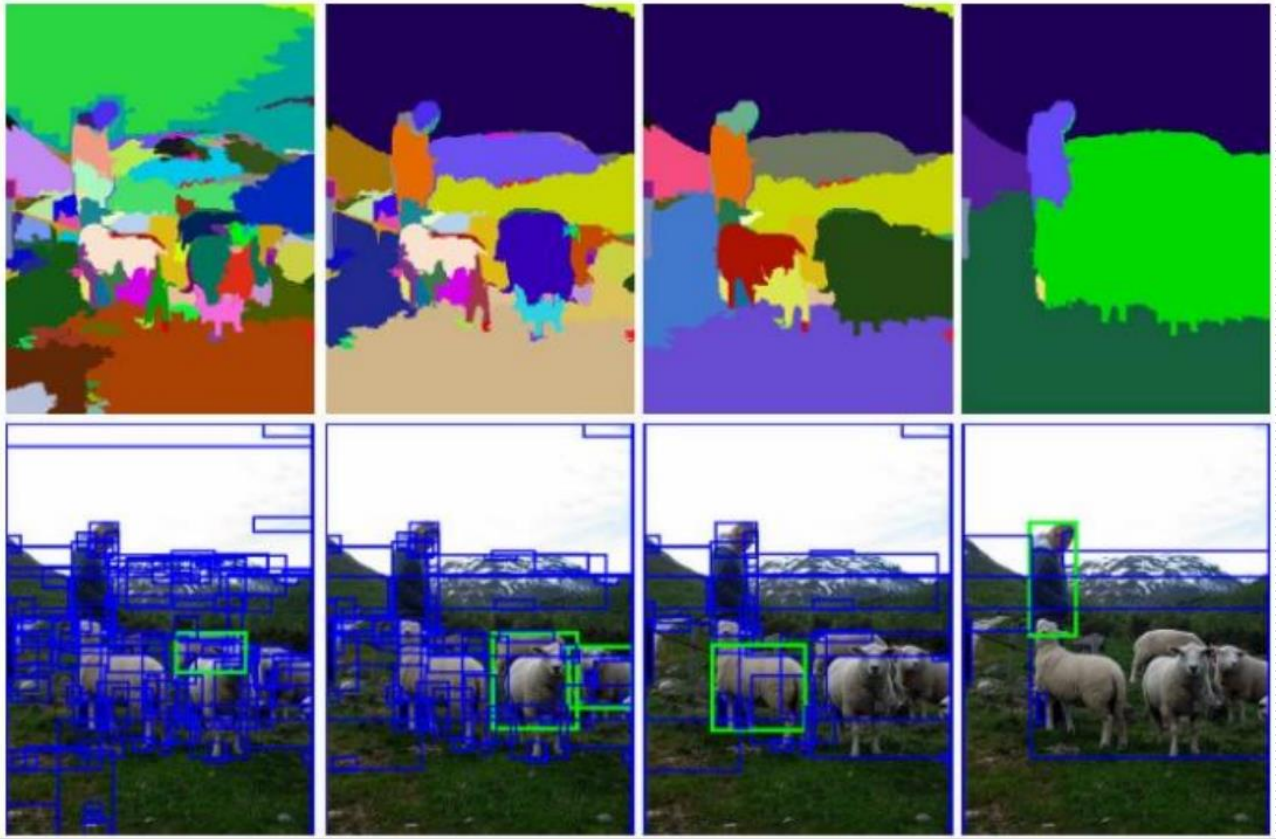
R-CNN

- **Rich feature hierarchies for accurate object detection and semantic segmentation[2]**
- **Two-stage**



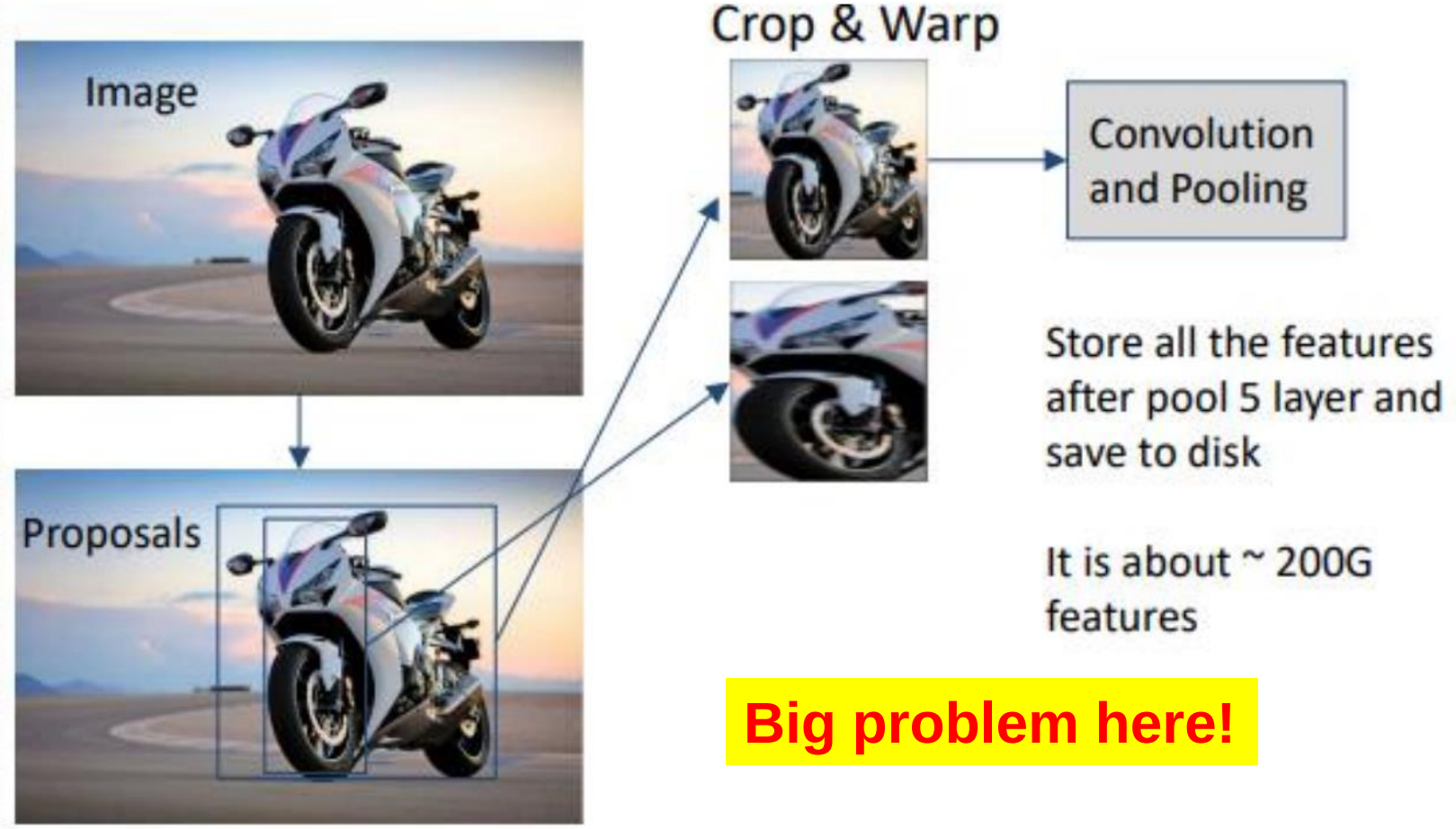
Step 1: Selective Search

- Adopting HAC algorithm (segmentation) can generate **2k region proposals**
- A mask to a bbox



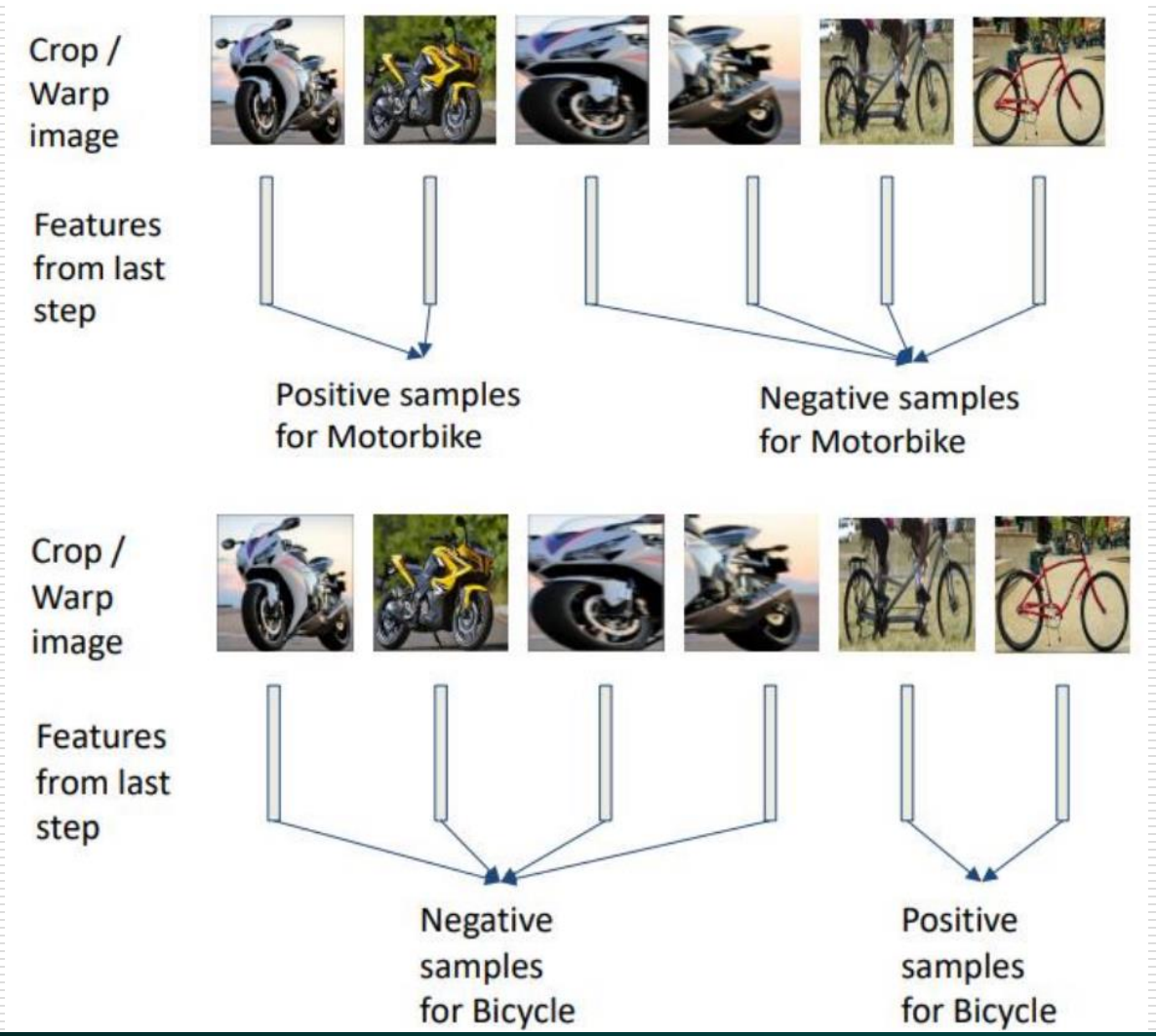
Step 2: Feature Extraction

- Resizing **each region proposal** and going through CNN to extract features



Step 3: Per-Class SVM

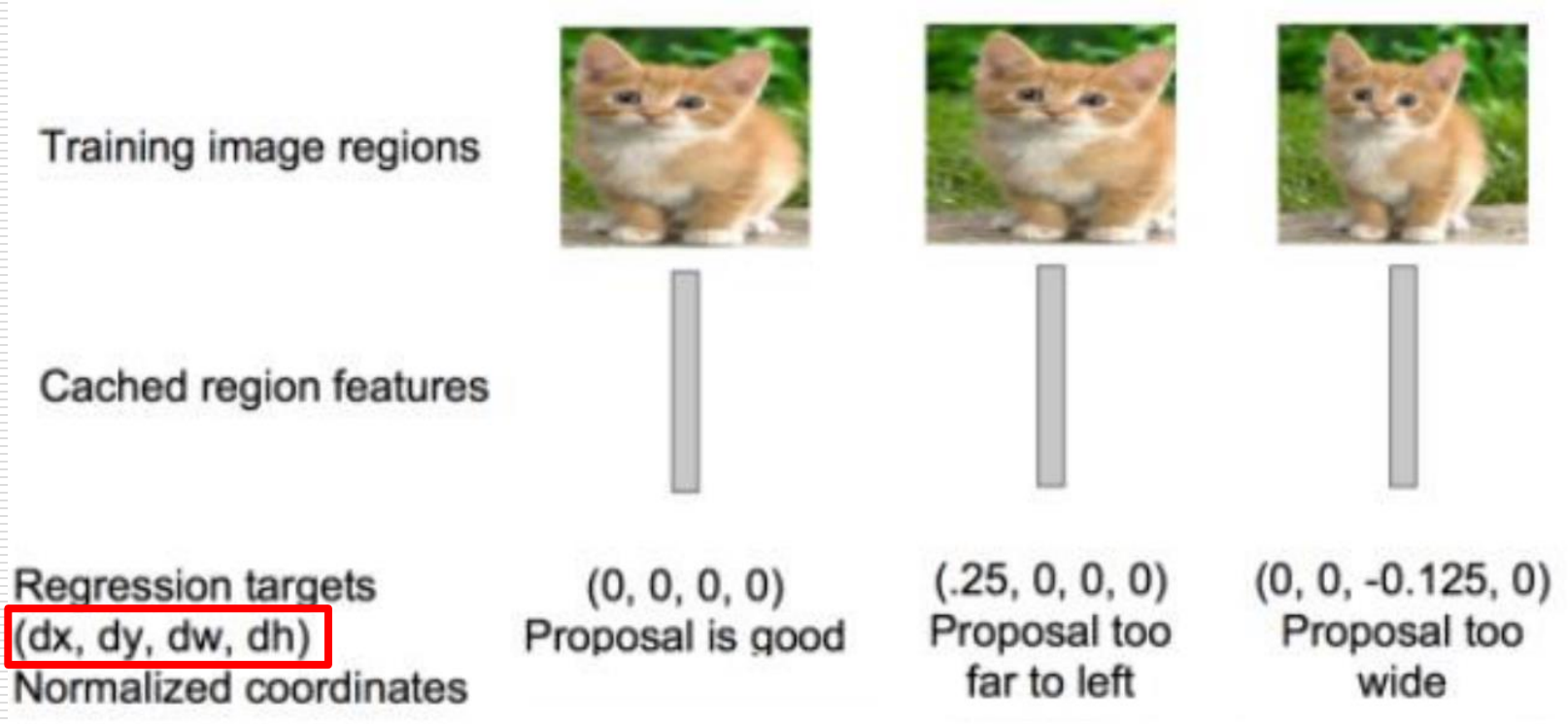
- Positive sample if **IoU(RoI, tgt)>0.3**, negative otherwise



Step 4: Bbox Regression

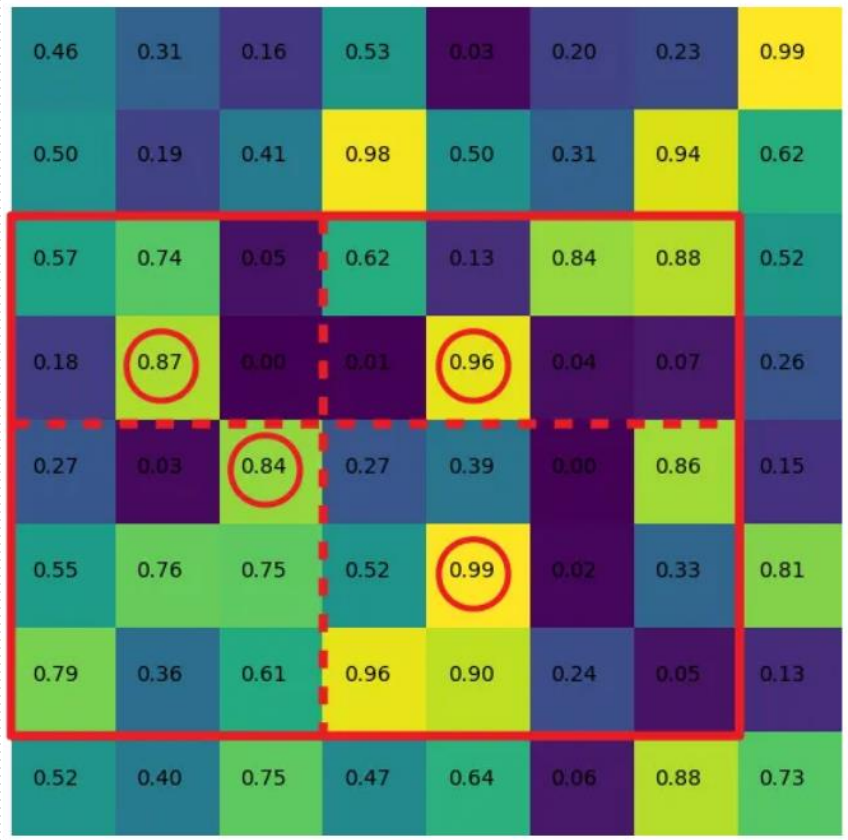
- Computing **new coordinates** of each RoI(bbox)
- Make closed RoIs **overlapped**

$$x' = x + dx \quad w' = w * e^{dw}$$
$$y' = y + dy \quad h' = h * e^{dh}$$

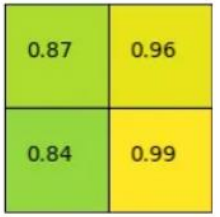


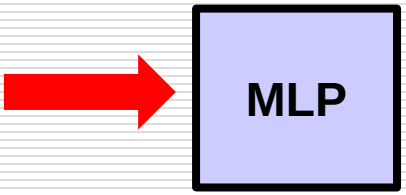
Fast R-CNN

- **Fast R-CNN[3]**
- All the RoIs share feature maps ➔ Only **one time** of features extraction is needed



RoI map is 7x5, we want 2x2, HOW?

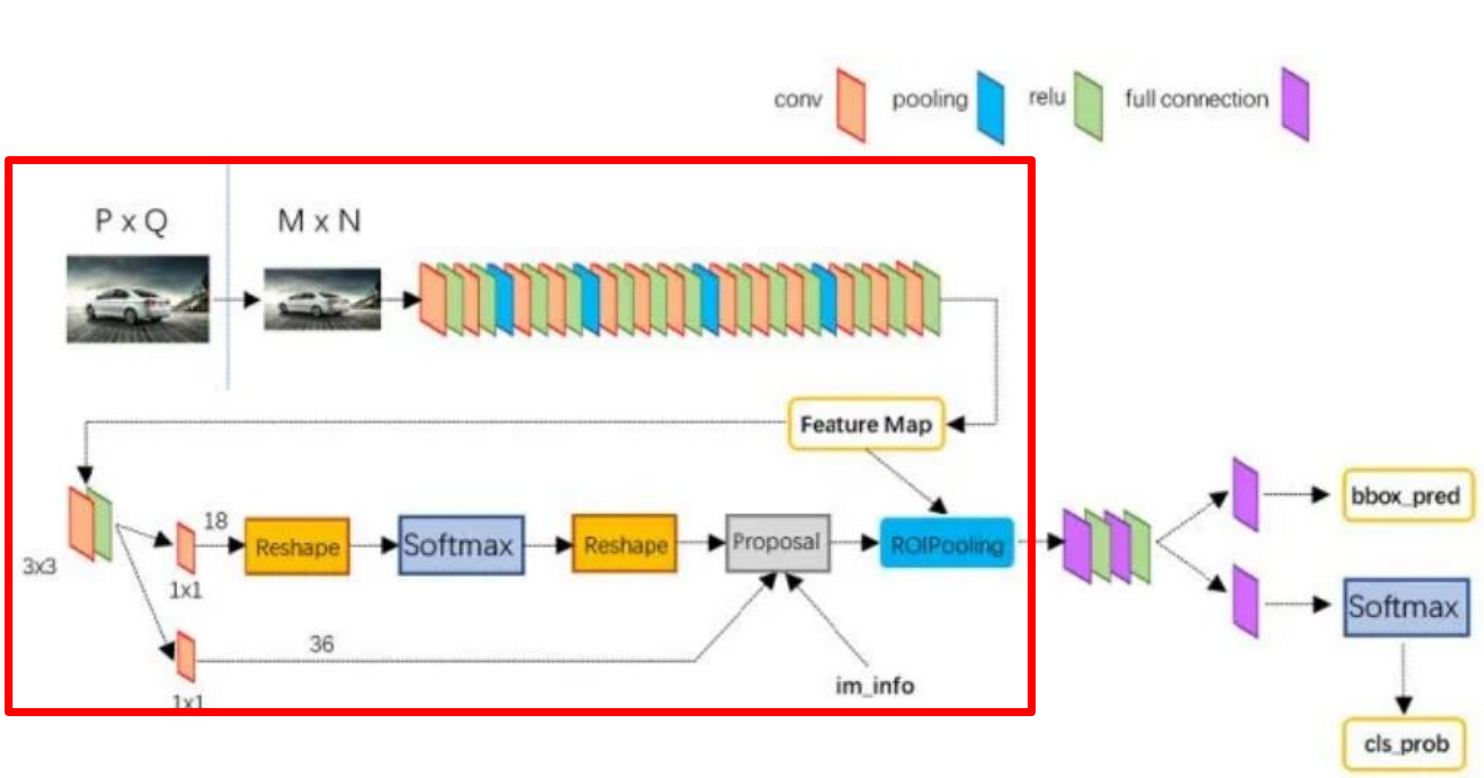




Classification
Localization

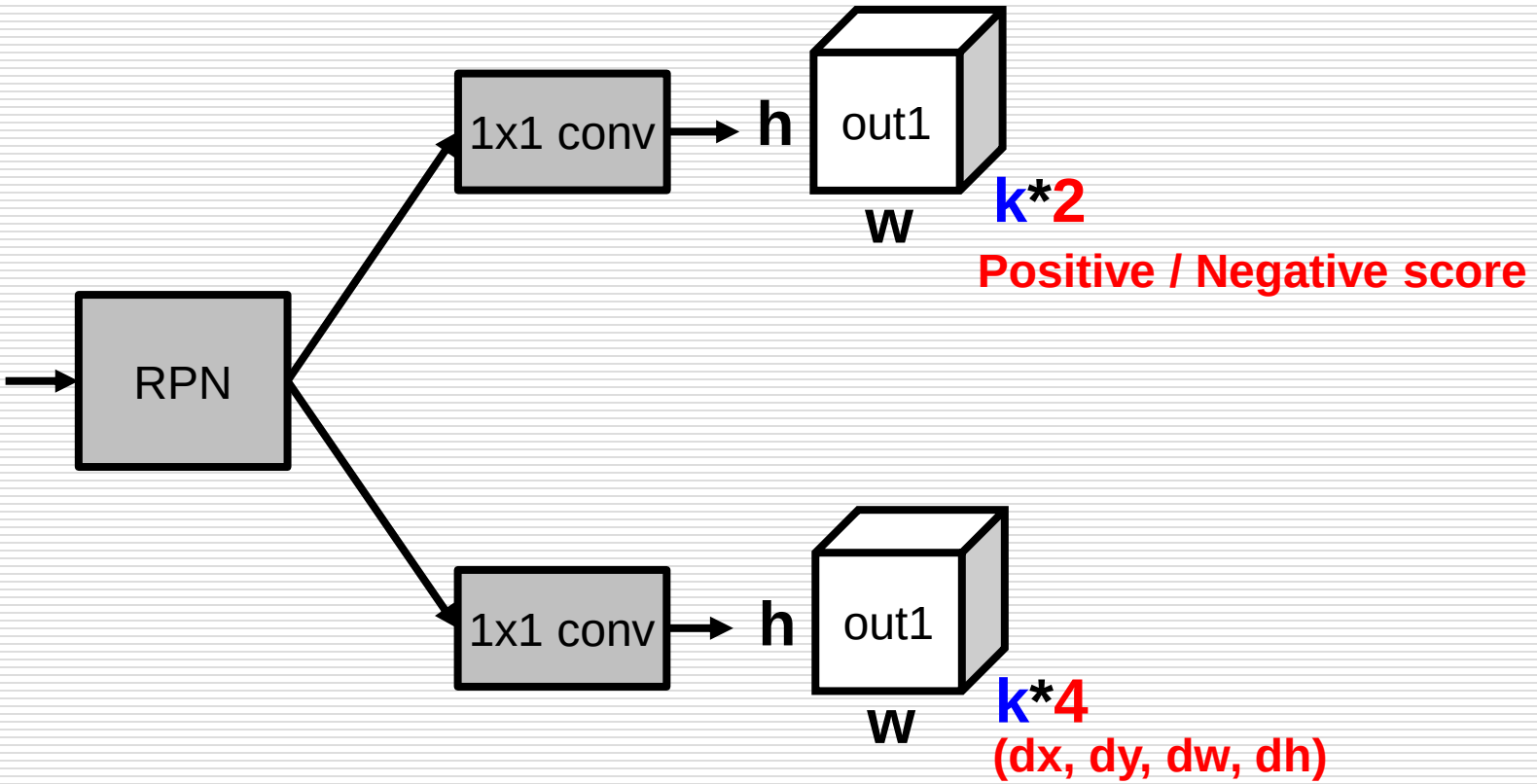
Faster R-CNN

- **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[3]**
- Replace HAC with **Region proposals Network (RPN)**



RPN

- Input an image, and **output RoIs**

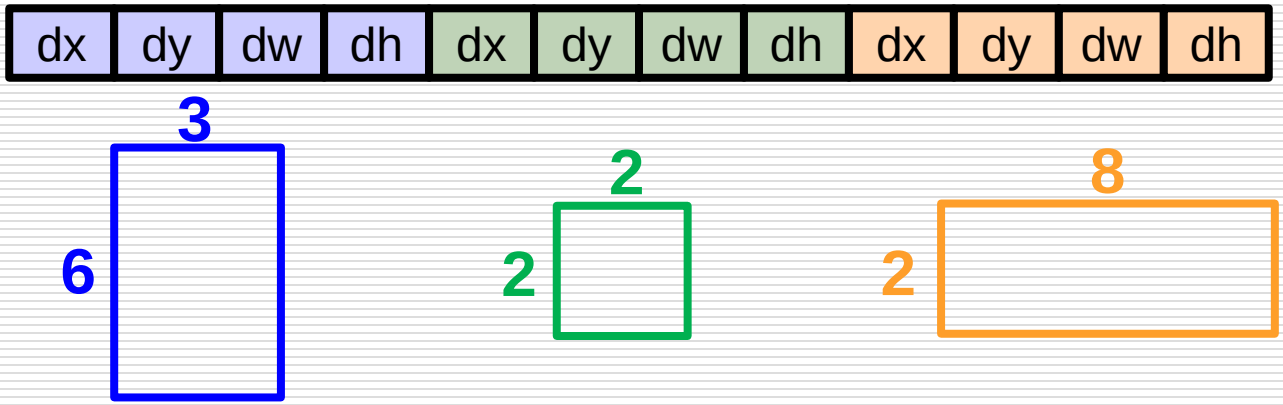


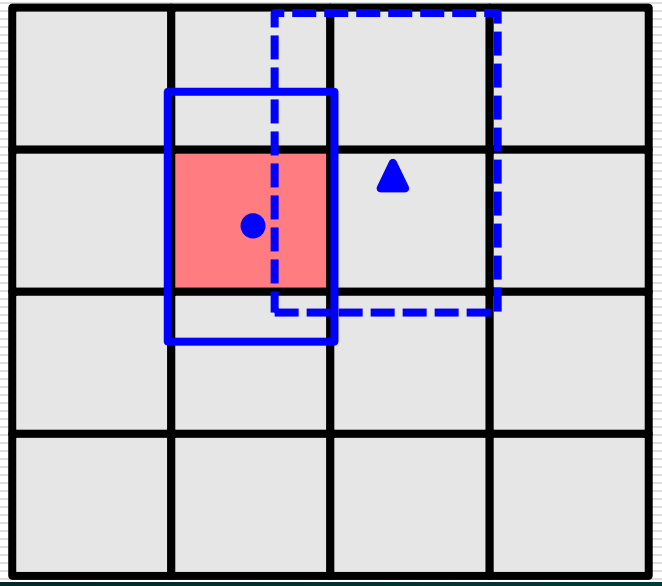
k anchors per grid

Total # of anchors = w*h*k

Anchor

- **Fixed-shape bboxes** are given, and modified with regression





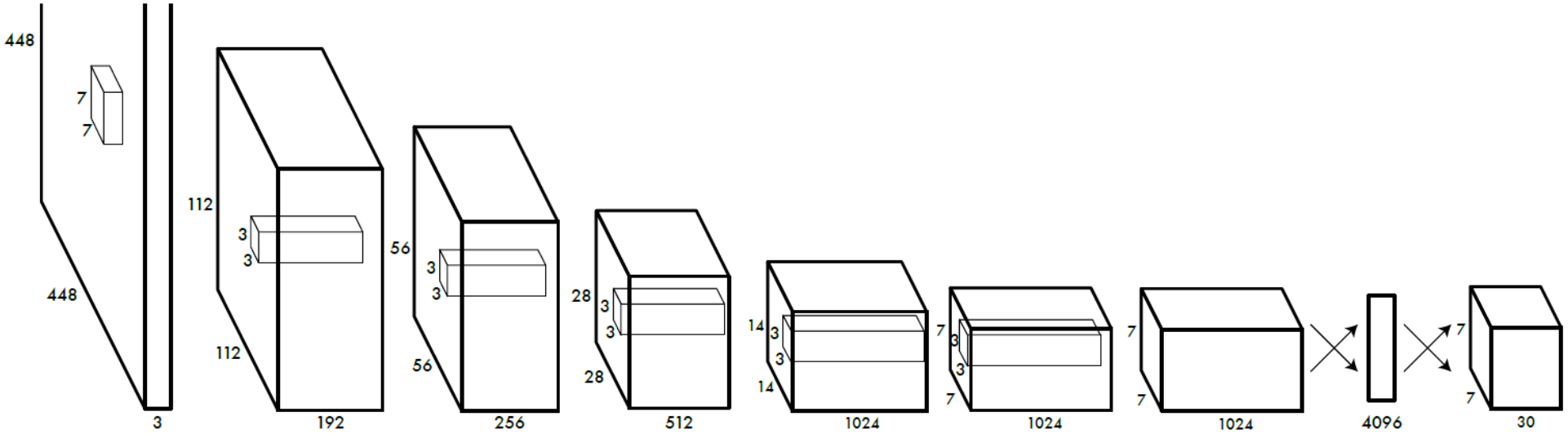
(1, 1, 3, 6) ➔
(1+dx, 1+dy, 3*dw, 6*dh)

Summary of R-CNN Family

- **Feature map sharing** improves the speed and eliminates redundant computation
- **Anchor-base prediction** makes accuracy better
- **Two-stage** framework is not perfect enough
  - Cropping issue ➔ **slow, cannot end-to-end training**

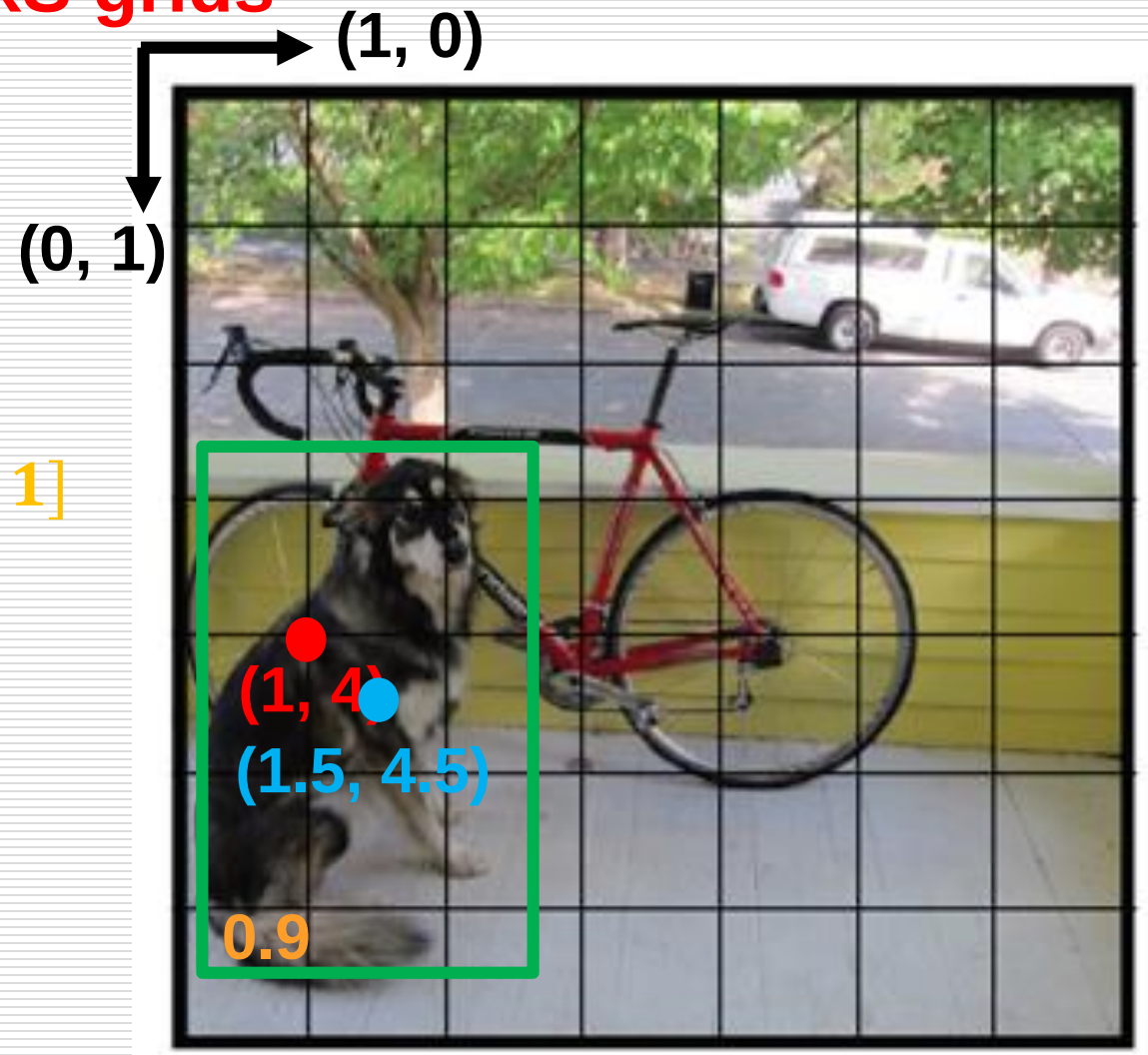
YOLO

- You Only Look Once[4]
- One-stage (No RoI)



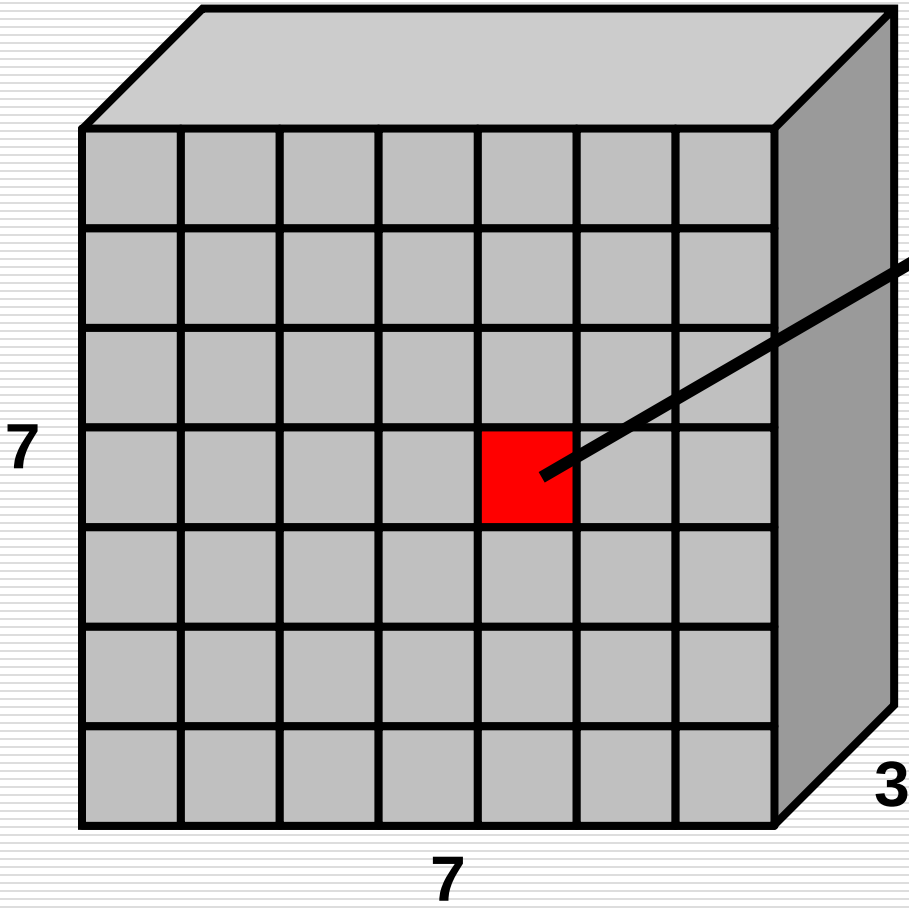
Main Idea

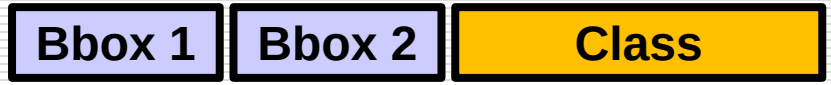
- Divide image into **SxS grids**
- **Coordinate**
- Bbox
  - **Offset: x, y ∈ [0, 1]**
  - **Size: w, h ∈ [0, 1]**
  - **Confidence: c ∈ [0, 1]**



Output Format

- YOLO has **k different bboxes in each grid**





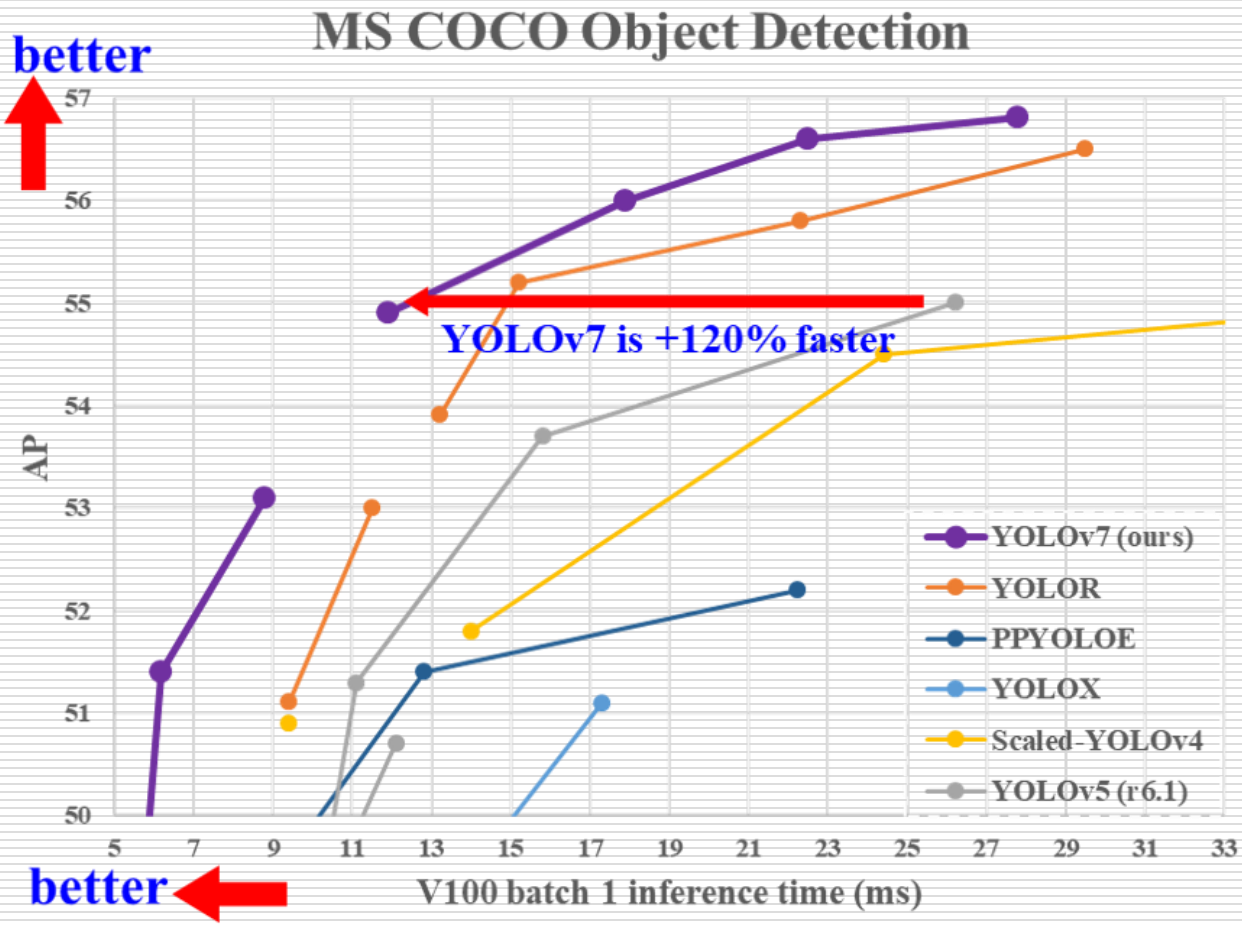
Bbox: (x, y, w, h, c)

Class: 20 classes

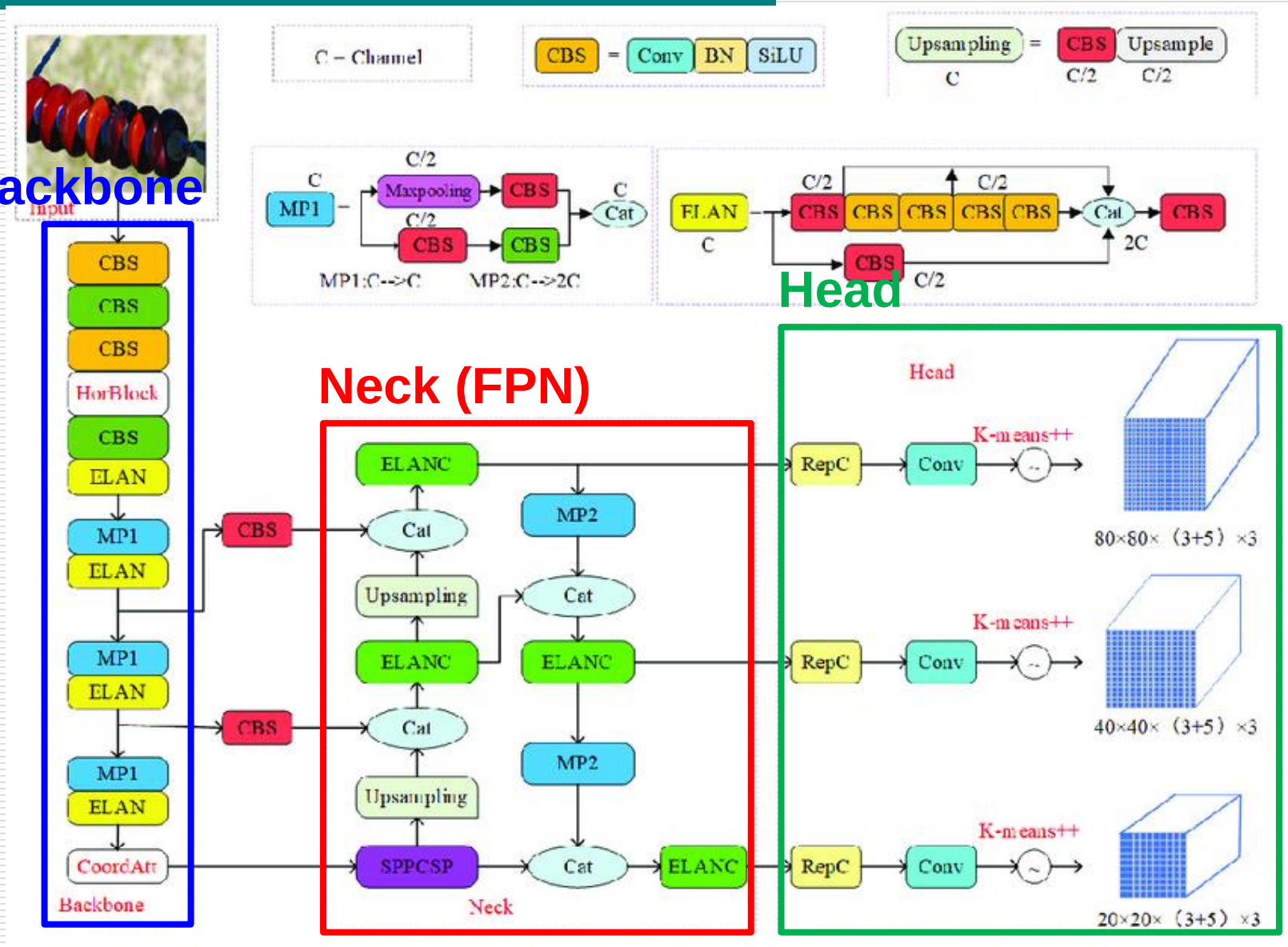
30 = 2*5 + 20

YOLOv7

- **YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors[5]**

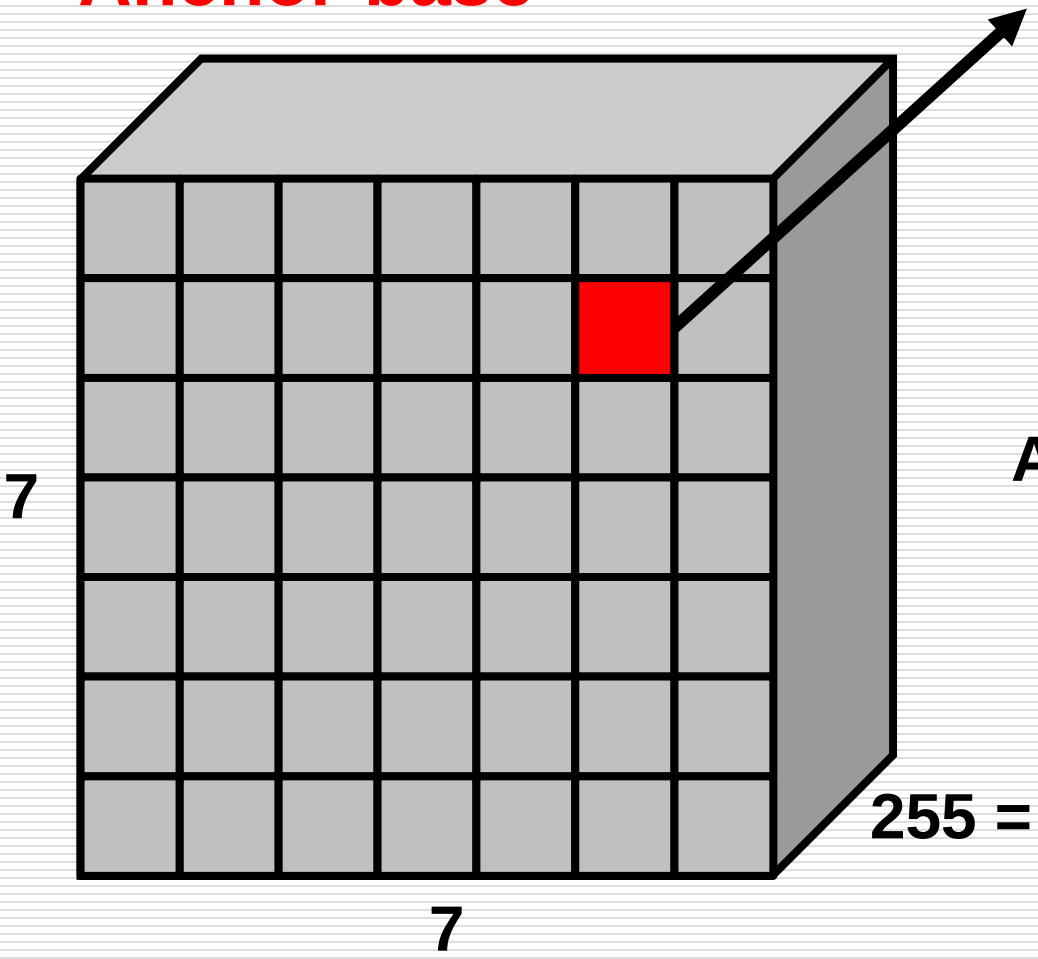


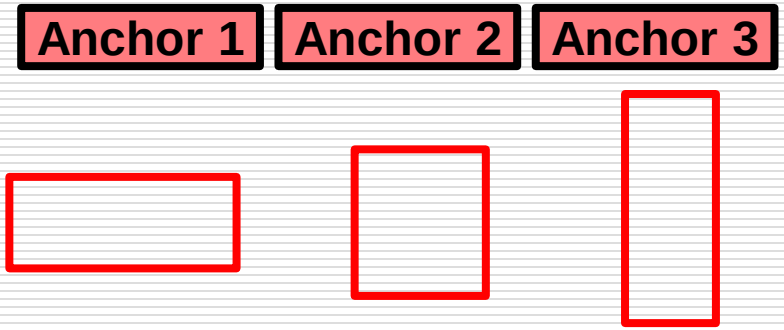
Architecture



Output Format

- Multiple boxes per grid
- **Anchor-base**





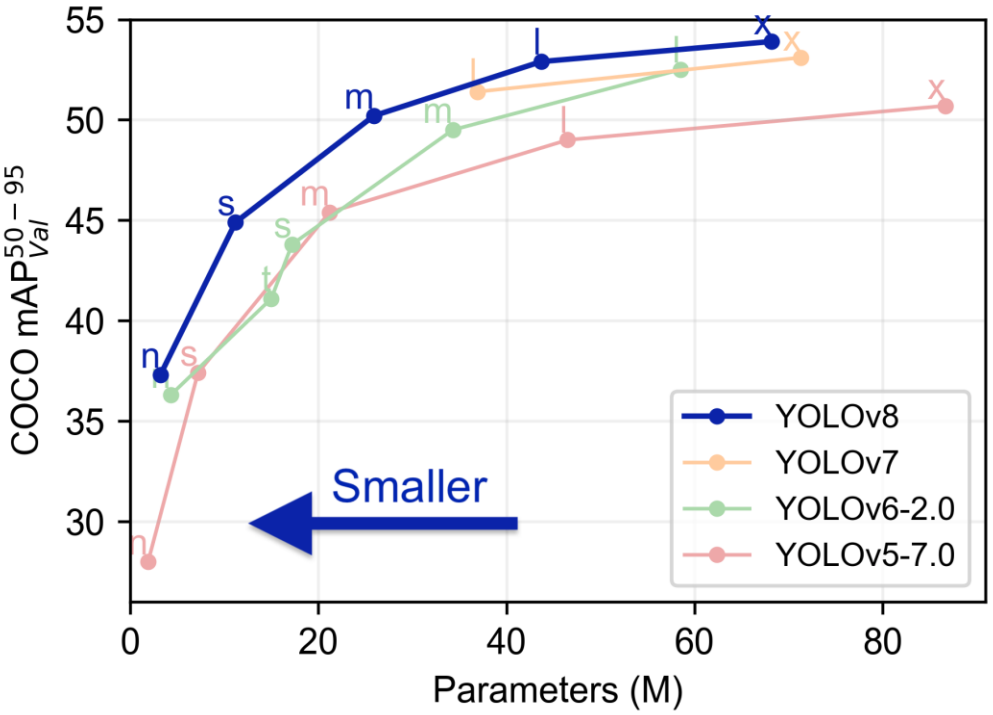
N-dim vector

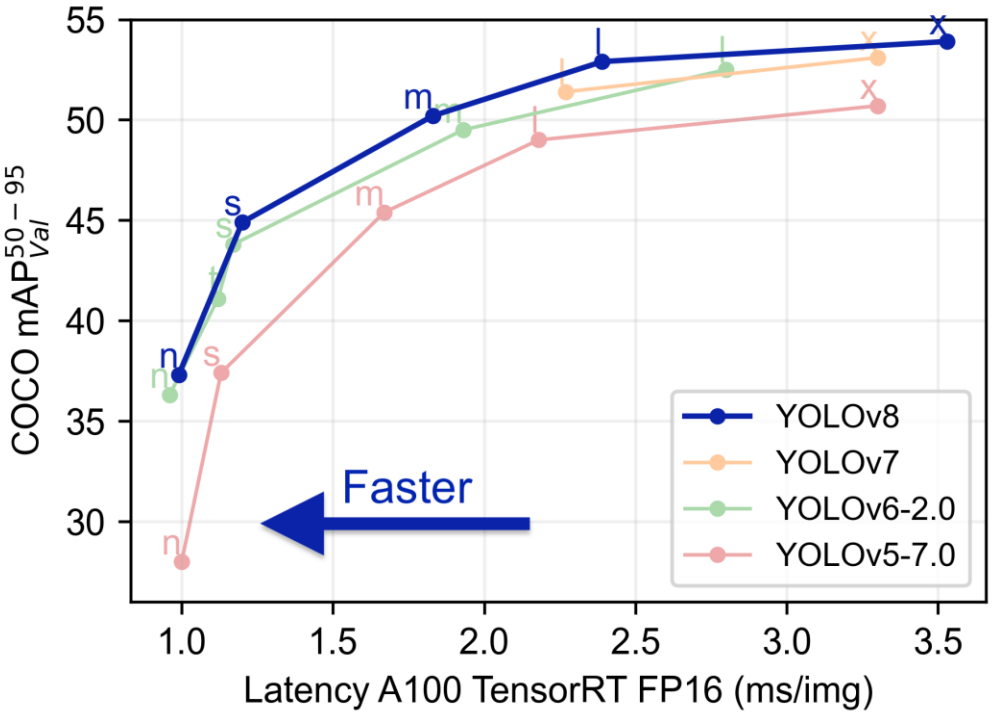
Anchor: (x, y, w, h, conf, …)

255 = 3 * (4 + 1 + 80)

YOLOv8

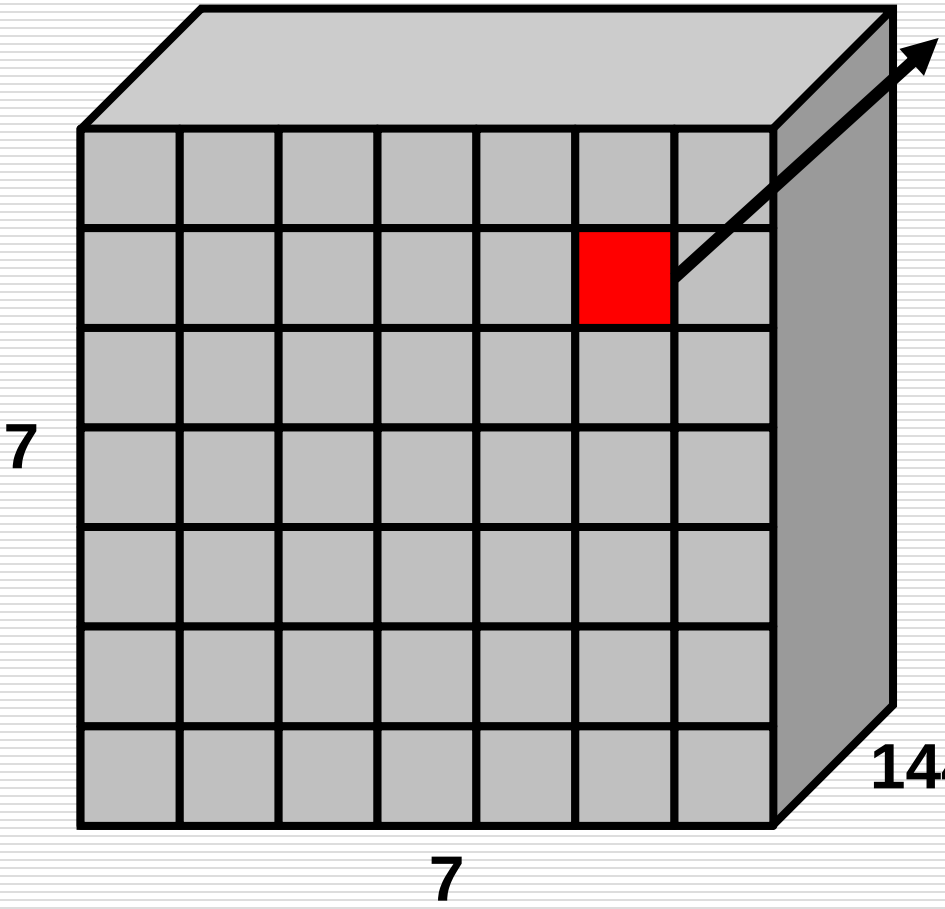
- **Ultralytics YOLOv8[6]**





Output Format

- Single box per grid
- **Anchor-free**



Class: 80 classes

Bboxes: (x, y, w, h)*16

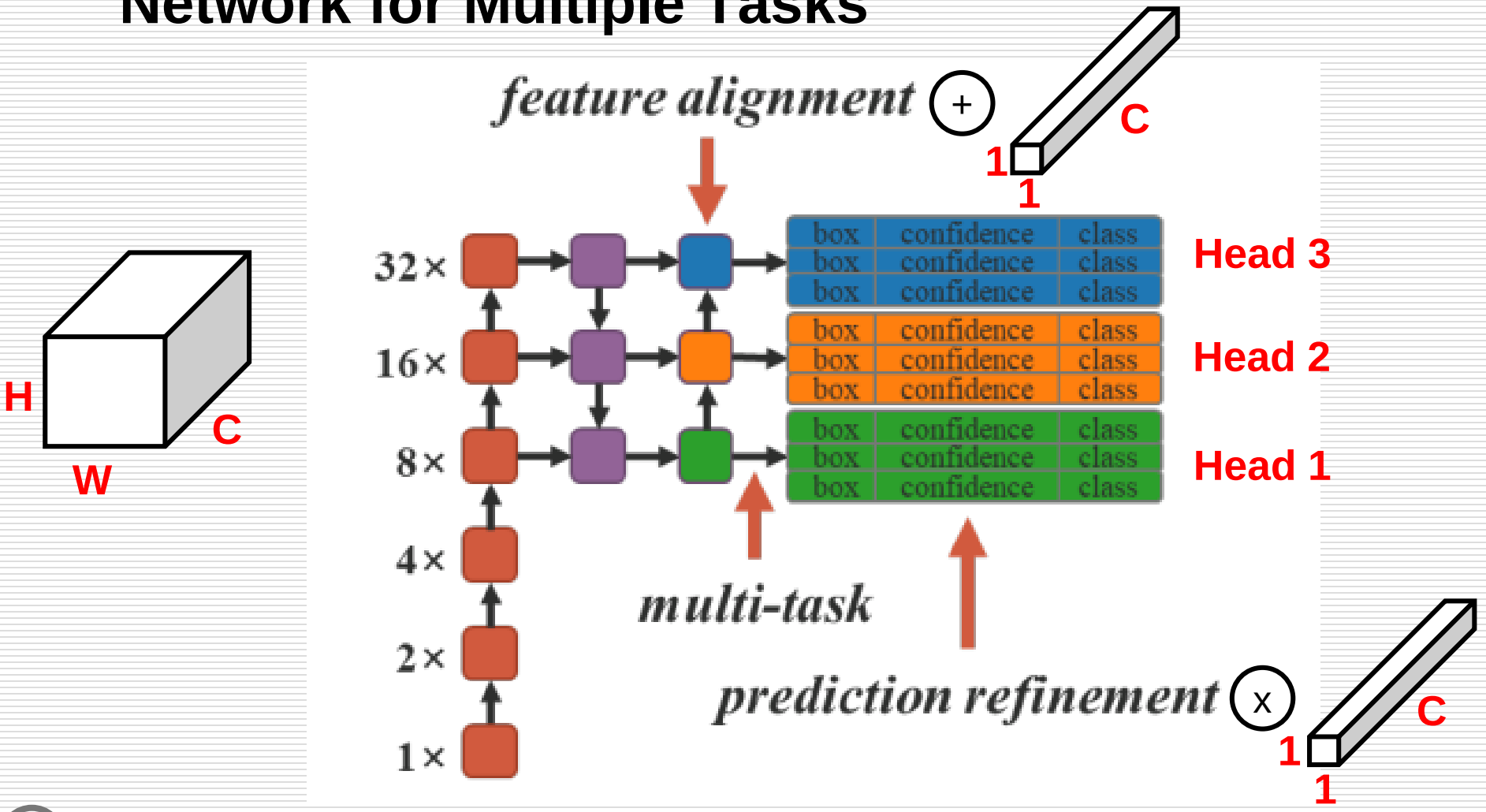
$$Pred = 1*a_0*w_0 + 2*a_1*w_1 + \dots + 16*a_{15}*w_{15}$$

$$w_0 + w_1 + \dots w_{15} = 1$$

144 = 4 * 16 + 80

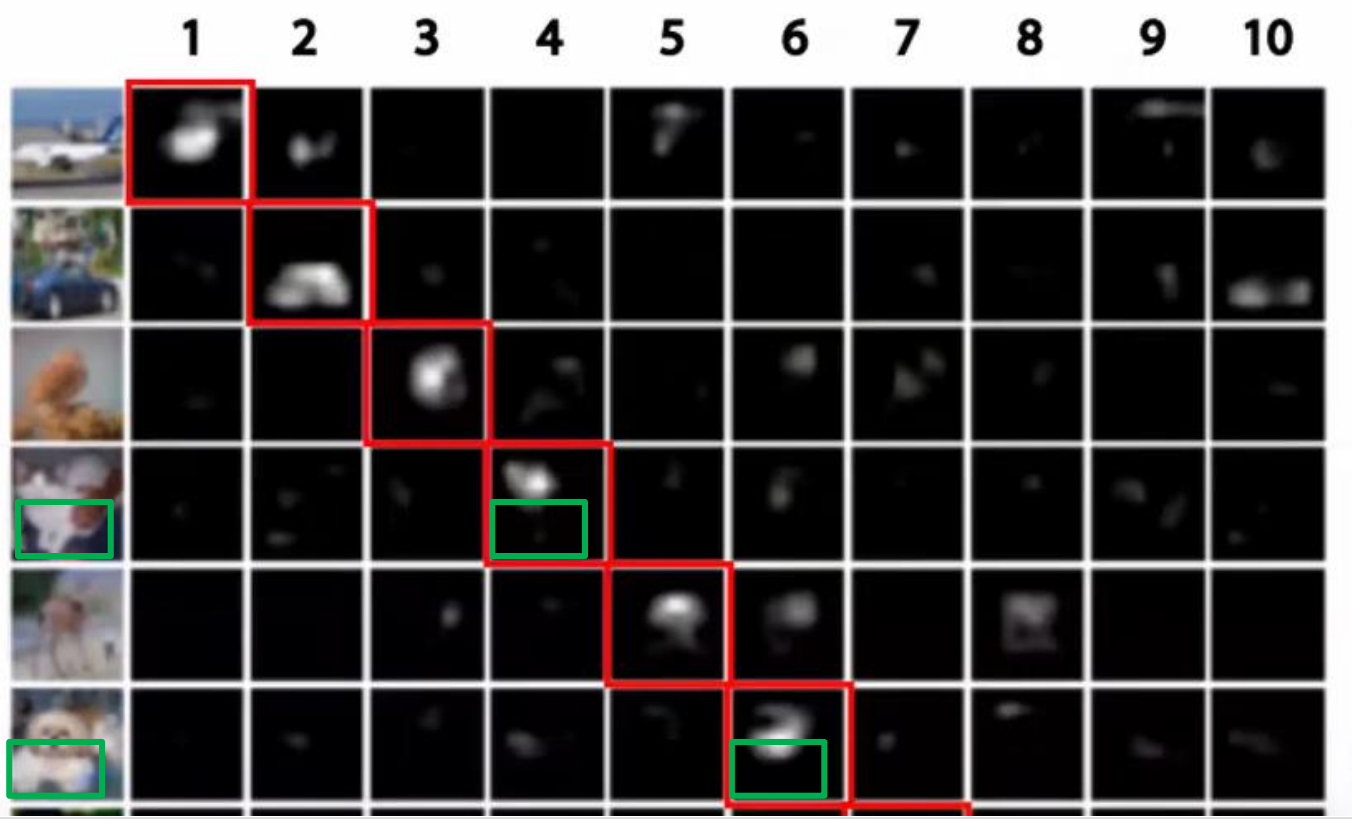
YOLOR

- You Only Learn One Representation: Unified Network for Multiple Tasks



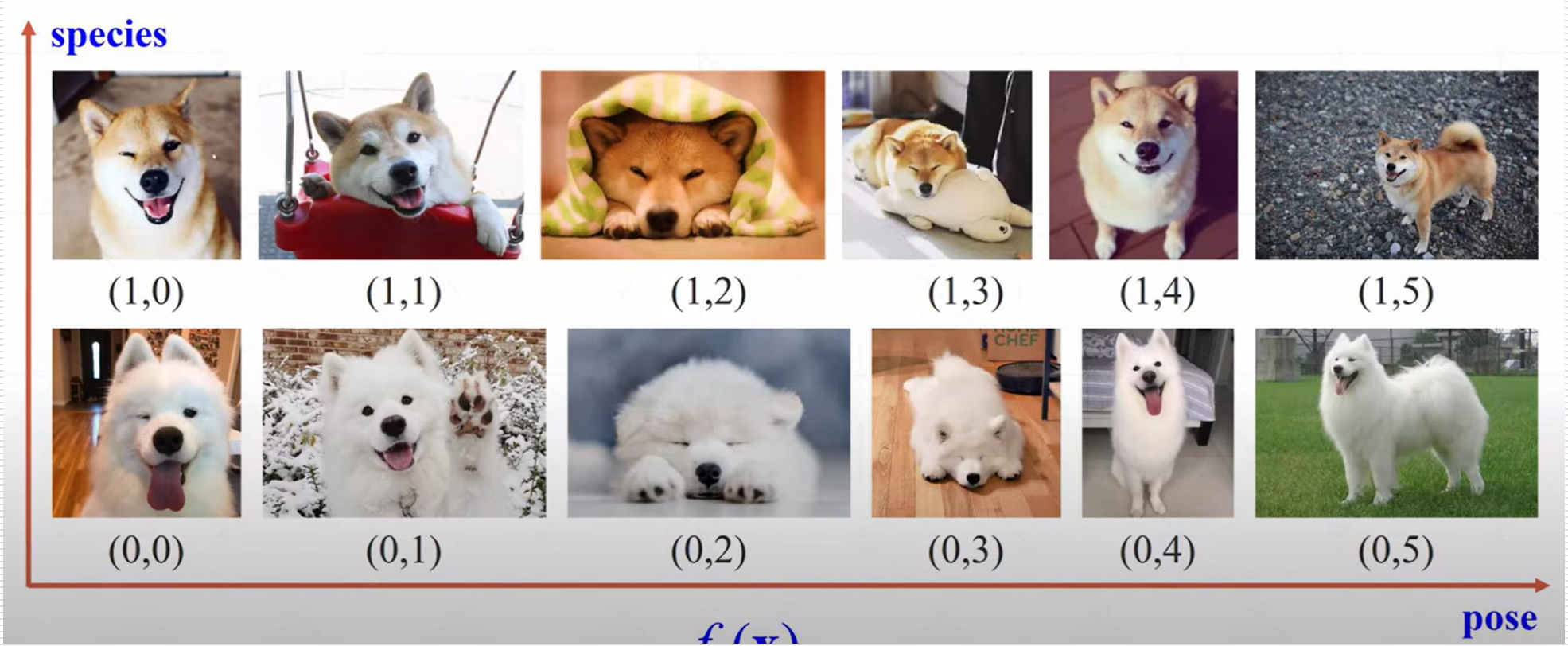
YOLOR – Motivation

- Activation map



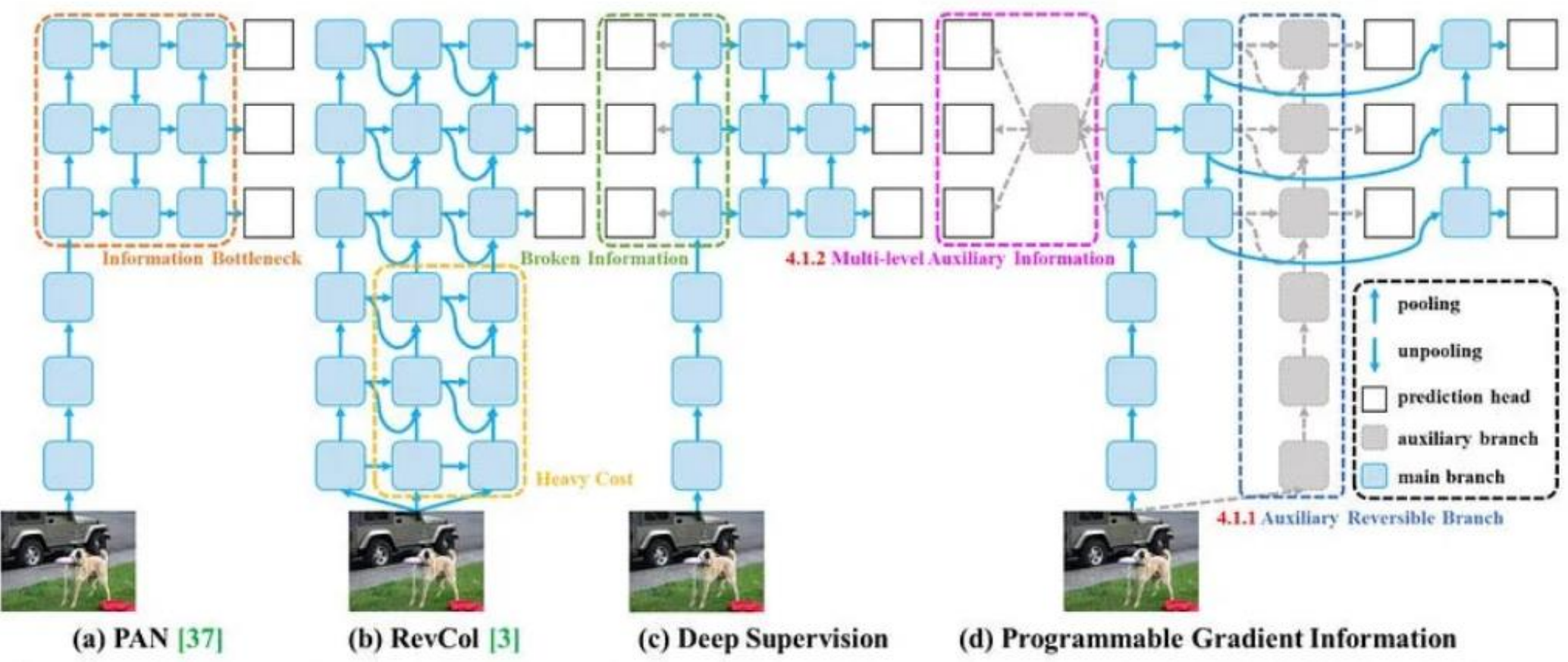
YOLOR – Concept

- Representation vector space



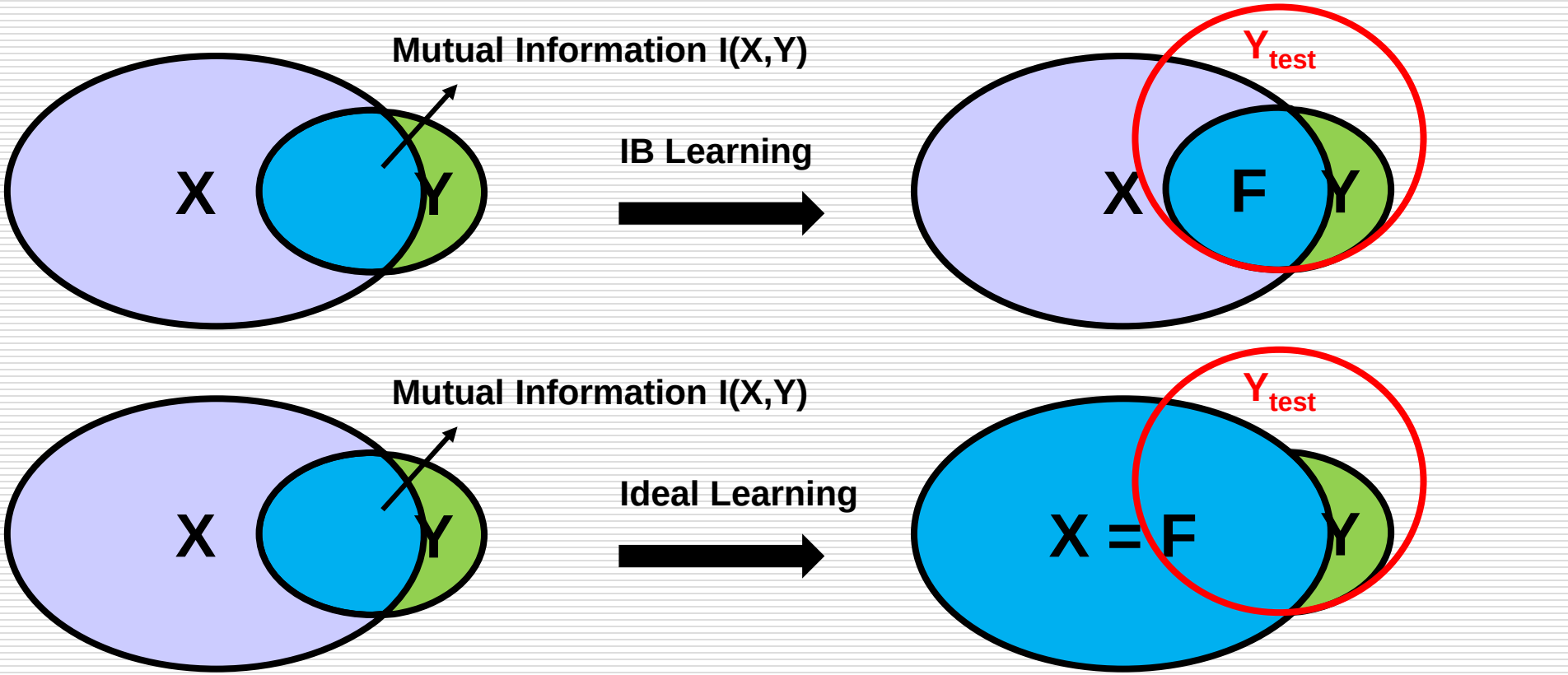
YOLOv9

- **YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information**

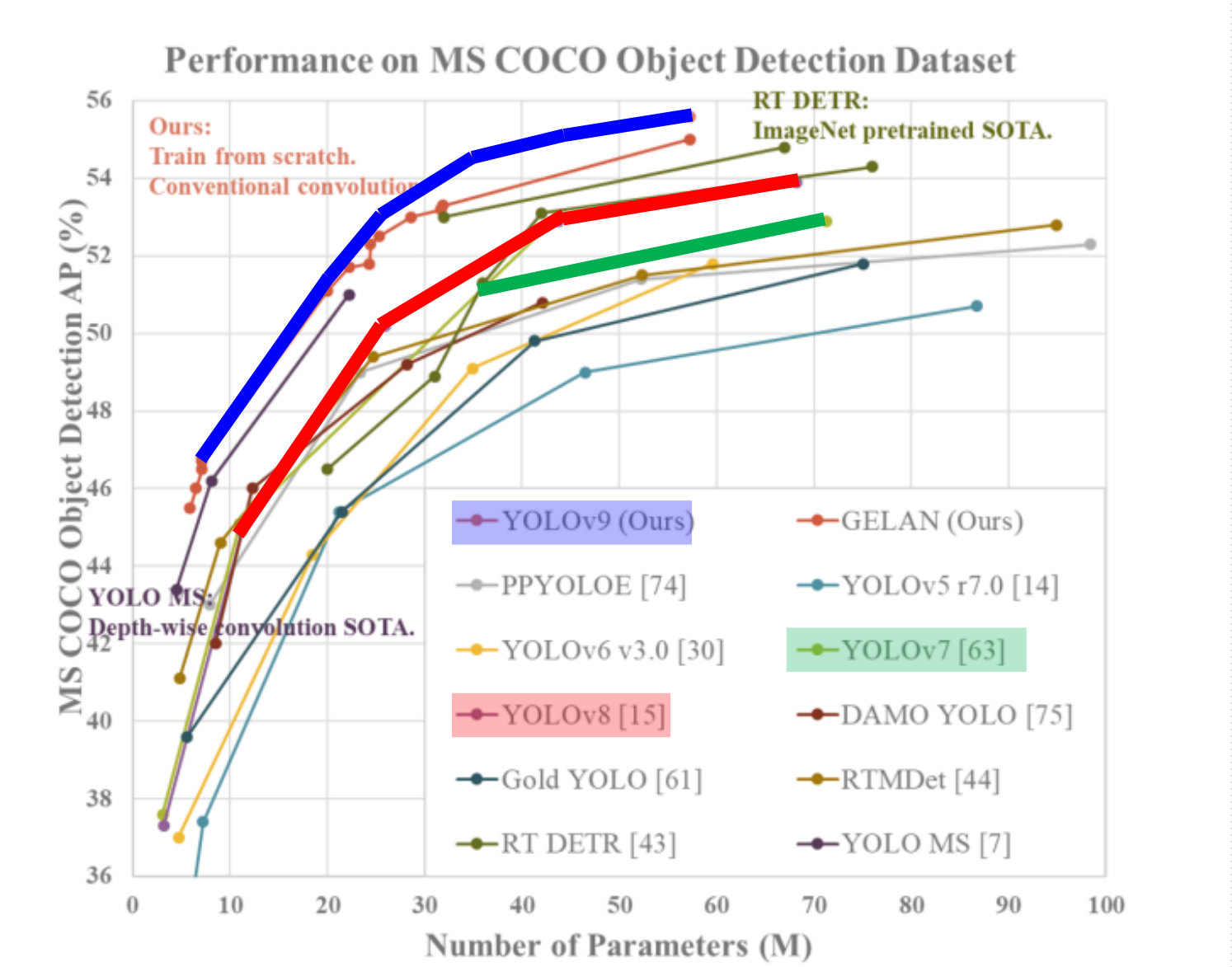


YOLOv9 – Motivation

- **Information Bottleneck** (IB) and **unreliable gradients** from incomplete information



Performance

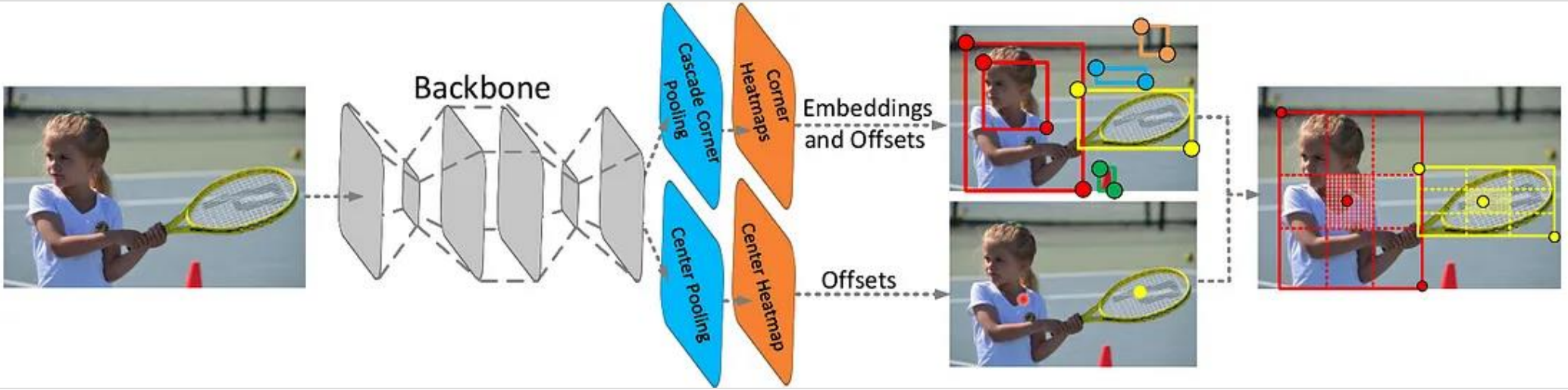


Summary of YOLO Family

- **One-stage framework** increase inference efficiency a lot
- **Regression outputs** have several formats
- **Complicated modules** in the model
- Training framework optimization is the trend

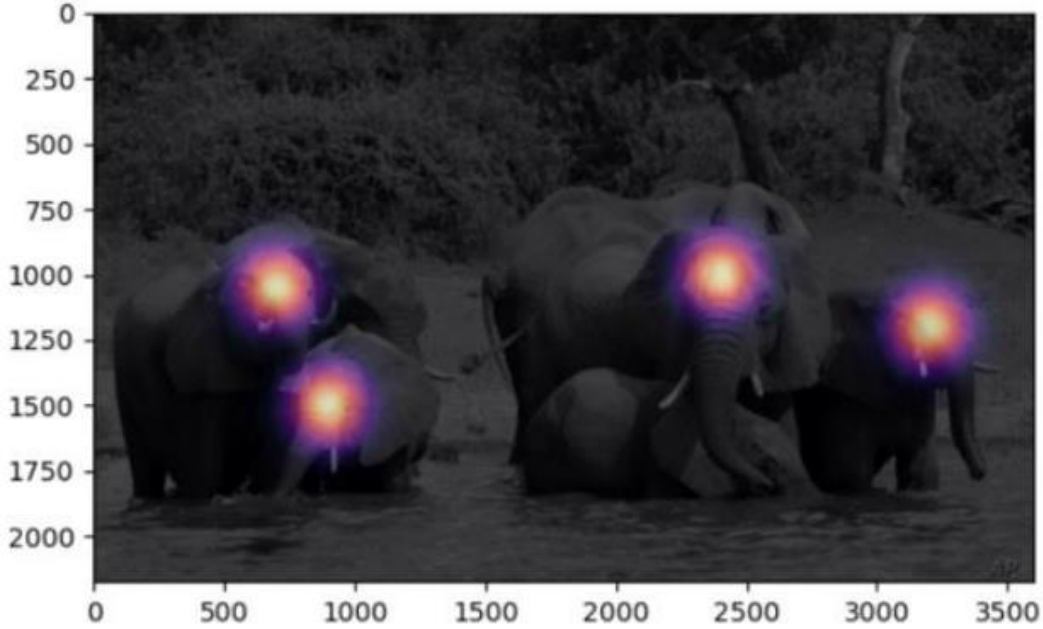
CenterNet

- **CenterNet: Keypoint Triplets for Object Detection[7]**
- One-stage
- **Heatmap-base**
- **No-NMS**



Main Idea (1/2)

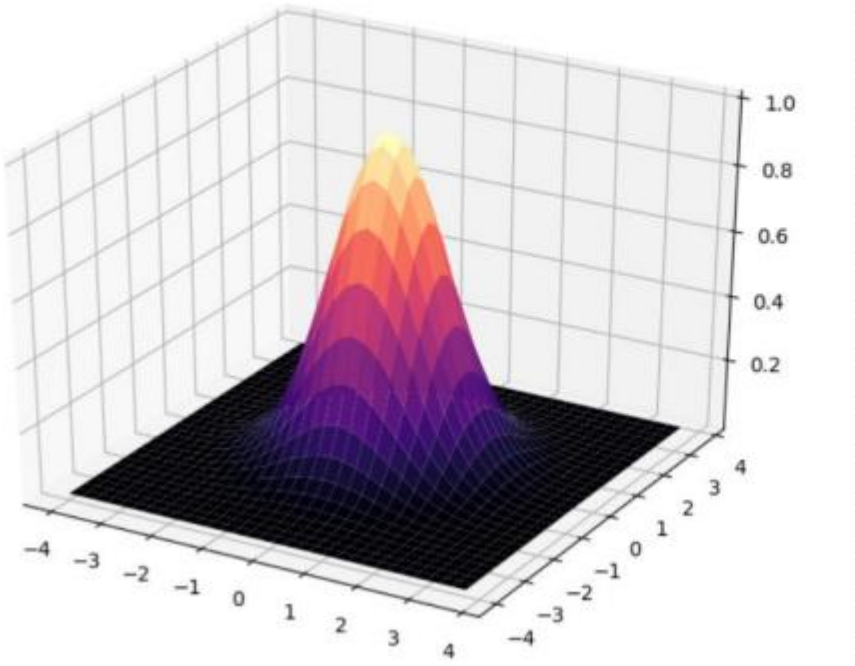
- Predict the **central grids** of objects



Main Idea (2/2)

- Use **gaussian function** to present heatmap

$$Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$$



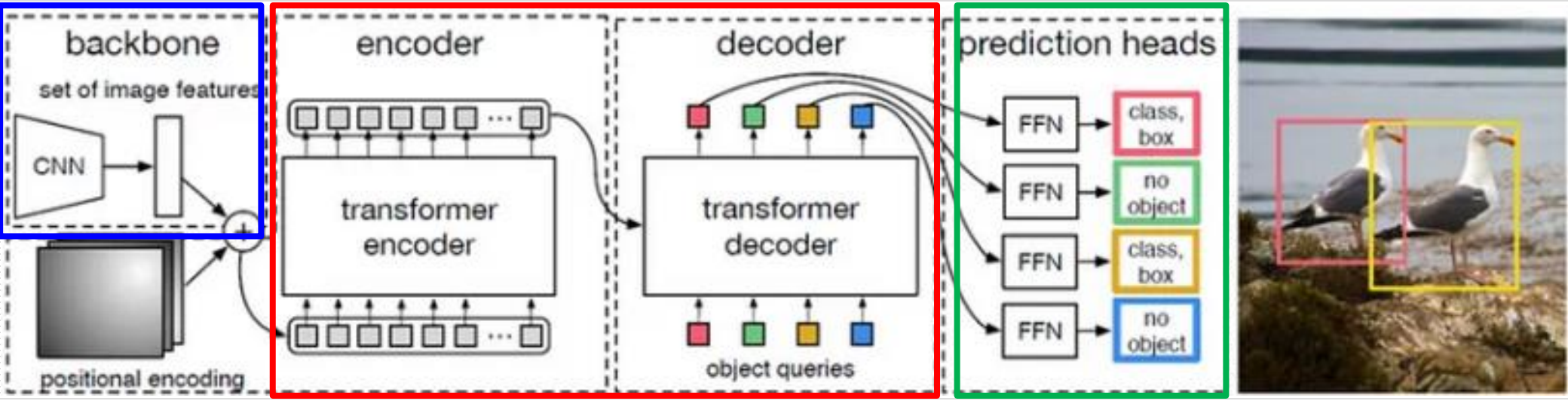
Summary of CenterNet

- **Heatmap** can be seen as **confidence**
- Predict **offset(dx, dy)** and **size(w, h)**
- **NMS is not needed**
- **Dense objects with same class** will be hazard

DETR

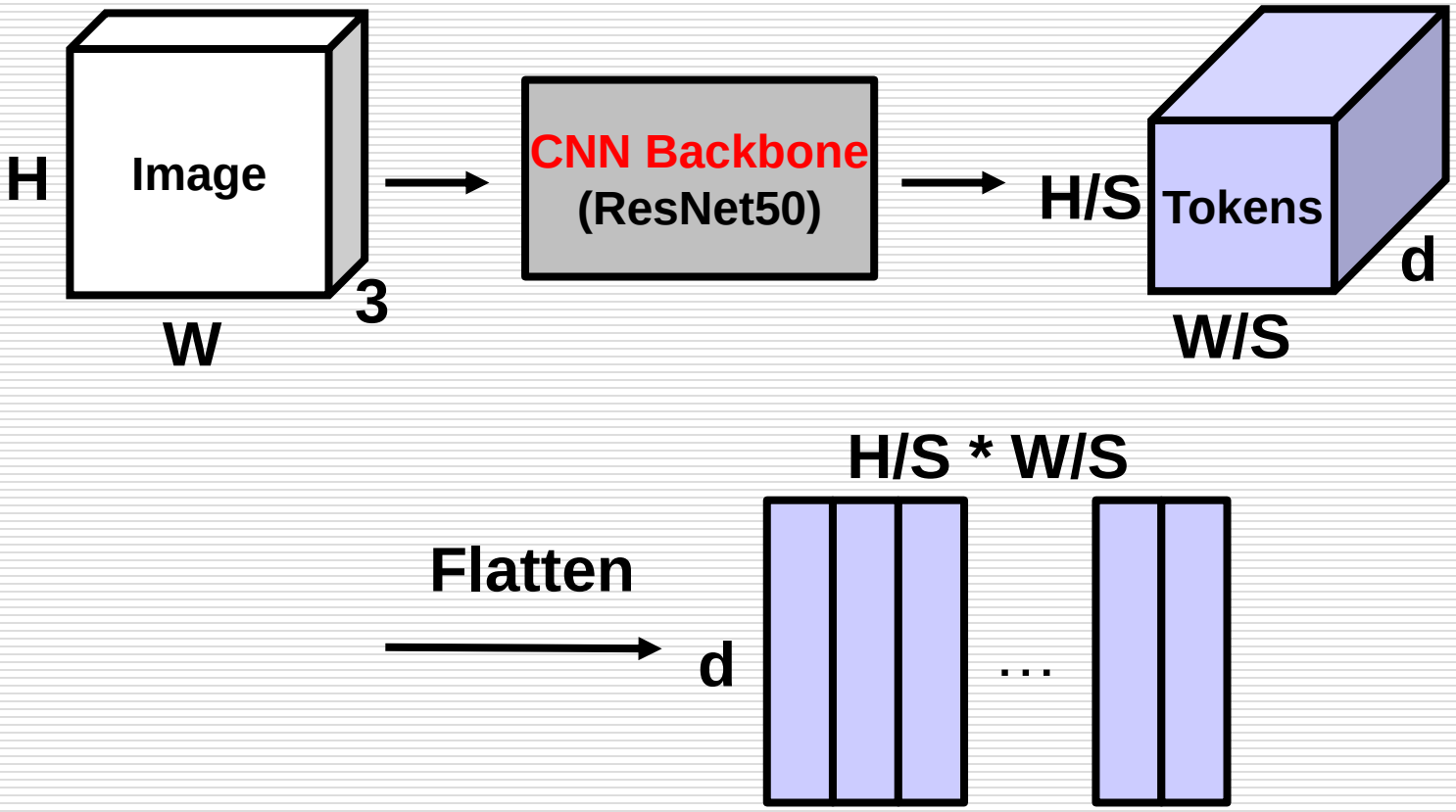
- **DETR: End-to-End Object Detection with Transformers[8]**
- One-stage
- **Transformer-base**

CNN Backbone **Encoder-Decoder** **Head**



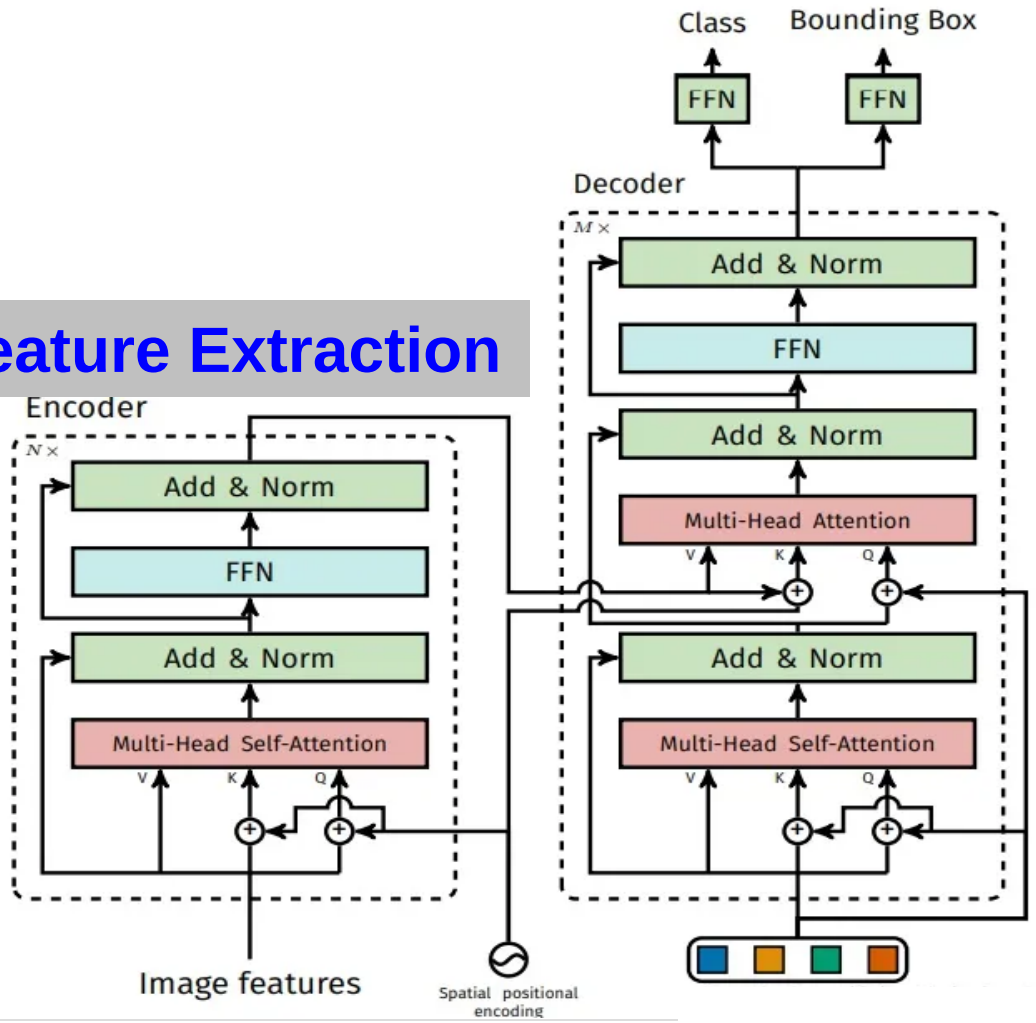
Step 1: Tokenization

- **CNN backbone** is used to **tokenize an image** into several tokens



Step 2: Feature Extraction

- NLP-like **encoder-decoder** framework



a. Global Feature Extraction

d. Decode Anchors

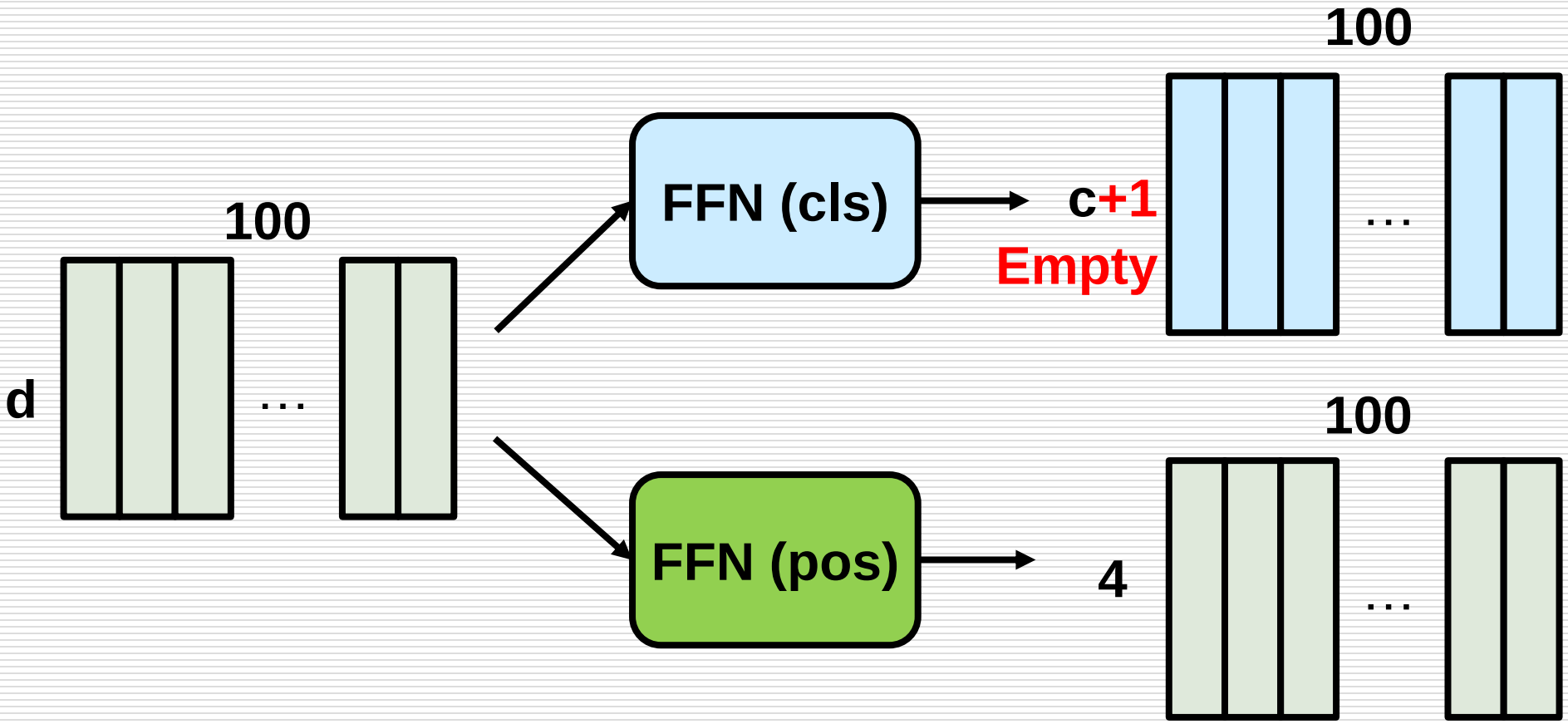
c. Modify Anchors

b. Initialize Anchors

Object Queries (learnable params)

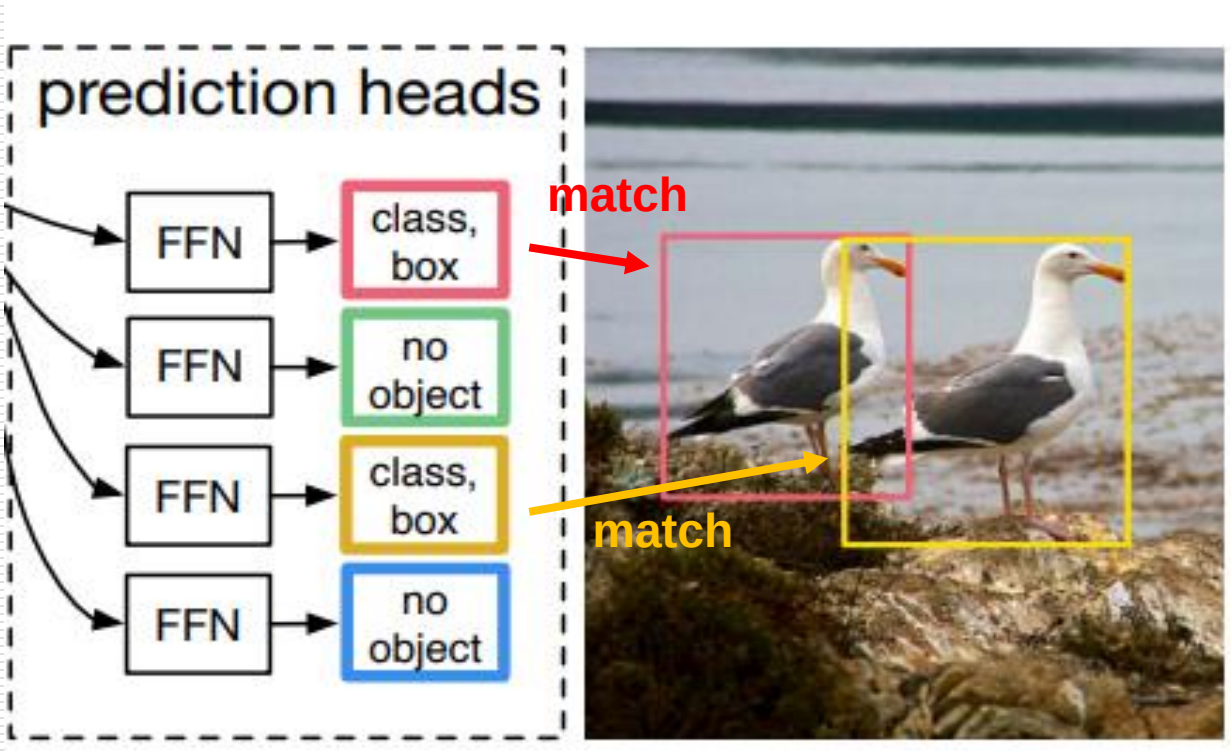
Step 3: Bboxes Prediction

- FFN completes **classification** **and** **localization**



Step 4: Inference

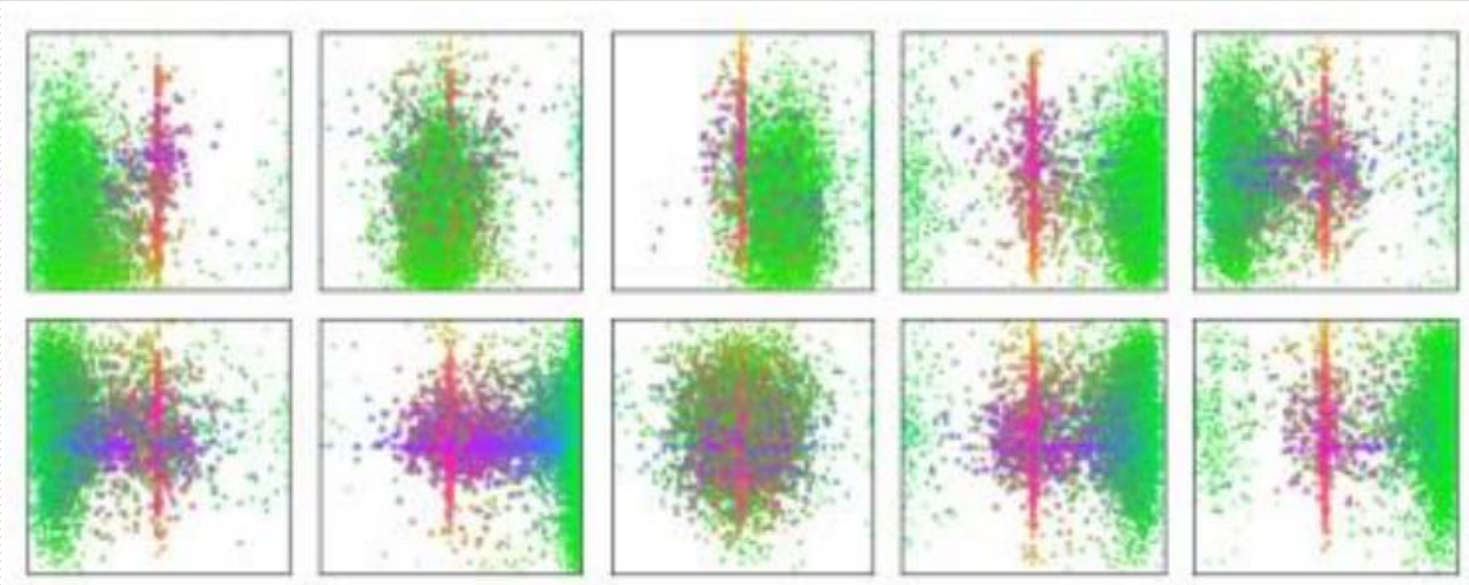
- Objects and predictions are matched 1-to-1 by **Hungarian algorithm during training phase**
  - **n** indicates the number of objects
  - $\mathbf{C_{i,j}}$ indicates the **cost** of **Box(i)** matching with **Object(j)**



$$\begin{array}{c|cccc} & O_0 & O_1 & \cdots & O_n \\ P_0 & C_{0,0} & C_{0,1} & \cdots & C_{0,n} \\ P_1 & C_{1,0} & C_{1,1} & \cdots & C_{1,n} \\ \vdots & \vdots & \vdots & & \vdots \\ P_{99} & C_{99,0} & C_{99,1} & \cdots & C_{99,n} \end{array}$$

Visualization

- Each object query captures objects in **specific position** in images
  - 10 of 100 queries are shown below
  - **Points** indicate the centers of predictions
  - **Colors** indicate the size of bboxes
  - **# of points** indicate the number of images in the dataset



Summary of DETR

- A **COOL** detection model with Transformer
- Bad ability to detect **small-size objects**
- Difficult implementation because of **massive computation**

Outline

- Introduction
- Data preparation
- Models
- Training and Testing
  - Target Matching
  - Loss Functions
  - Post-processing
- Discussion
- Homework
- Reference

Training and Testing

- In the **training phase**
  - # of predictions are always **much larger than** # of targets
  - **Classification and localization** should be completed

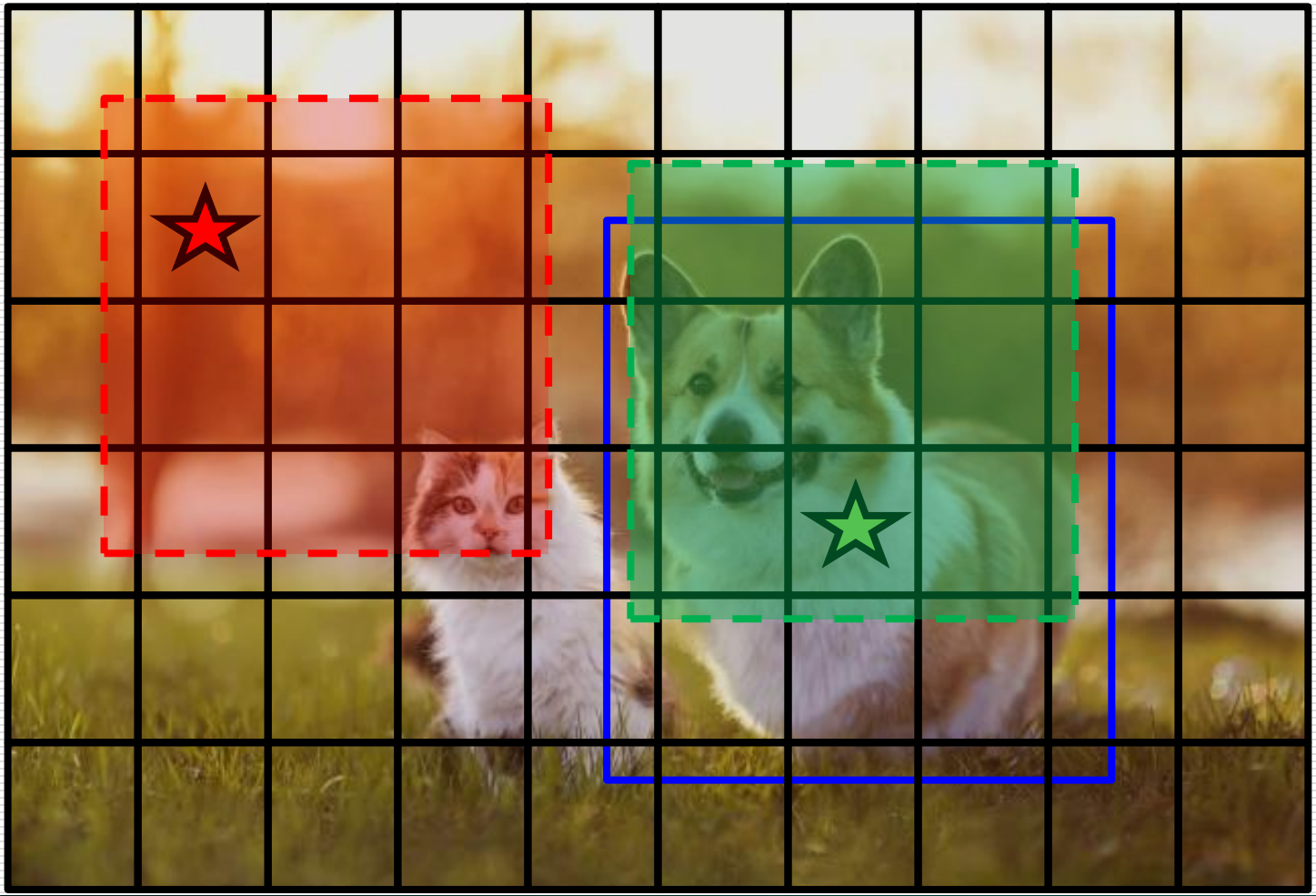
So, how to train?

- In the testing phase
  - Predictions **may capture the same object**
  - Predictions **may capture nothing**

So, how to test?

Target Matching (1/2)

- During training, not all the predictions are considered

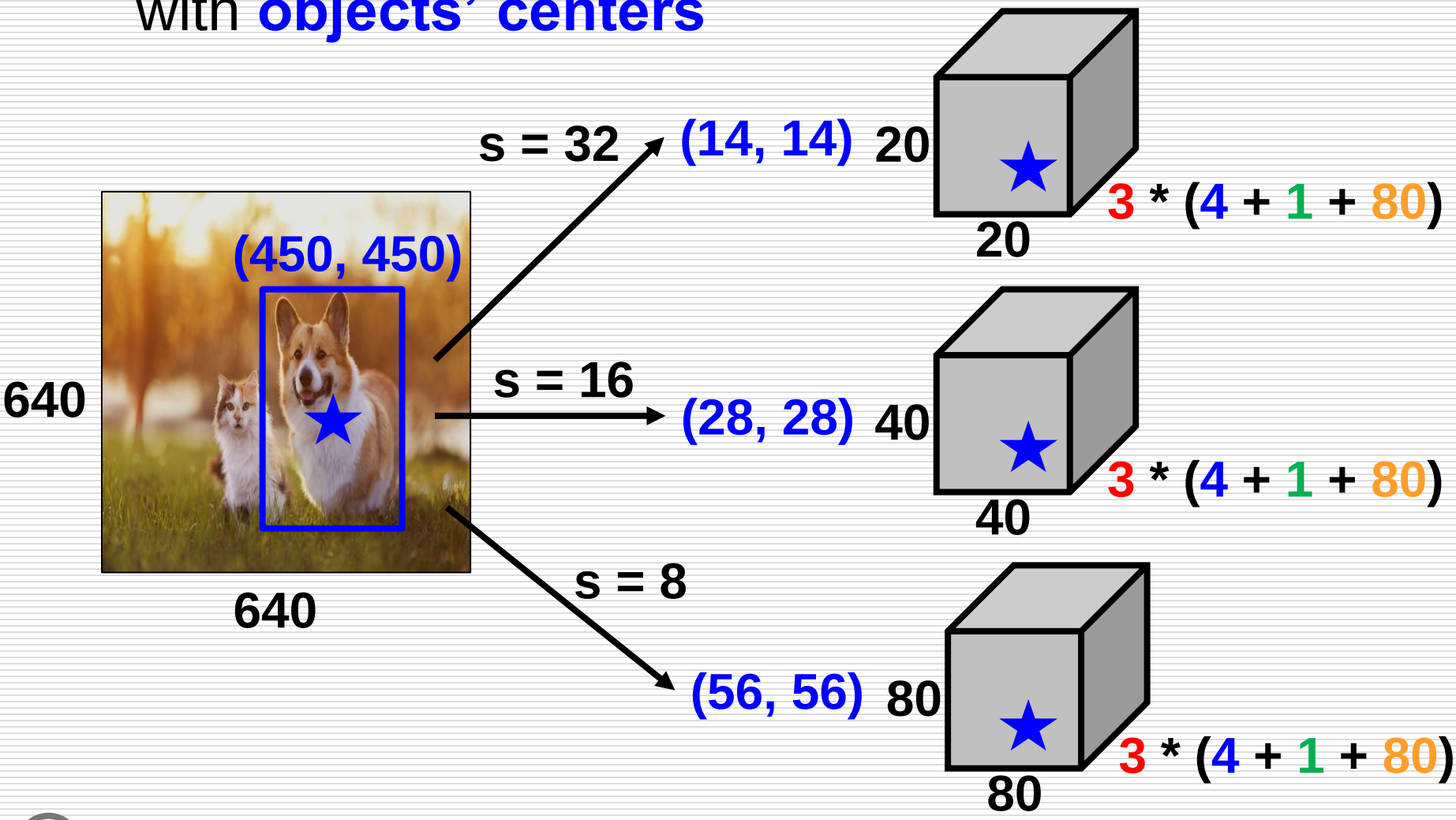


Hard to fit target

Easy to fit target

Target Matching (2/2)

- YOLO's method: reserve predictions overlapping with **objects' centers**

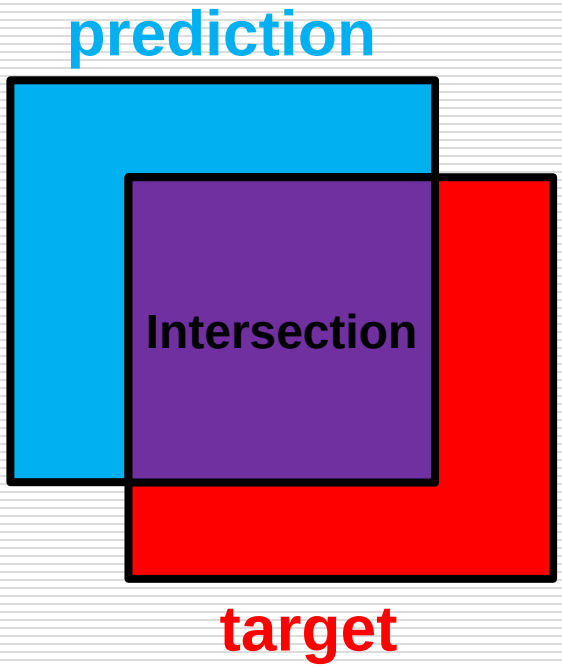


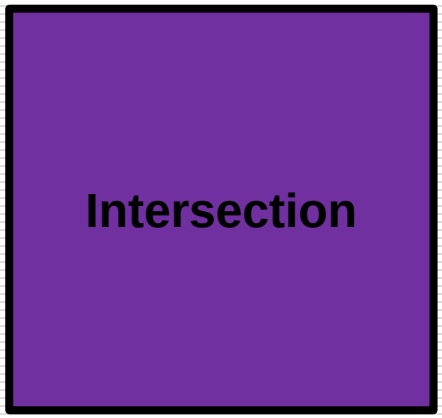
Loss Function (1/3)

- Box loss

  bbox: (x, y, w, h, conf, cls)

  – $\text{Loss}_{box} = 1. - \text{box\_iou}(\text{box}_{pred}, \text{box}_{target})$





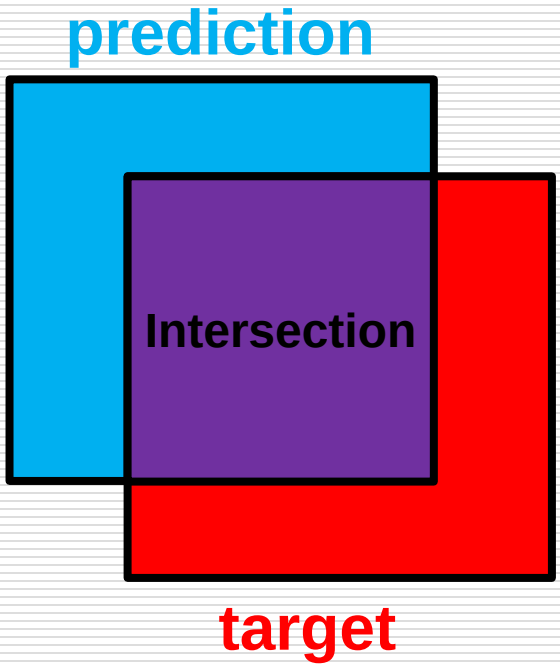
Prefect prediction!
IoU = 1. ➔ $\text{Loss}_{box} = 0.$

Loss Function (2/3)

- Confidence loss

bbox: (x, y, w, h, conf, cls)

  - **How much confidence you should have** with the current prediction to the target?
  - **$\text{Loss}_{\text{conf}}$ = cross_entropy(conf, box_iou(box_{pred}, $\text{box}_{\text{target}}$))**



Loss Function (3/3)

- Class loss

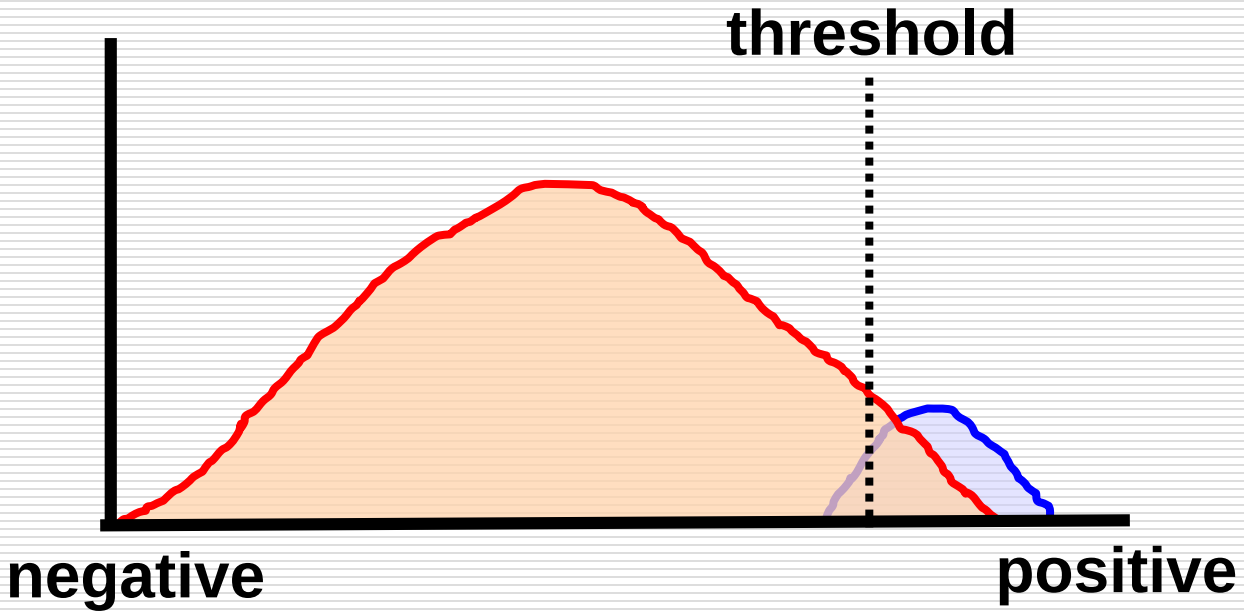
  bbox: (x, y, w, h, conf, cls)

  - $\mathbf{Loss_{cls} = cross\_entropy(cls, one\_hot(ID_{class}))}$

- Total loss
  - $\mathbf{Loss_{total} = a * Loss_{box} + b * Loss_{cls} + c * Loss_{conf}}$

Advanced Loss Function

- In some situations, the **imbalanced sample distribution** makes training results useless
  - Ex. Binary classification task – **negative:positive = 99:1**
  - FP usually more than FN ➔ **Hard to pull FN to positive**
  - Always predict negative, get 99% accuracy ➔ **Useless**



Weighted Loss

- **Weighted binary cross entropy(wbce)** is common used

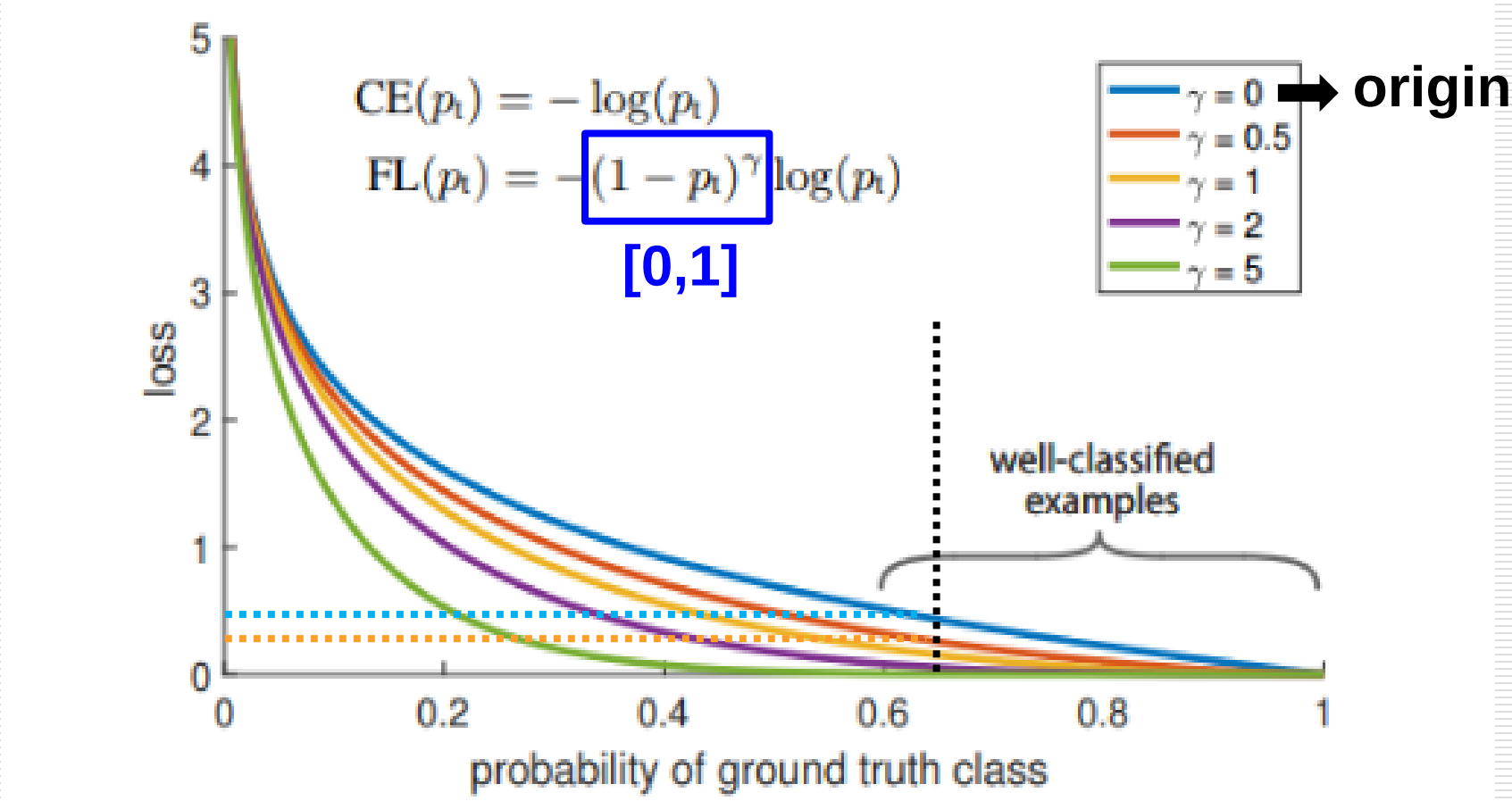
$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} \underbrace{y_i \cdot log(p(y_i))}_{W_{pos}} + \underbrace{(1-y_i) \cdot log(1-p(y_i))}_{W_{neg}}$$

Positive Loss

Negative Loss

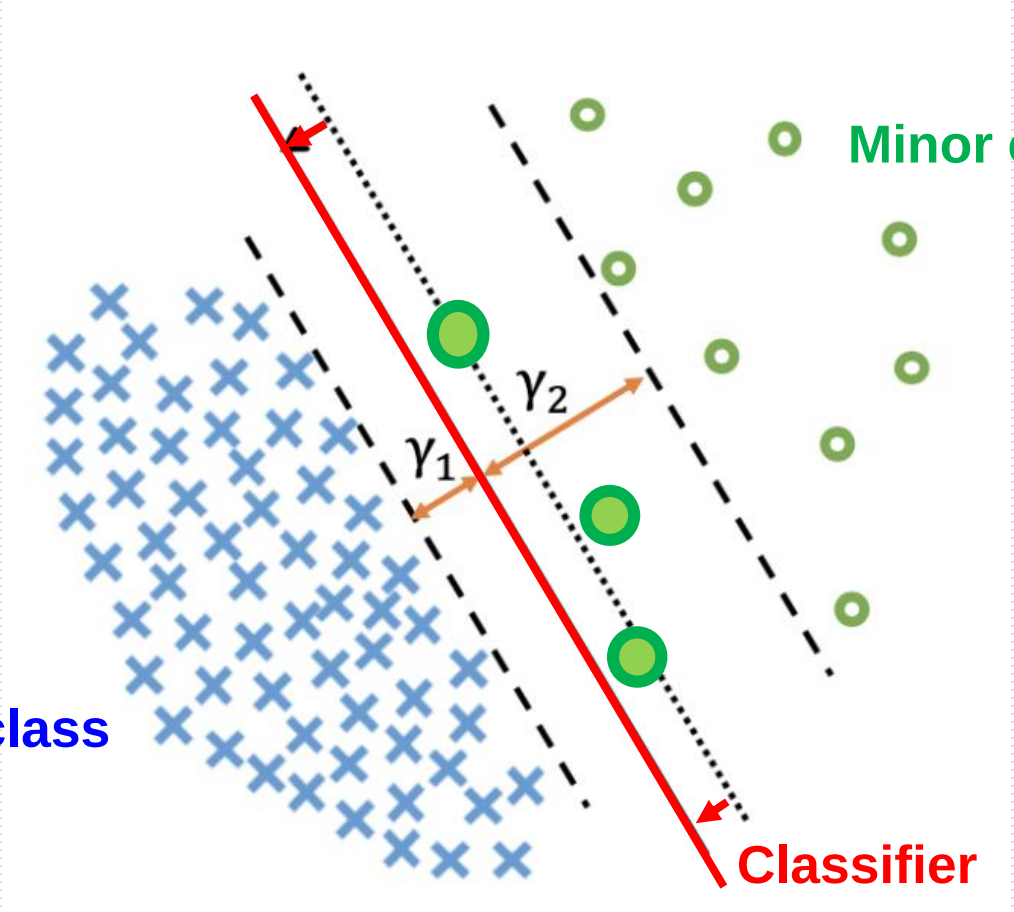
Focal Loss

- Firstly used in RetinaNet[9]
- **Suppress the loss which is good enough**



LDAM Loss (1/2)

- **LDAM: Label-Distribution-Aware Margin**[10]
- Push margin **far from minor class**



LDAM Loss (2/2)

- Encourage $pred_{minor}$ more extremely closed to, even beyond target ➔ **loss of minor class become larger**
- Opposite concept to focal loss

$$\gamma_j = \frac{C}{n_j^{1/4}} \quad \Rightarrow \quad \gamma_{\text{minor}} > \gamma_{\text{major}}$$

$$\boldsymbol{loss = BCE\left(\begin{bmatrix} pred_{minor} \\ pred_{major} \end{bmatrix} - \begin{bmatrix} \gamma_{minor} \\ \gamma_{major} \end{bmatrix}, \begin{bmatrix} label_{minor} \\ label_{major} \end{bmatrix}\right)}$$

Postprocessing

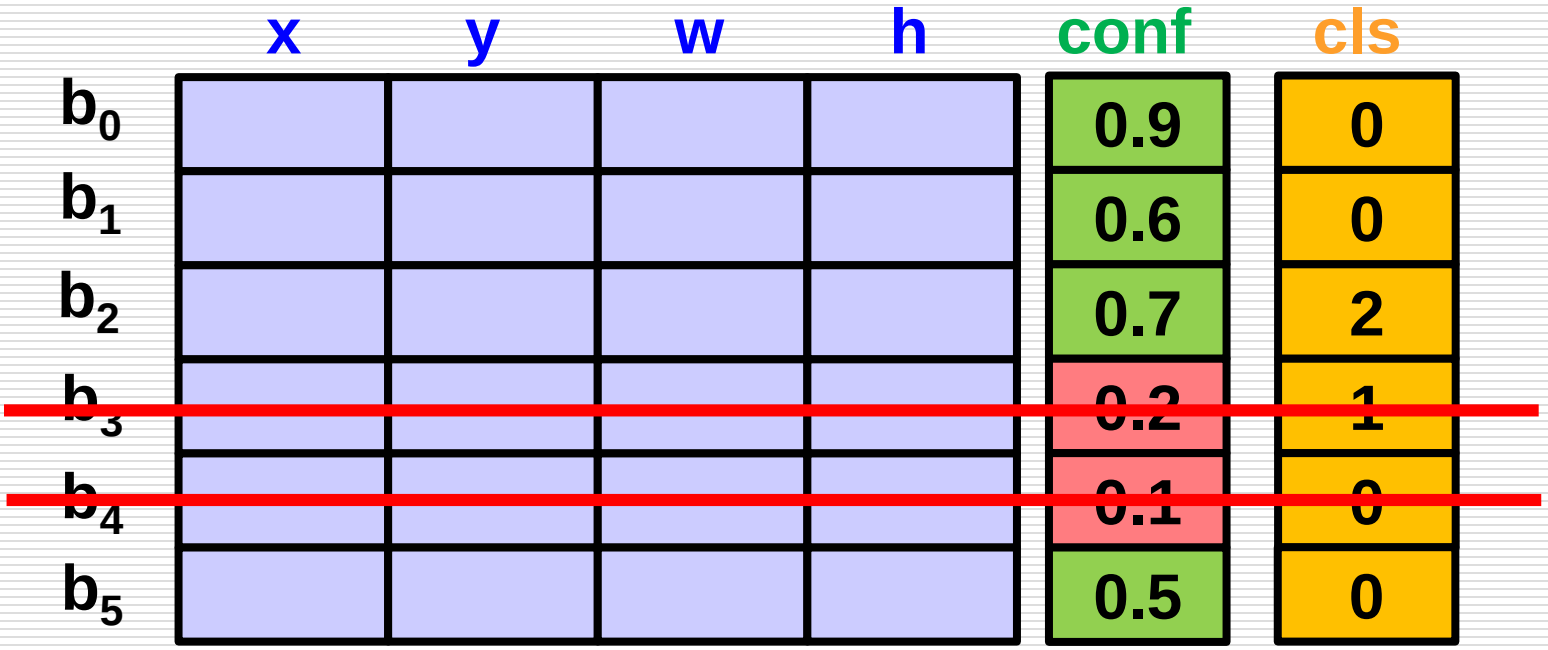
- Eliminate redundant bboxes
  - **Confidence threshold**
  - **Non-max suppression(NMS)**
- Draw bboxes on images
- **Example.**

	x	y	w	h	conf	cls
b_0					0.9	0
b_1					0.6	0
b_2					0.7	2
b_3					0.2	1
b_4					0.1	0
b_5					0.5	0

Step 1: Threshold

- Eliminate the bboxes whose confidence scores are **less than confidence threshold**

conf_threshold = 0.3



Step 2: NMS - Sort Confidence

- Before

	x	y	w	h	conf	cls
b_0					0.9	0
b_1					0.6	0
b_2					0.7	2
b_5					0.5	0

- After

	x	y	w	h	conf	cls
b_0					0.9	0
b_2					0.7	2
b_1					0.6	0
b_5					0.5	0

Step 3: NMS – Compute IoU

- **Iteration 1**
  - **The top bbox is determinand, compute IoU to others**
  - You can determine whether different classes should be considered ➔ We will consider in this example
  - Eliminate the bboxes whose confidence scores are **larger than IoU threshold**

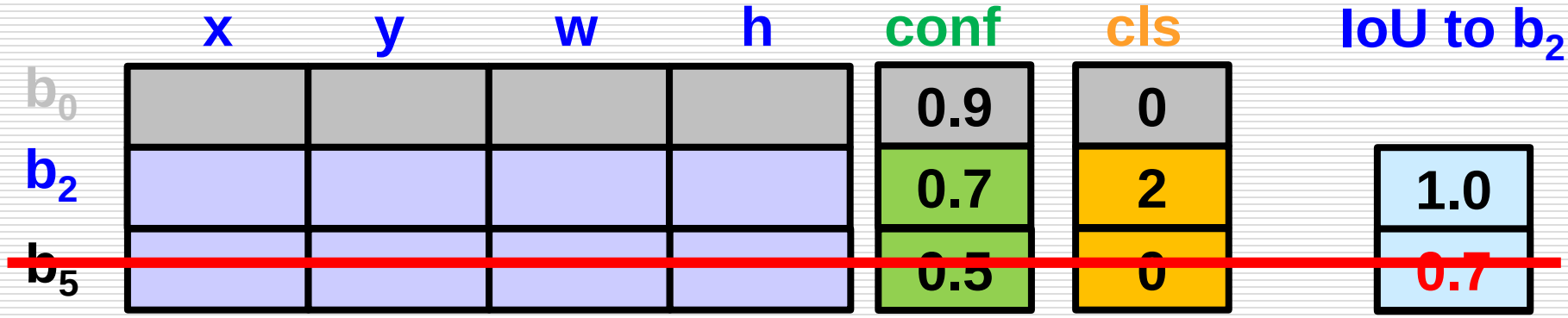
iou_threshold = 0.5

	x	y	w	h	conf	cls	IoU to b_0
b_0					0.9	0	1.0
b_2					0.7	2	0.2
~~b_1~~					~~0.6~~	~~0~~	~~0.6~~
b_5					0.5	0	0.3

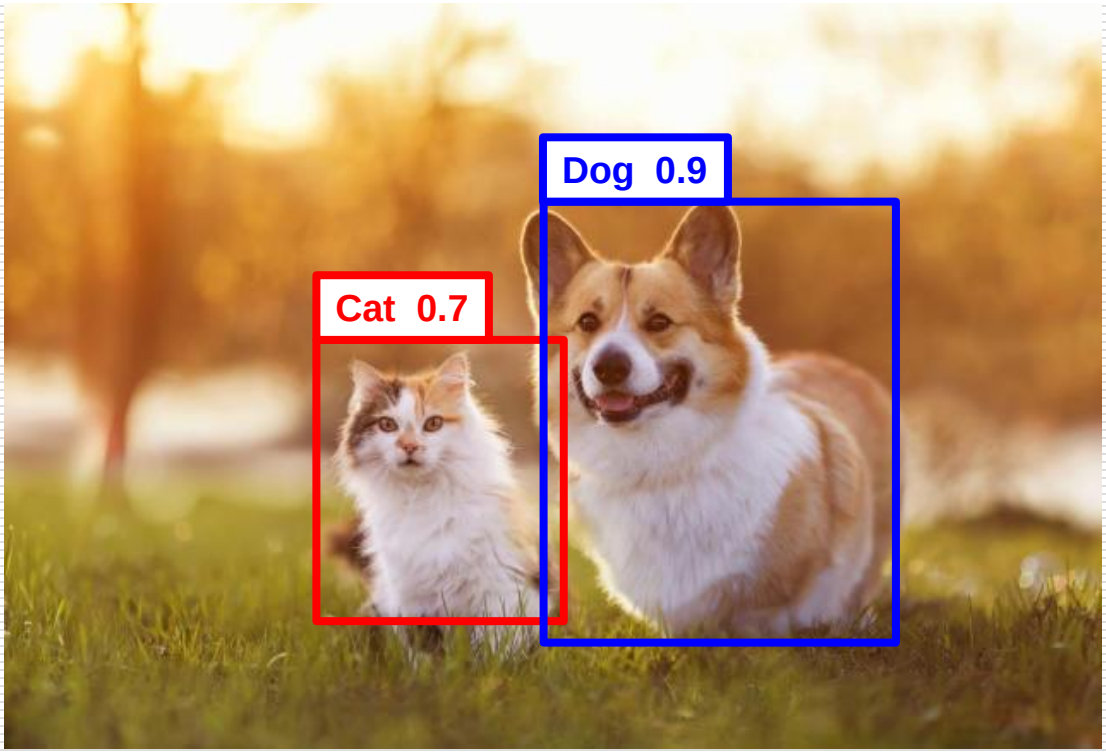
Step 4: NMS – Repeat Iteration

- **Iteration 2**
  - **b_0 is reserved, and the top bbox become b_2**
  - **Compute IoU to others**
  - Eliminate the bboxes according to **IoU threshold**

iou_threshold = 0.5



Step 5: Draw bboxes



	x	y	w	h	conf	cls	
b_0					0.9	0	Dog
b_2					0.7	2	Cat

Outline

- Introduction
- Data Preparation
- Models
- Training and Testing
- Discussion
  - Keypoint Detection
  - Polyp Detection
- Homework
- Reference

Fix the problems!
Express your creativity!

Keypoint Detection

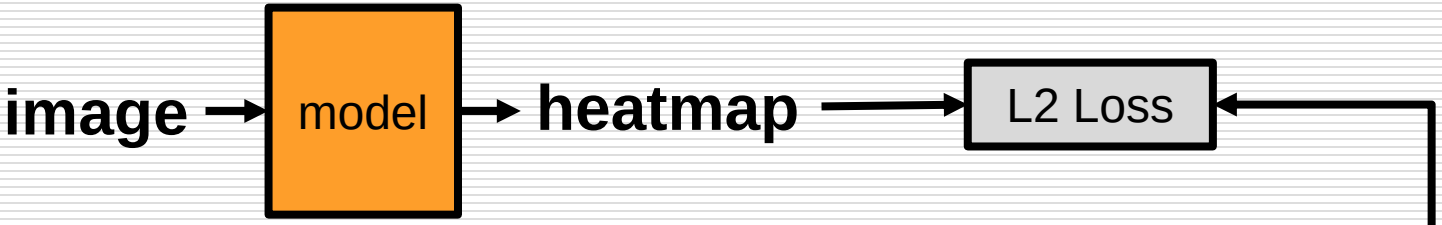
- COCO Dataset
  - Human keypoint estimation
  - **17 keypoints** for each human
  - **Multiple people** in an image

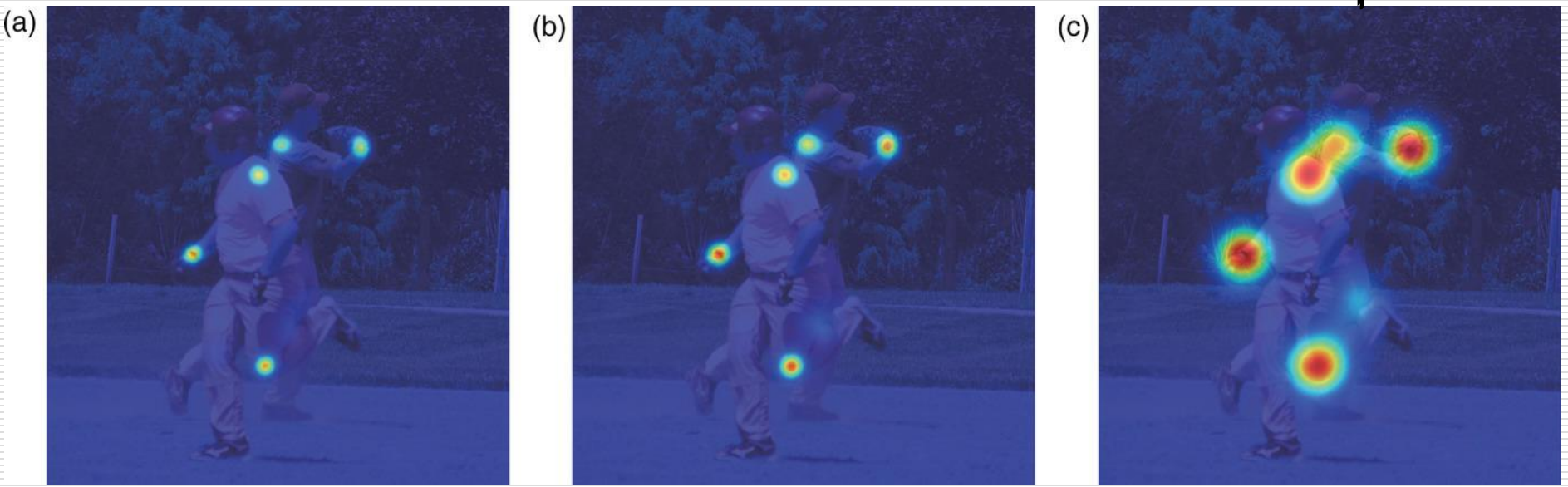
- Object Detection
  - See each keypoint as an object

- Matching Issue
  - **N people** in an image ➔ **17N keypoints** should be detected

Previous Work (1/2)

- **CNN-base** model, with **heatmap-base** training

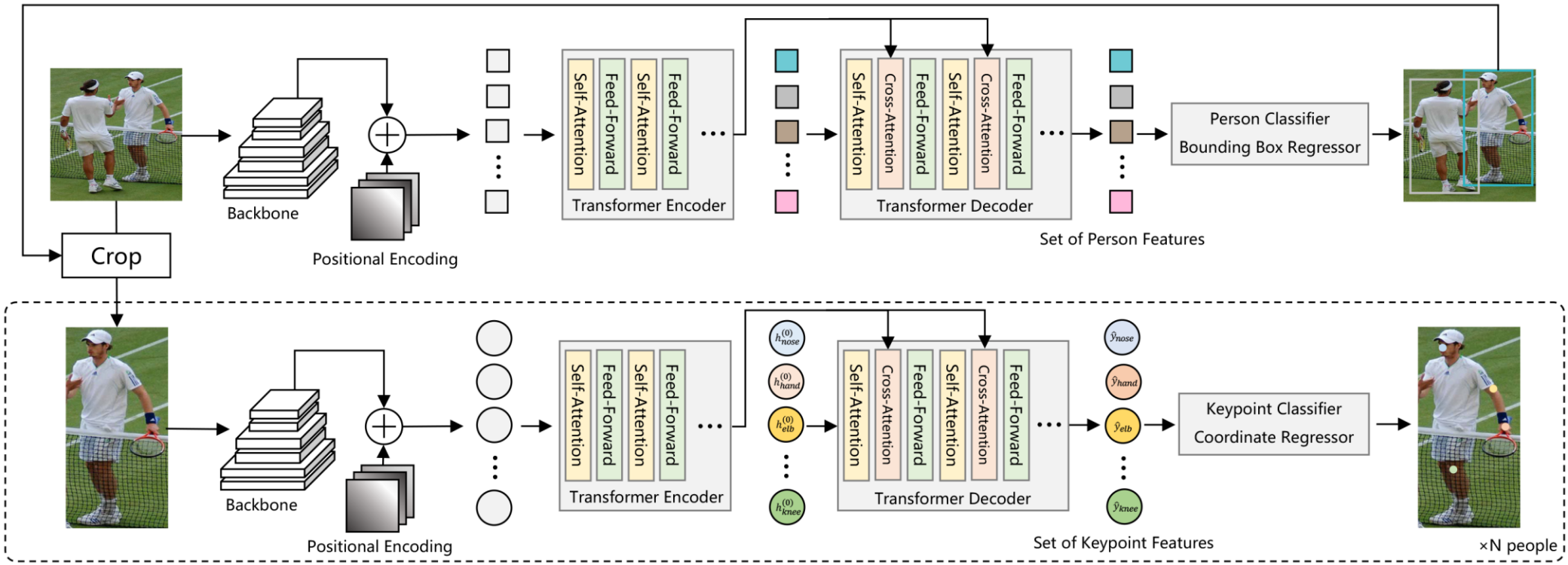




Gaussian Distribution Heatmap

Previous Work (2/2)

- **Transformer-base** with **regression** training
  - **PRTR: Pose Recognition with Cascade Transformers**



Summary of Keypoint Detection

- To avoid matching issue ➔ cropping and inference one by one; **however…**
  - No target bboxes in the real world ➔ **Not practical**
  - Additional model to get bboxes ➔ **Not light**
  - Cropping and inference several times ➔ **Not fast**
  - Conclusion ➔ **Useless**

- So, is it possible to save?
  - One-stage
  - Regression
  - Anchor-base

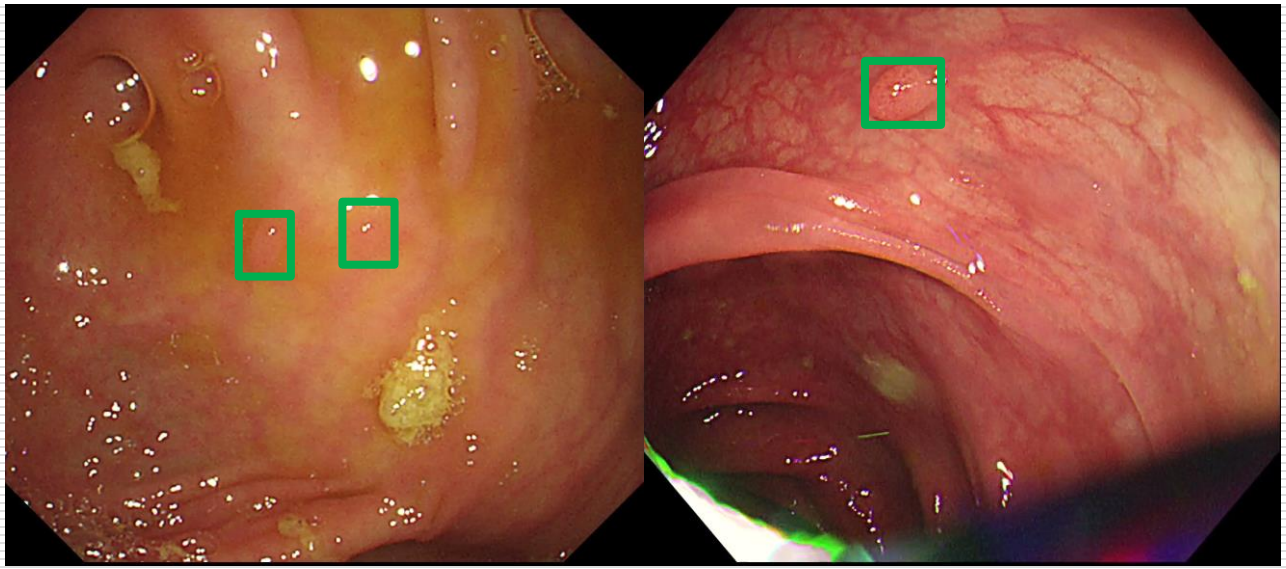
Polyp Detection

- Dataset (from CGMH)
  - 6,210 images for training
  - 704 images for validation
  - 15 20-minute videos for testing (no labeling)
  - 2 classes: **Hyperplastic (H)** / **Adenoma (A)**

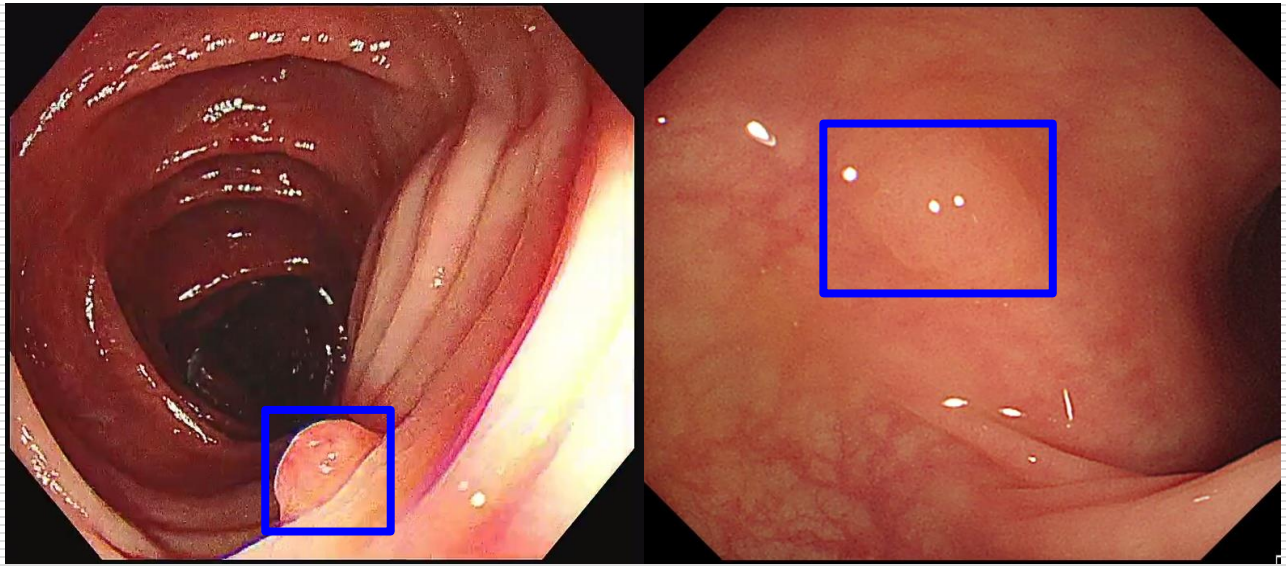
- Tasks
  - Localization: Precision, Recall
  - Classification: Accuracy

Categories

Adenoma



Hyperplastic



Objective

- **High performance** with **efficient computation**
  ➔ Avoid models with complicated architecture in our experiments
  ➔ Separate evaluations of two tasks (**localization** and **classification**)

- **Detection**: YOLO series(for **localization**)
  ➔ Real-time model with high performance

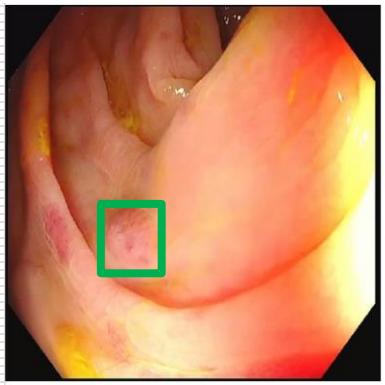
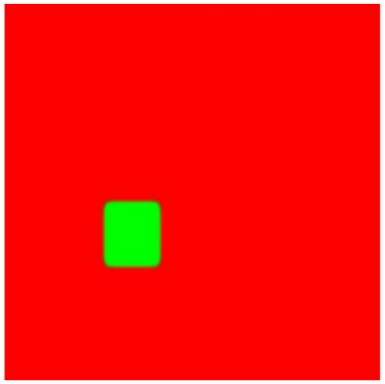
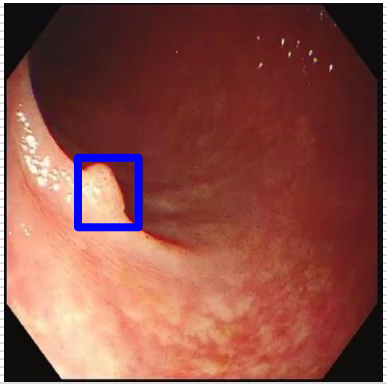
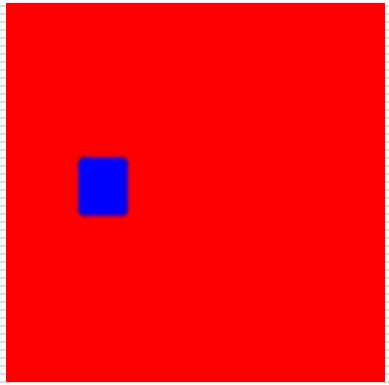
- **Segmentation**: UNet framework (for **classification**)
  ➔ The most common segmentation architecture

Method (1/2)

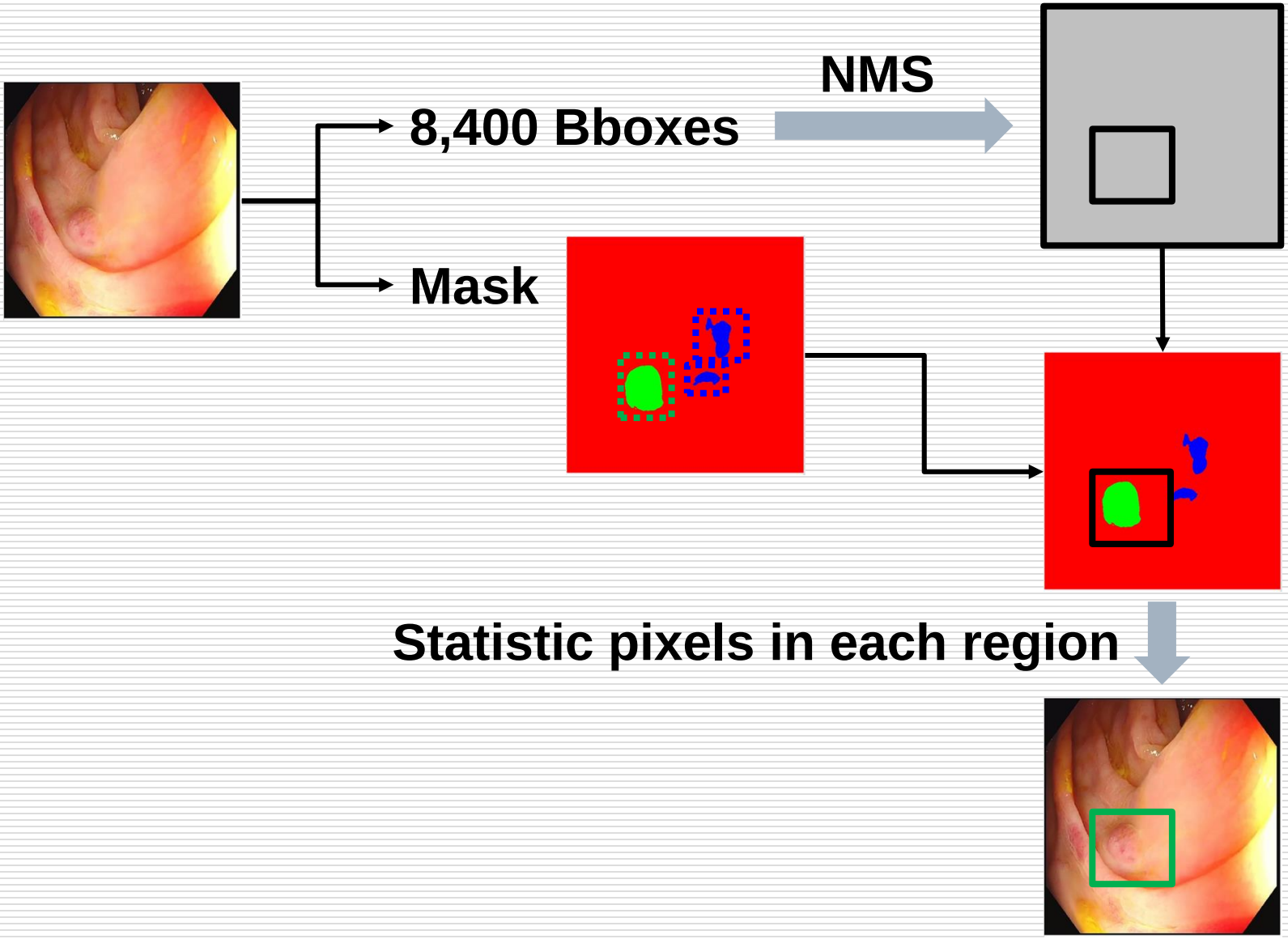
image

Detection → Bboxes (Localization)
- (x, y, w, h, conf) for each grid

Segmentation → Mask (Classification)
- One-hot labeling
- Zero vector for normal regions

Method (2/2)

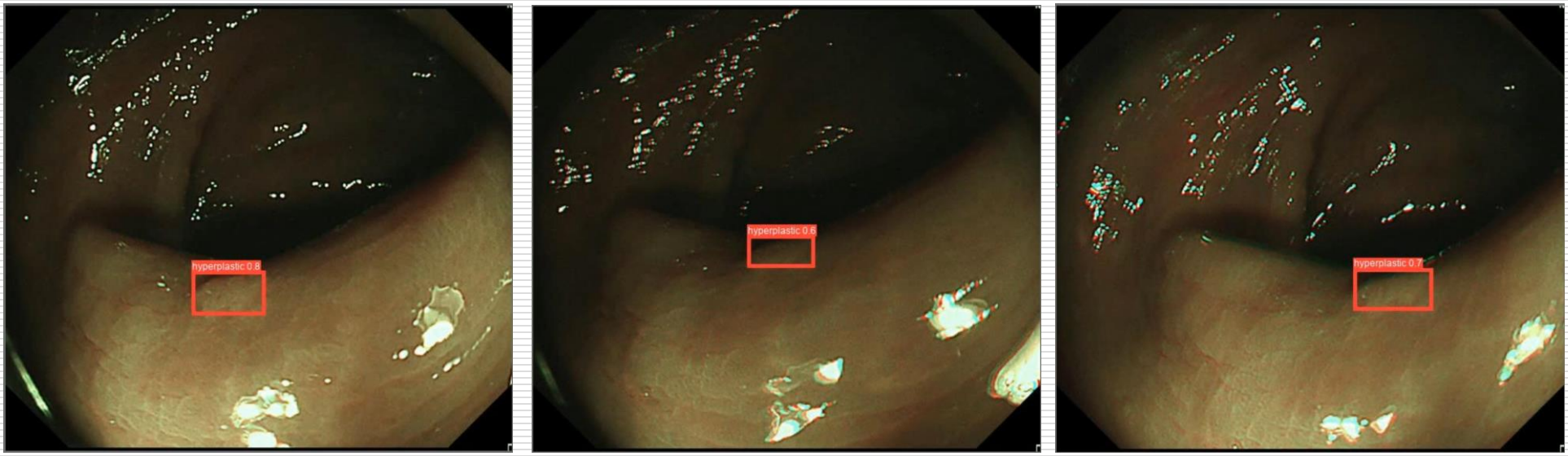


Projects (1/3)

- Anchor-based key points detection

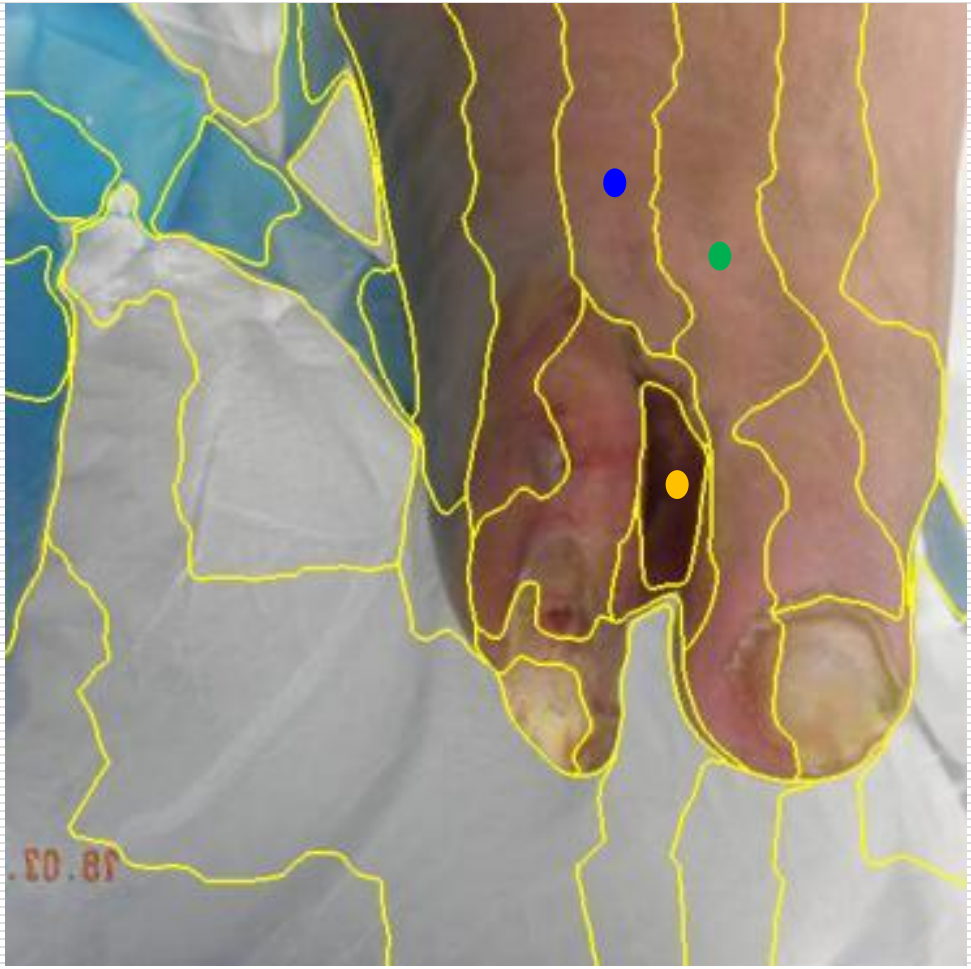
Projects (2/3)

- Object tracking



Projects (3/3)

- Region-based image segmentation



Outline

- Introduction
- Data preparation
- Models
- Training and Testing
- Discussion
- Homework ➔ How much you learned at this class?
- Reference

Homework

- Questions and Answers

- Email to **havenchen.ee11@nycu.edu.tw**
  - Filename: **<姓名>_lec8.pdf**
  - Email Topic: **AI Training Lec8 homework**

- ChatGPT should be an assistance, **NOT YOUR ANSWER!**

Questions (1/3)

- Q1: Explain the meanings of true positive(**TP**), False positive(**FP**), true negative(**TN**), false negative(**FN**).

- Q2: What is the correlation between **precision and recall**? Positive or negative? Why?

- Q3: What makes a **two-stage model slow**? List at least two reasons.

Questions (2/3)

- Q4: If there is a **384x384 image** and inferenced by a **YOLO** model. It has **4 output feature maps**, which are **stride 4, stride 8, stride 16, and stride 32**. It also has **2 different types of anchors**. How many bboxes before post-processing?

- Q5: There are two images name img1 and img2. img1 has a target (x_1, y_1, w_1, h_1, cls[0, 0, 1]), and img2 has a target (x_2, y_2, w_2, h_2, cls[0, 1, 0]). How to **mixup** them to a new image img3? And what "are" the targets of img3?

Questions (3/3)

- Q6: Considering the worst case of **NMS**, what is its time complexity?

- Q7: A **two-stage** model usually has better ability to **classification** than an **one-stage**. Why?

Outline

- Introduction
- Data Preparation
- Models
- Training and Testing
- Discussion
- Homework
- References

References (1/2)

- [1] COCO Format: https://cocodataset.org/#format-data
- [2] R-CNN介紹: R-CNN學習筆記. N | by Lung-Ying Ling | Taiwan AI Academy | Medium
- [3] Fast R-CNN介紹: Object Detection : R-CNN, Fast-RCNN, Faster RCNN | by John Shen | Medium
- [4] YOLO介紹: 深度學習-物件偵測:You Only Look Once (YOLO) | by Tommy Huang | Medium
- [5] YOLOv7 介紹: [Object Detection_YOLO] YOLOv7 論文筆記 - HackMD

References (2/2)

- [6] YOLOv8介紹: YOLOv8 深度详解！一文看懂，快速上手 - 知乎 (zhihu.com)
- [7] CenterNet介紹:物件偵測 - CenterNet 介紹. CenterNet目前應該算是很好入門的一篇One-stage目標檢測... | by 22 12 | Medium
- [8] DETR介紹: 理解DETR - 知乎 (zhihu.com)
- [9] Focal Loss介紹: https://zhuanlan.zhihu.com/p/49981234
- [10] LDAM介紹: https://zhuanlan.zhihu.com/p/308298563

Thank you