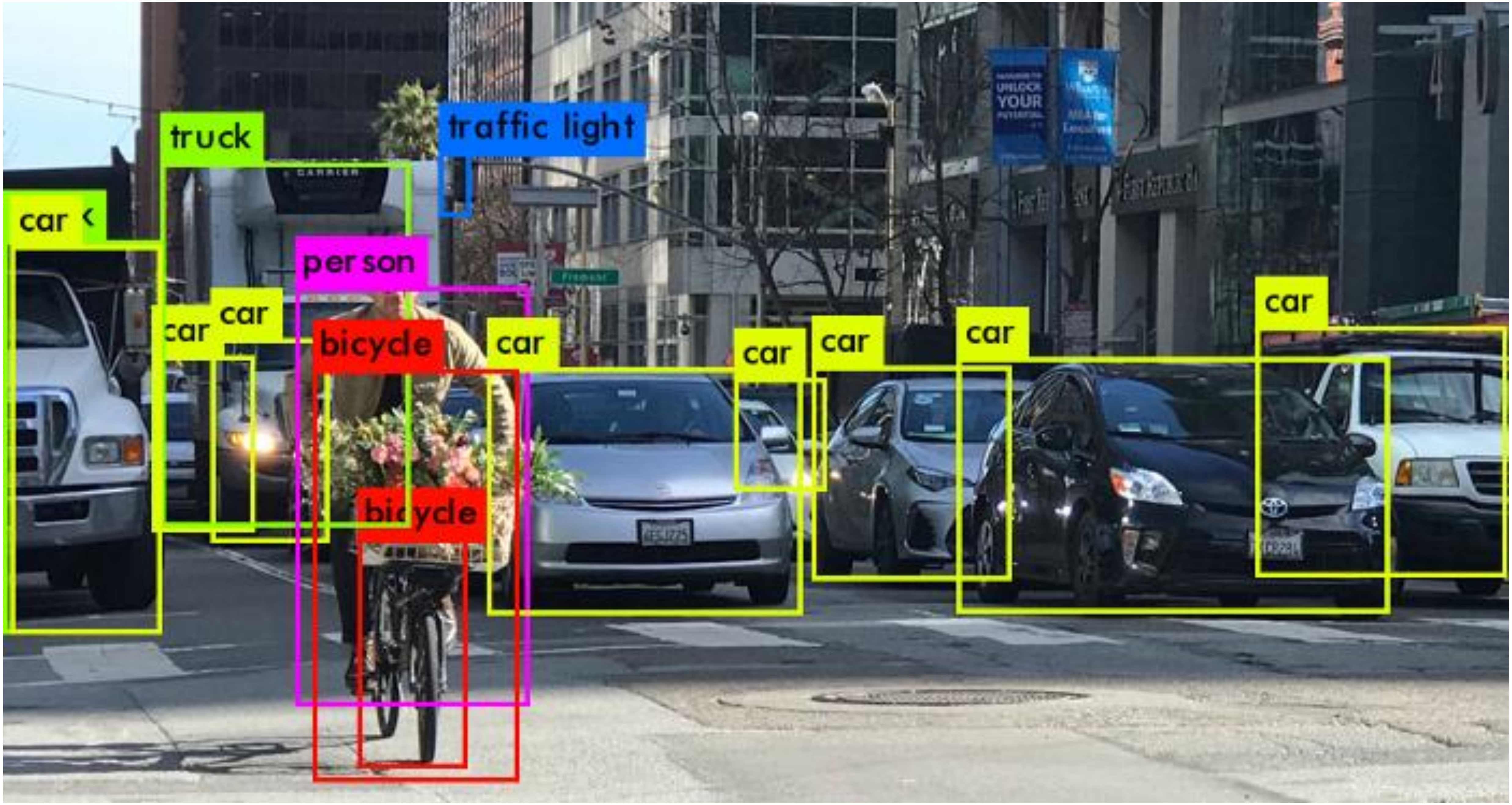
YOLO ALGORITHM

YOU ONLY LOOK ONCE



truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle
UNLOCK YOUR

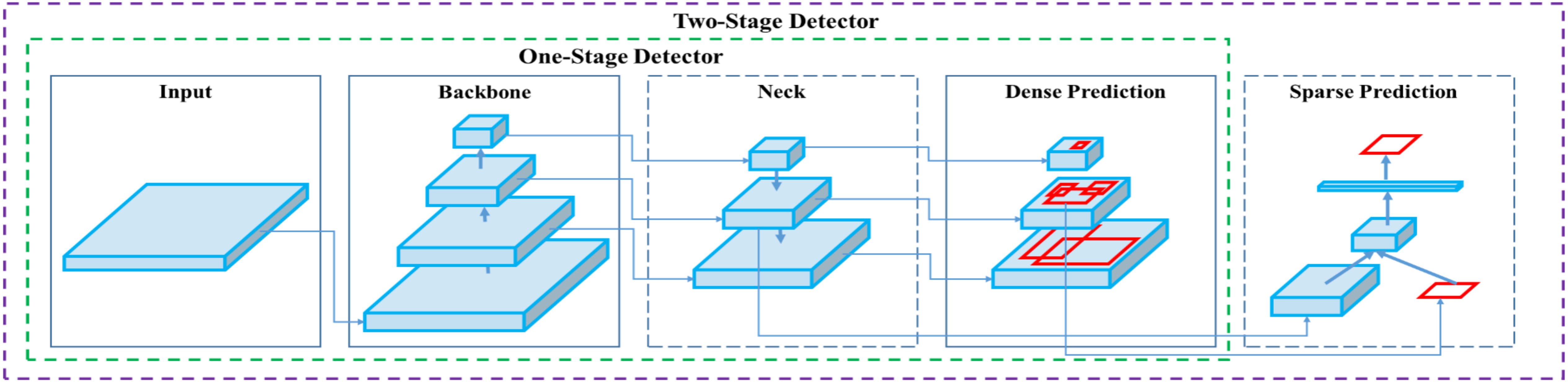
YOLO Object detector

- A modern detector is composed of two parts backbone and head.
- Backbone is pre-trained on ImageNet and a head is used to predict classes and bounding boxes of objects.
- All real-time object detectors aim to minimize inference time and **maximize** accuracy for achieving the optimal tradeoff between speed and accuracy.

Two different object detectors

- Two-stage object detectors and one stage-detectors which are further **subdivided** into **anchor-based** and **anchor-free detectors**.
- Two-stage detectors **decouple** the task of object localization and classification for each bounding box.
- One-stage detectors make the **predictions** for object localization and classification at the **same time**.

YOLO building blocks



- **Input:** Image, patches, Pyramid
- **Backbone:** VGG16, ResNet-50, SpineNet, EfficientNet-B0-B7, CSPResNext50, CSPDarknet53.
- **Neck:** Additional Blocks: SSP, ASPP, RFB, SAM and

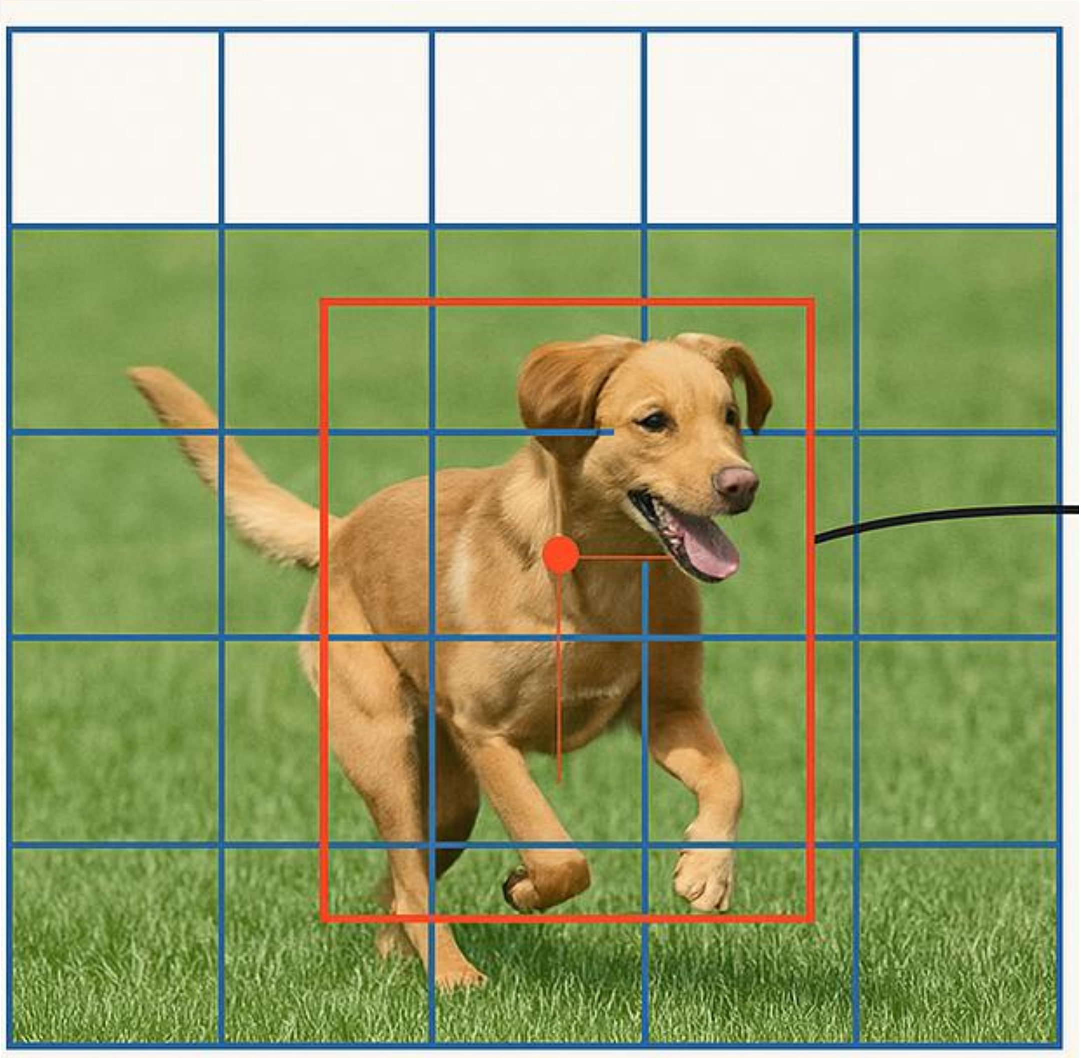
  Path-aggregation blocks: FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM
- **Heads:**
  - **Dense prediction (one-stage):**

    RPN, SSD, YOLO, RentinaNet (anchor-based), and CornerNet, CenterNet, MatrixNet (anchor-free)
  - **Sparse prediction:** Faster R-CNN, R-FCN, Mask R-CNN (anchor-based), and RepPoints (anchor-free)

YOLO building blocks

- **Backbone (Feature Formation)** is acts as a feature extractor. All of the backbone models are basically classification models. E.g., VGG16, SqueezeNet, MobileNet, ShuffleNet, but all are CPU based training only.
- All object detectors take an image as input and compress features down through a CNN backbone.
- In image classification, these backbones are the end of the network and prediction can be made off of them.
- **Neck (Feature Aggregation)** is a subset of the bag of specials. It **collects feature maps** from different stages of the backbone called feature aggregator.
- In object detection, multiple bounding boxes need to be drawn around images along with classification, so the feature layers of the convolutional backbone need to be mixed and held up in light of one another.
- The combination of backbone feature layers happens in the neck.
- **Head (Detection)** is the object detector that finds the region where the object might be present. But doesn't tell about which object is present in that region.

What is YOLO?

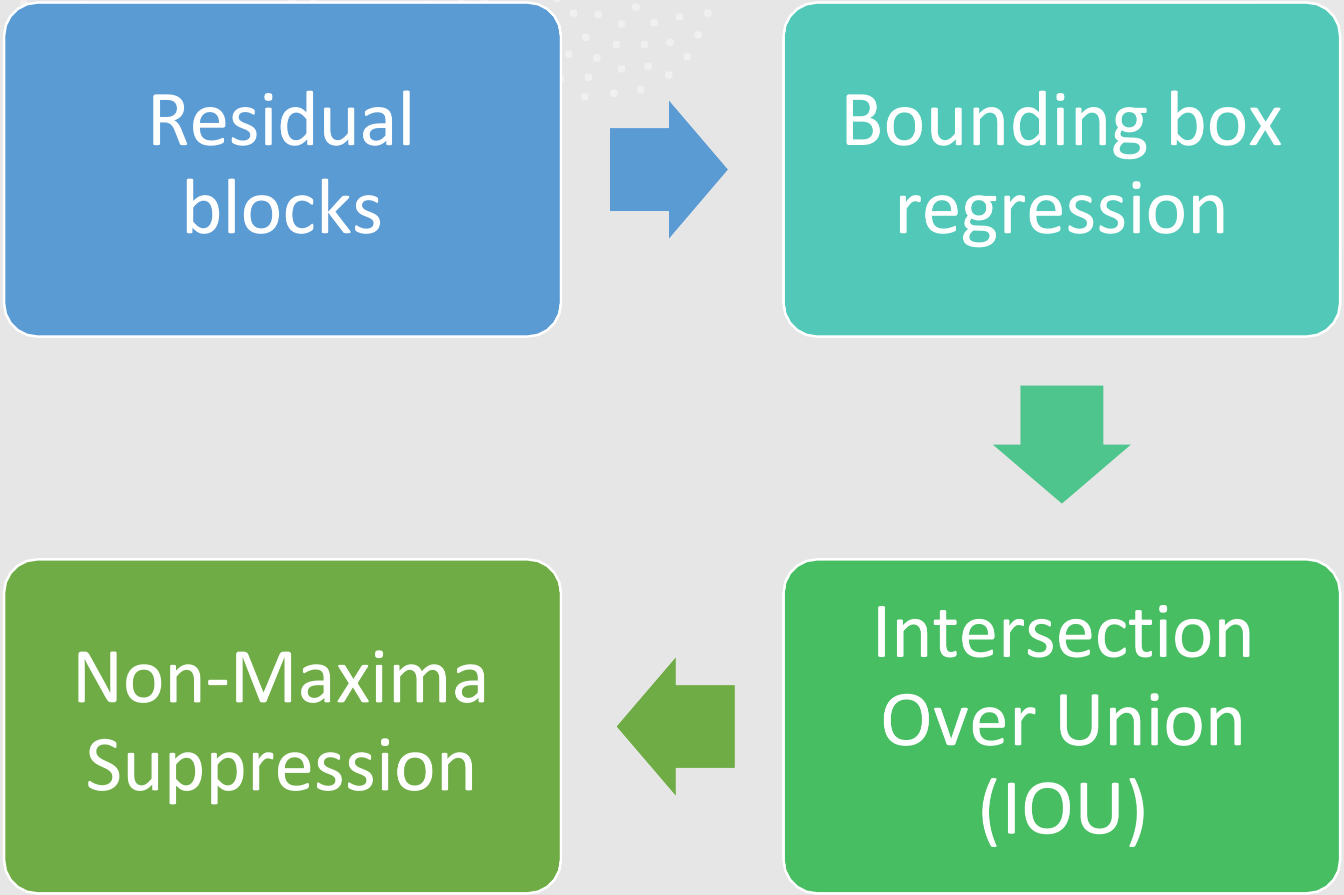


- Algorithm that detects and recognizes various objects in a picture – **in real-time**
- Treats object detection as a **regression problem** instead of a classification problem.
- Regression task concerning **spatially separated bounding boxes** and **associated class probabilities**, using a single neural network
- Requires only a **single forward propagation** through a neural network to detect objects
- Sees the entire image during training and testing, as a result of which it can **encode contextual information about the classes as well as their appearance**

Why use YOLO?

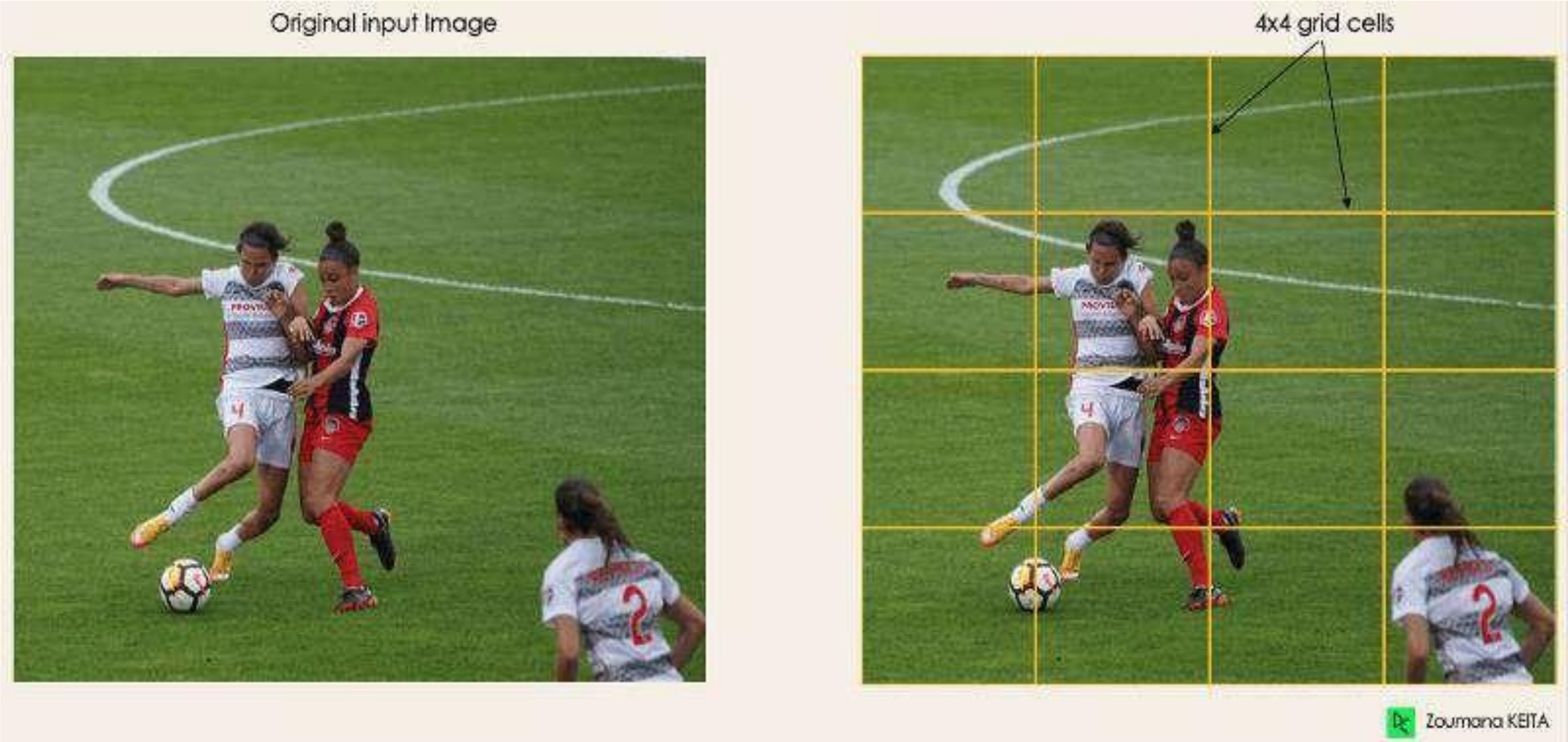
- **Speed:** This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

Working of YOLO



Residual blocks

- Dividing the original image (A) into NxN grid cells of equal shape
- Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability or confidence value
- If an object center appears within a certain grid cell, then this cell will be responsible for detecting it



Residual blocks

Bounding Box Regression

- A bounding box is an outline that highlights an object in an image.

- Every bounding box consists of the following attributes:

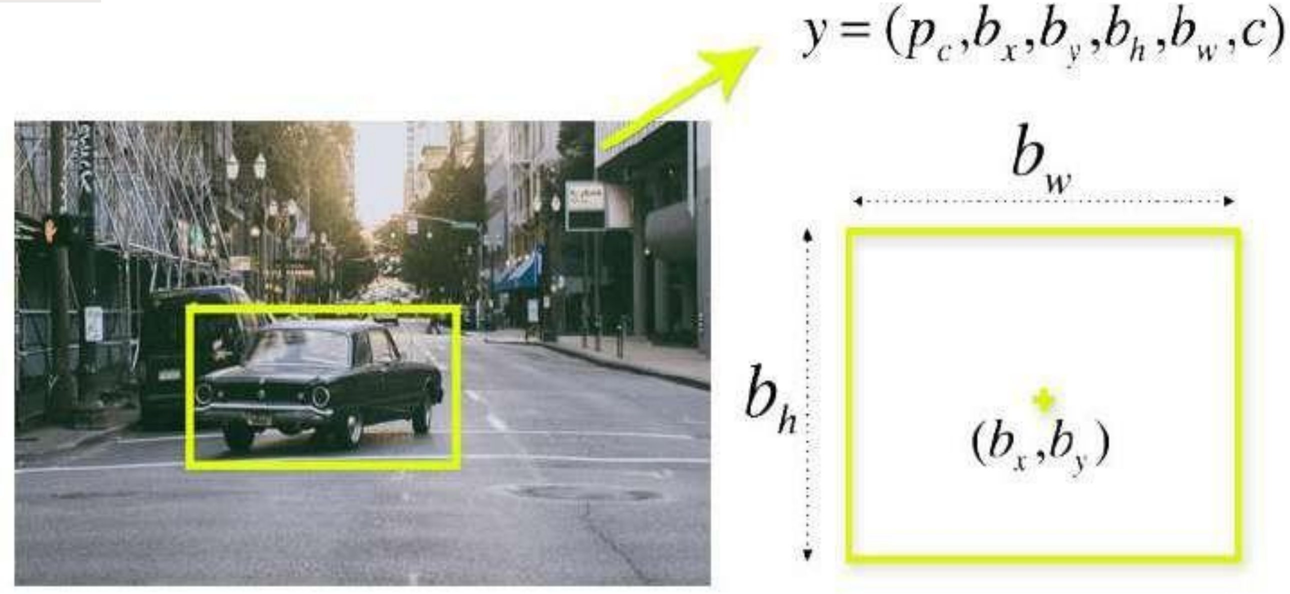
-Width (bw)

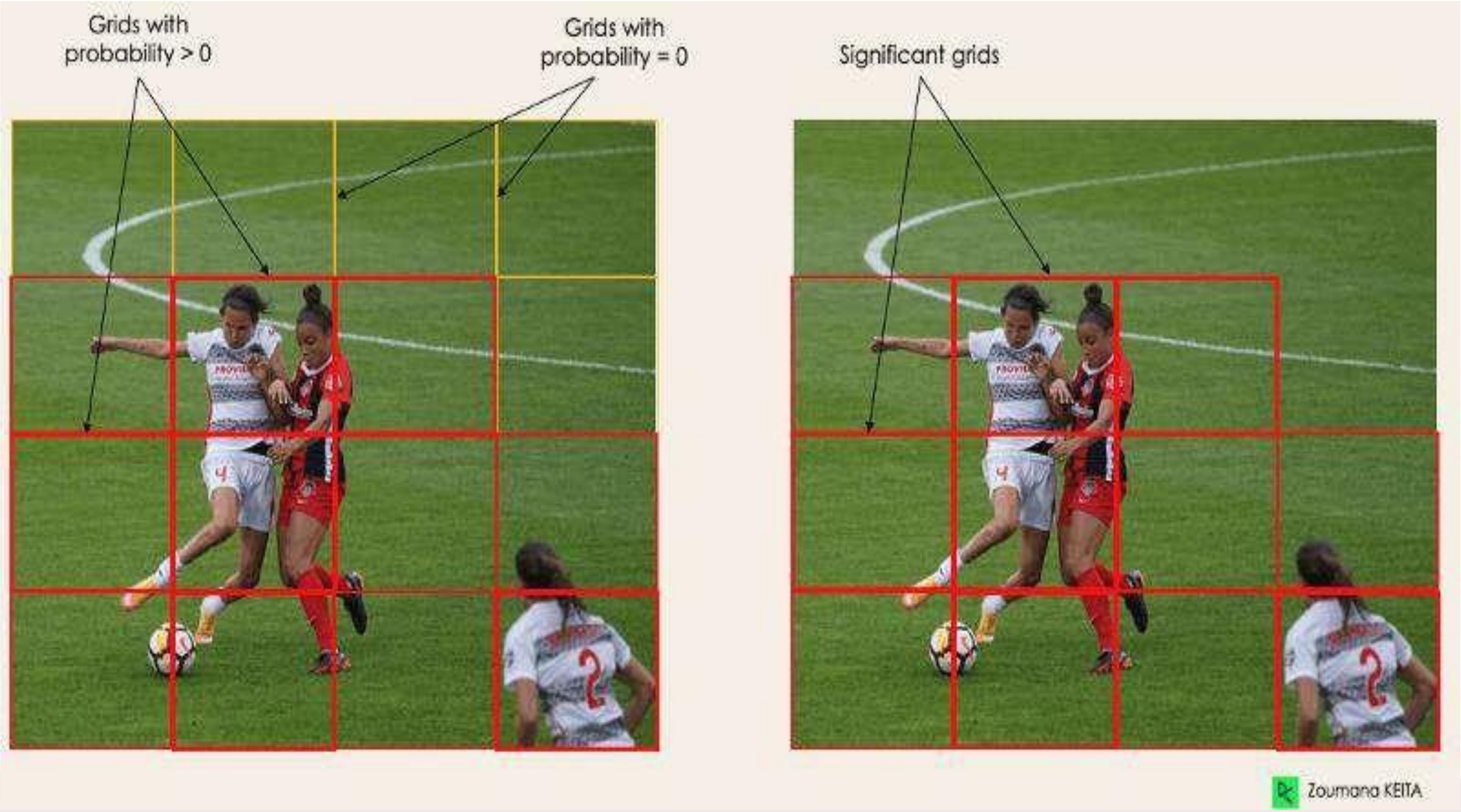
-Height (bh)

-Classes (ci)

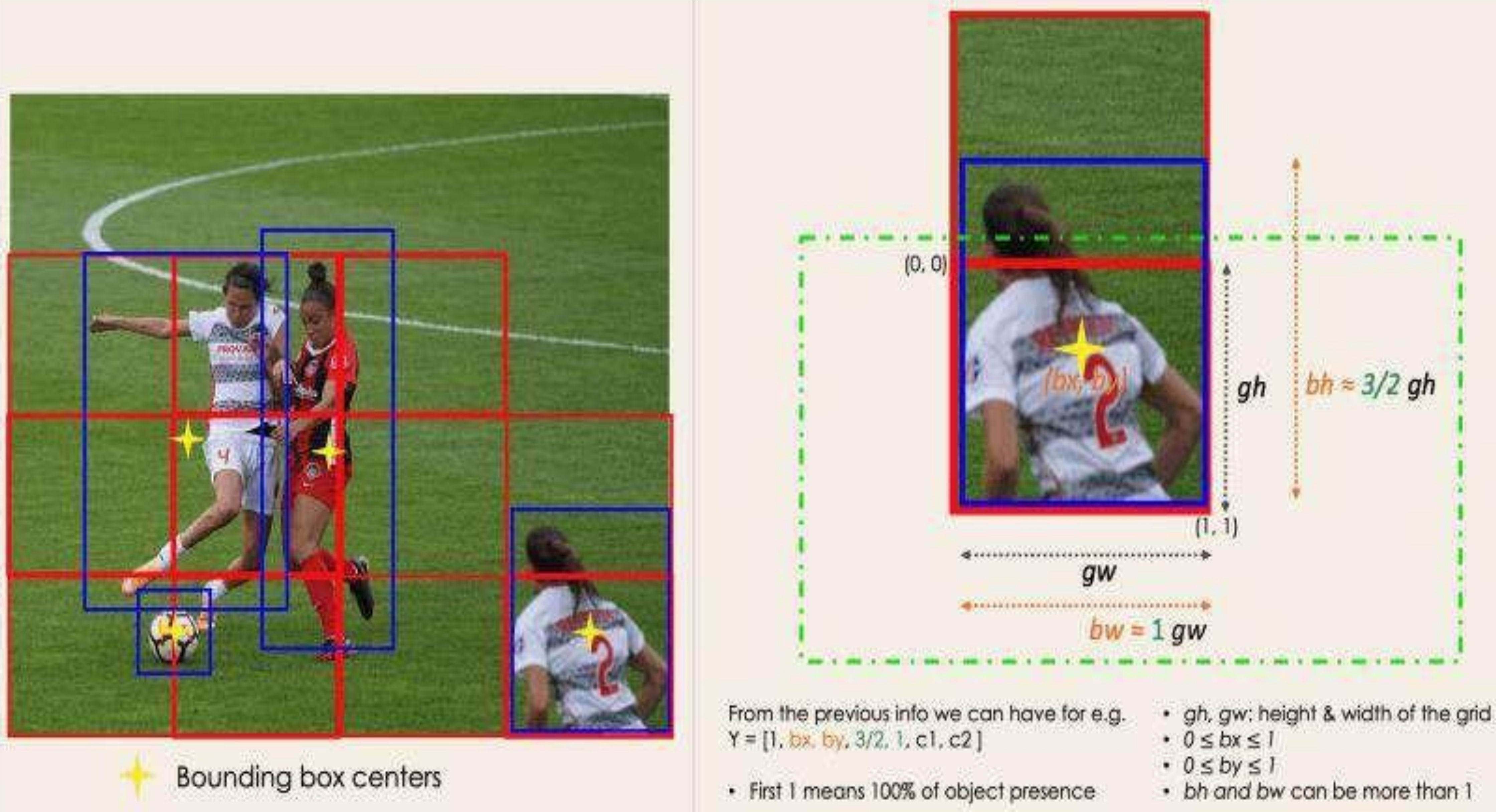
-Probability score (pc)

-Bounding box center (bx,by)



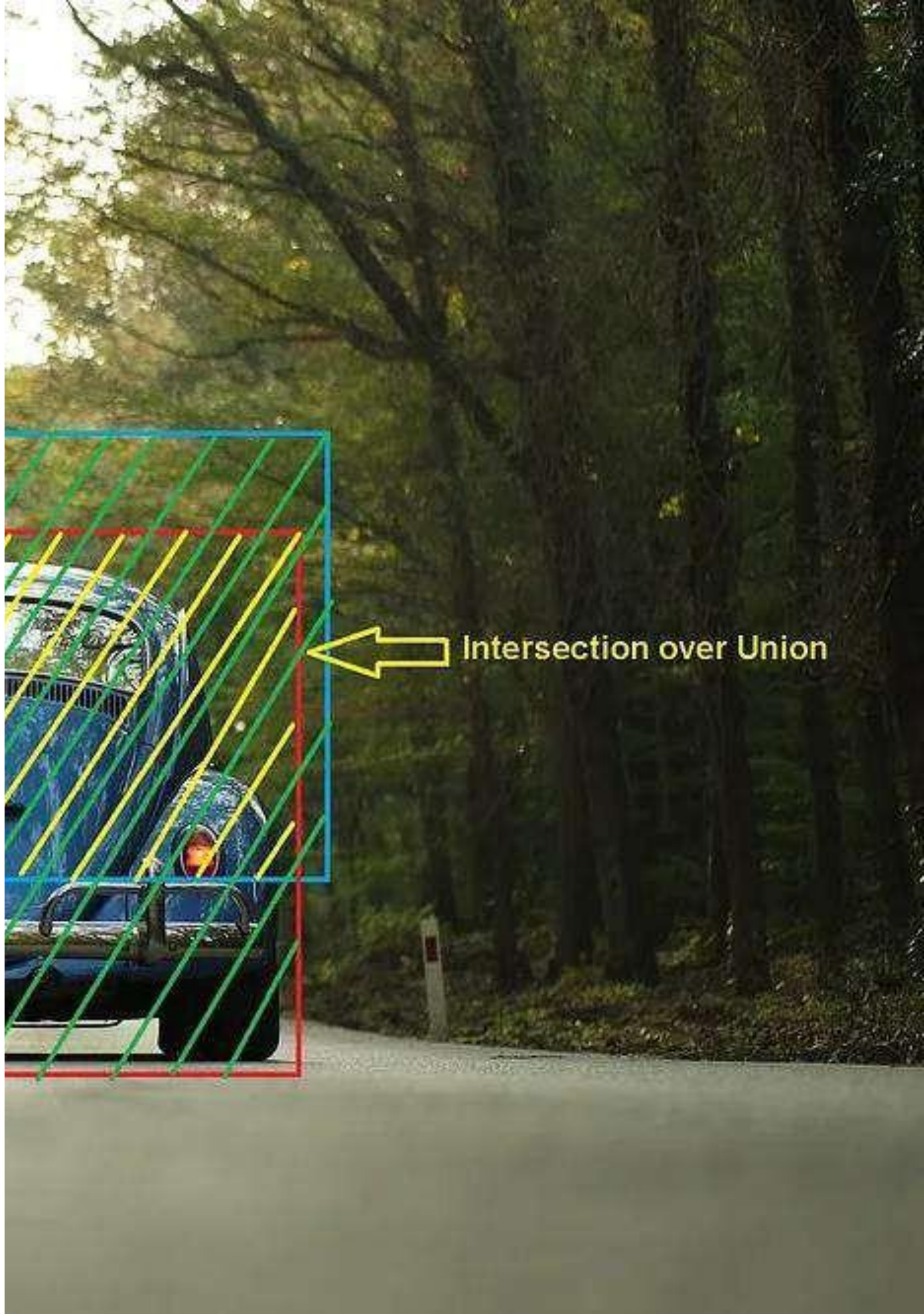


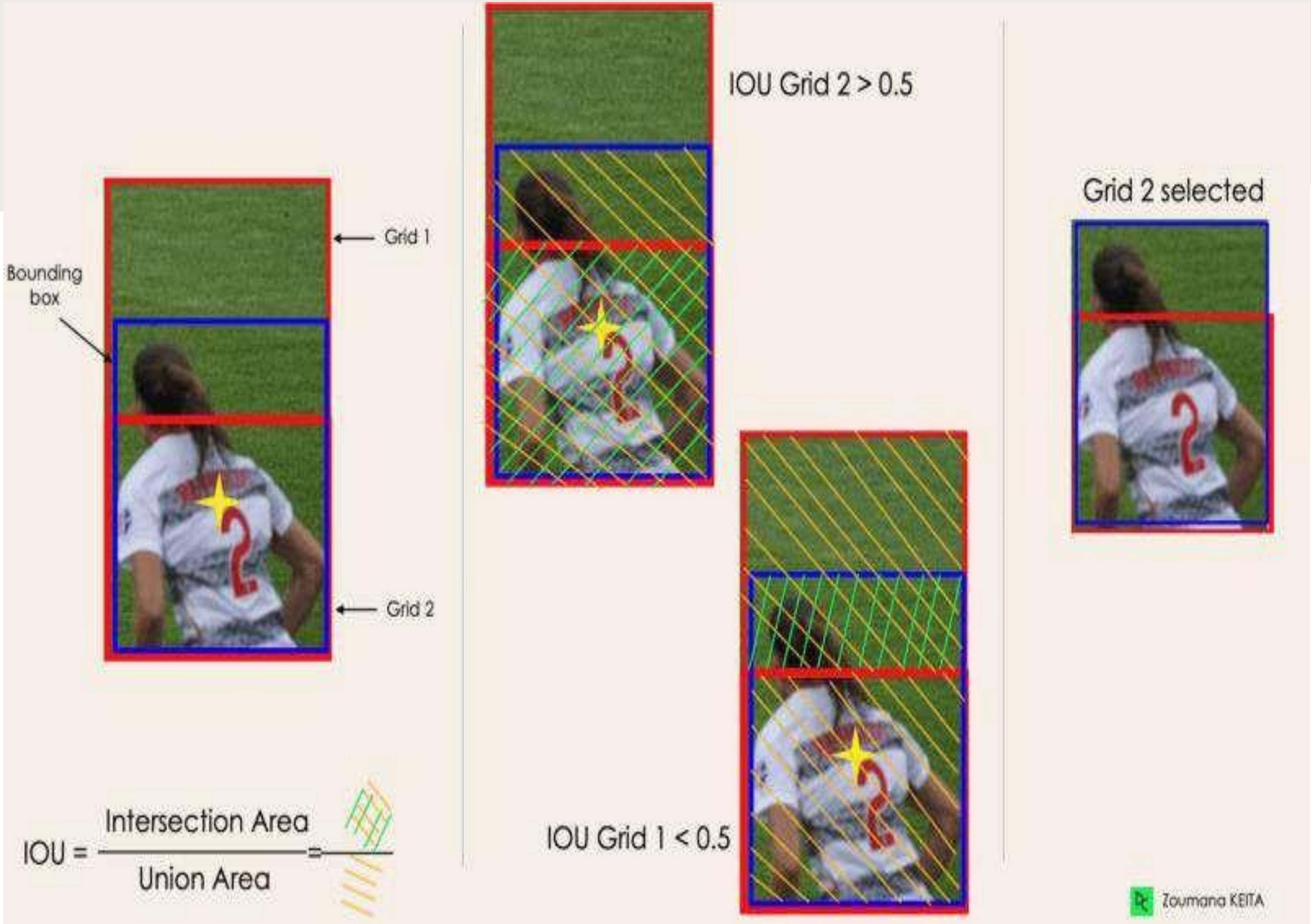
Bounding Box Regression

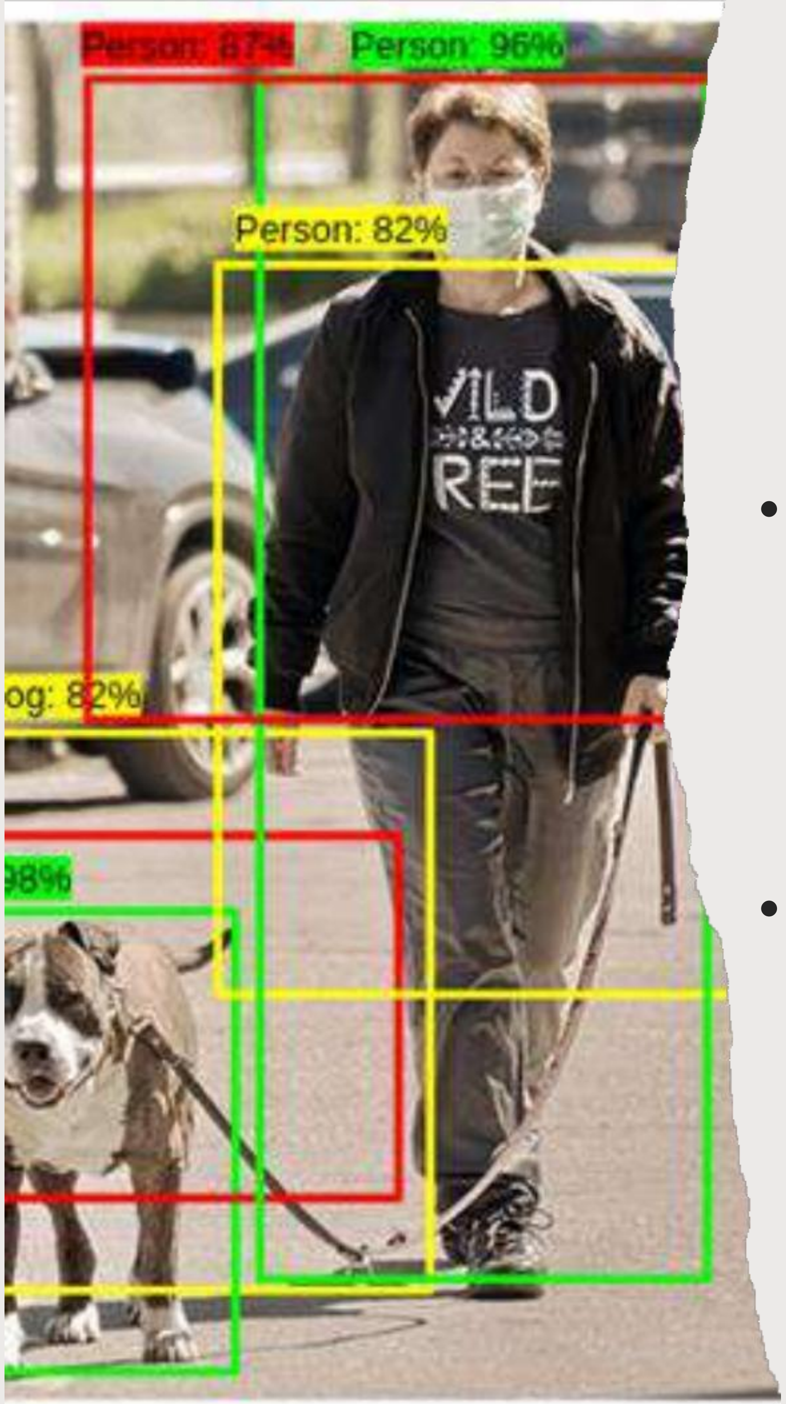

Bounding box centers
(0, 0)
(1, 1)
gh
bh ≈ 3/2 gh
gw
bw ≈ 1 gw
From the previous info we can have for e.g.
Y = [1, bx, by, 3/2, 1, c1, c2]
• First 1 means 100% of object presence
• gh, gw: height & width of the grid
• 0 ≤ bx ≤ 1
• 0 ≤ by ≤ 1
• bh and bw can be more than 1

Intersection over Unions IoU

- Single object in an image can have multiple grid box candidates for prediction
- IOU is the overlap between the ground truth and the predicted bounding box : calculates how similar the predicted box is with respect to the ground truth
- Goal of the IOU is to discard such grid boxes to only keep those that are relevant



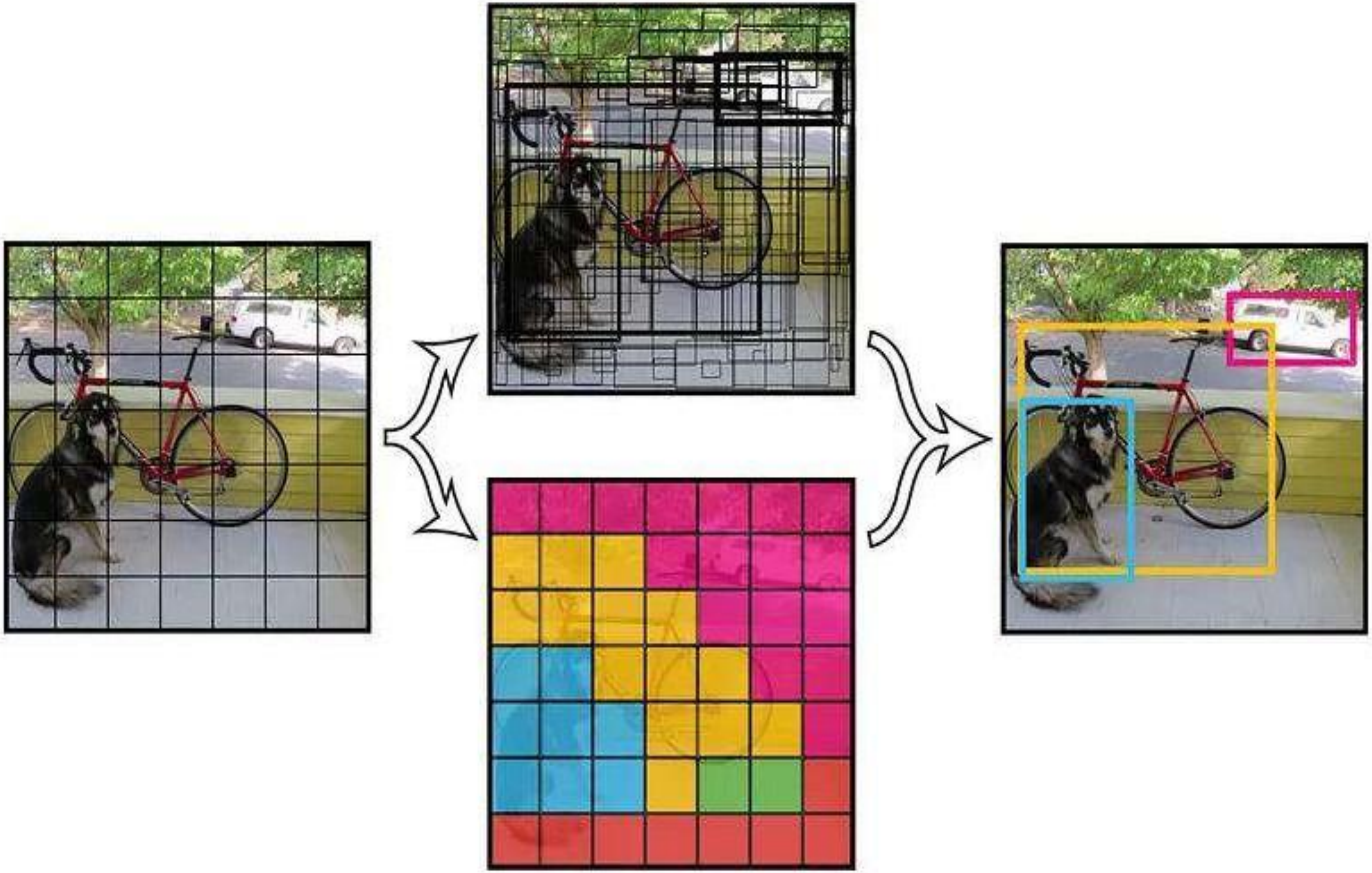

IOU Grid 2 > 0.5
Grid 2 selected
Grid 1
Bounding box
Grid 2
IOU Grid 1 < 0.5
IOU = Intersection Area / Union Area =
Zoumana KEITA



Non-Max Suppression

- Threshold for the IoU is not always enough because an object can have multiple boxes with IOU beyond the threshold

- Use NMS to keep only the boxes with the highest probability score of detection



TLDR

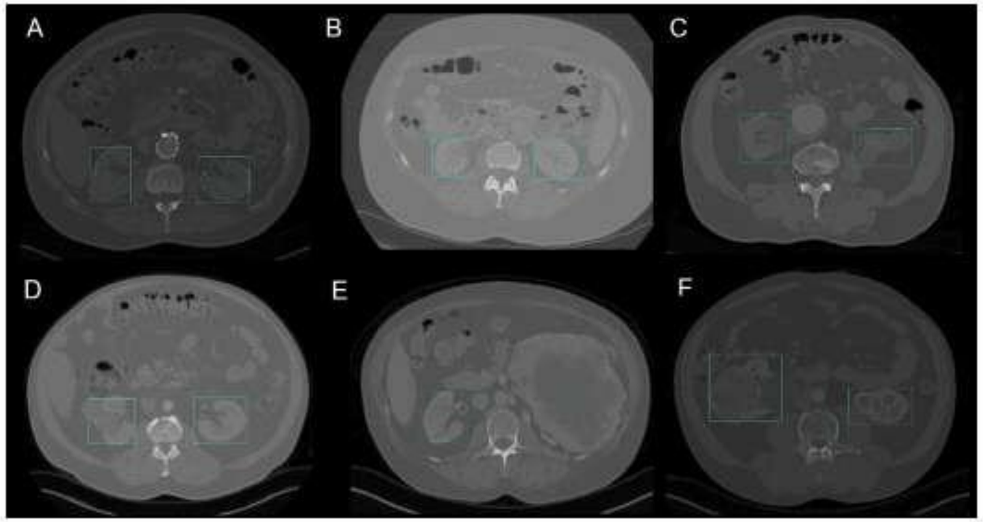
1. Input image is divided into a **grid of cells**
2. **Each cell predicts bounding boxes**: For each cell in the grid, YOLO predicts a set of bounding boxes, along with their confidence scores.
3. Predictions are refined using **IoU and non-max suppression**
4. **Class probabilities are assigned to each bounding box**: YOLO calculates the probability that the box contains each possible class of object. This is done using an activation function applied to the output of the last layer of the network
5. **Final predictions are made**: Finally, YOLO selects the bounding box with the highest-class probability as the final prediction for each object in the image.

APPLICATIONS

- Healthcare

- Can be challenging to localize organs in real-time, due to biological diversity from one patient to another

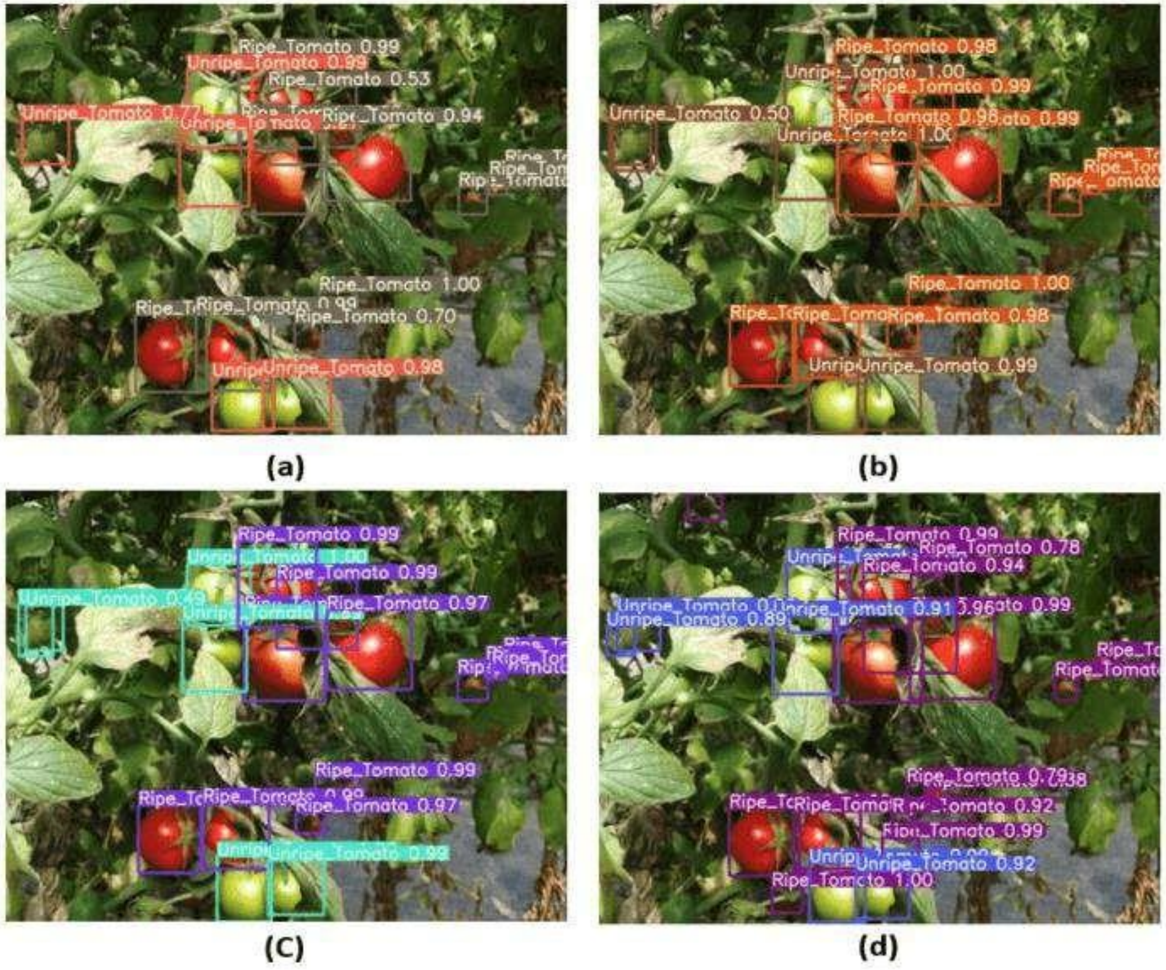
 - Kidney Recognition in CT used YOLOv3 to facilitate localizing kidneys in 2D and 3D from CT scans



APPLICATIONS

- Agriculture

In Tomato detection based on modified YOLOv3 framework: used YOLO to identify the types of fruits and vegetables for efficient harvest



APPLICATIONS

- Security surveillance

- Self-driving cars

The real-time aspect of YOLO makes it a better candidate compared to simple image segmentation approaches.

LIMITATIONS

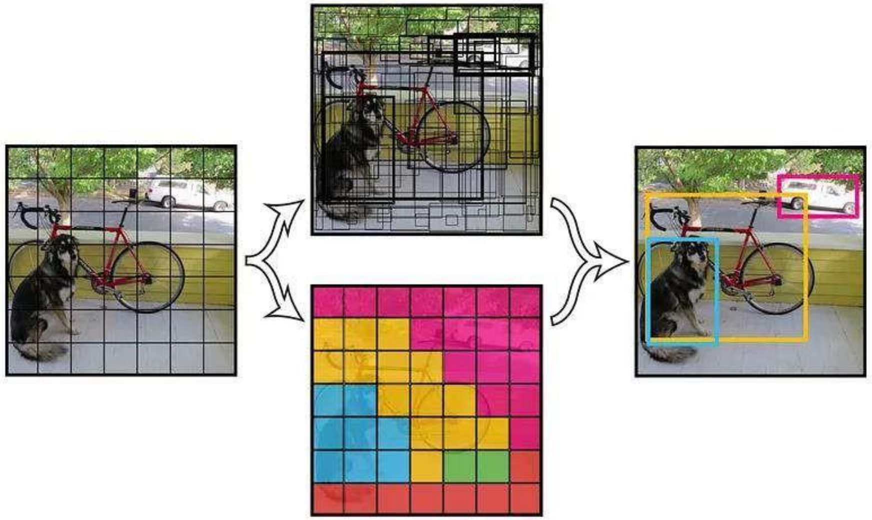
- Like many object detection algorithms, **struggles to detect small objects**. It might fail to accurately detecting objects in crowded scenes or when objects are far away from the camera

- YOLO imposes **strong spatial constraints** on bounding box predictions

- Difficulty in detecting objects that are either **very large or very small compared to the other objects in the scene**

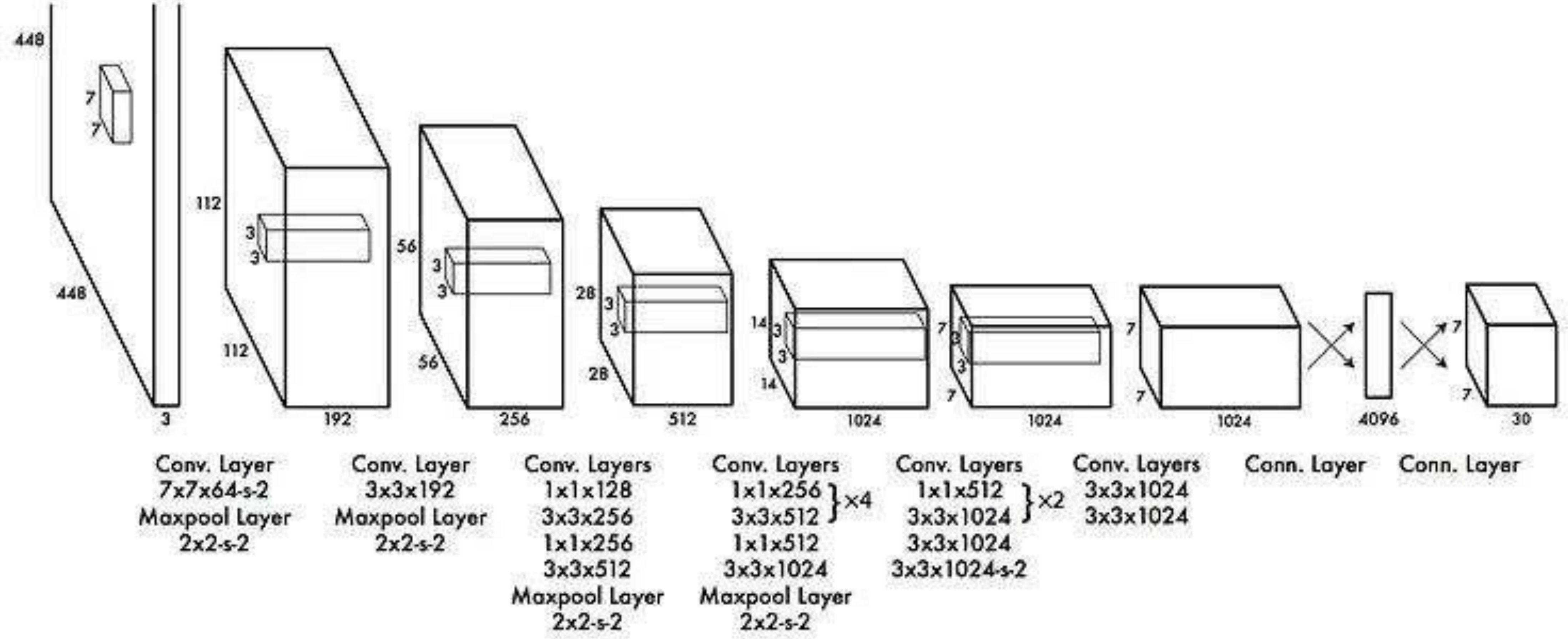
YOLO v1 Architecture

- The network consists of 24 convolutional layers and 2 fully connected layers. Also, a Leaky ReLU is used as an activation function
- It uses features from the entire image and predicts bounding boxes simultaneously.



YOLO v1 Architecture Cont…

- YOLO’s evaluation on the Pascal VOC detection dataset
- For evaluating YOLO on Pascal VOC, the parameters were set as follows: S = 7, B = 2 and C = 20, meaning there are 7 * 7 * (2 * 5 + 20) predictions per image



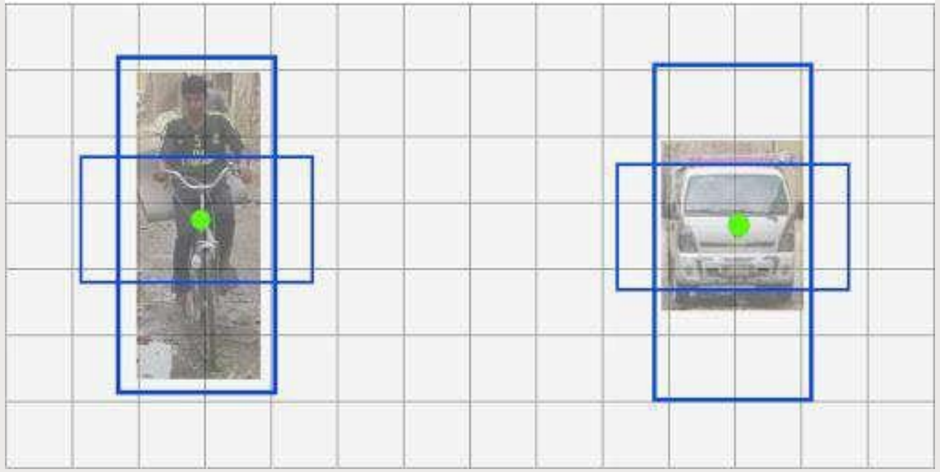
YOLO v1 Architecture

YOLO v2

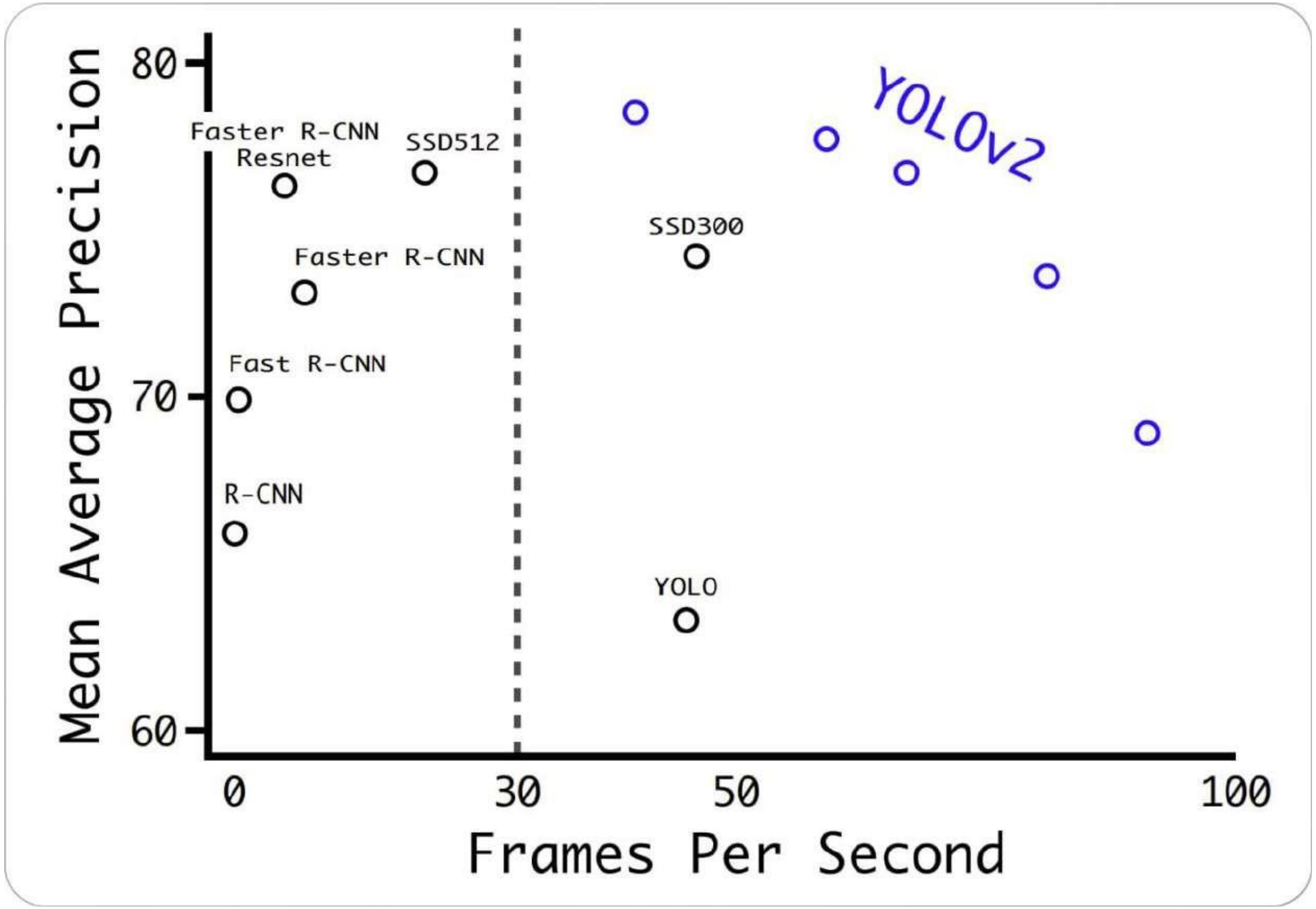
- Introduction
    - YOLO v2 was introduced in 2016
    - Developed to surpass YOLOv1 in terms of speed and accuracy and also be able to detect a wider range of object classes

YOLO v2 Cont...

- What were these improvements YOLO v2 brought?
    - Batch Normalisation
    - Higher input resolution
    - Convolutional Layers using Anchor Boxes
    - Dimensionality Clustering
    - Fine Grained Features



Bounding Boxes with more than 1 anchors (to provide more accurate localisation)



Mean Average Precision
80
70
60
Faster R-CNN
Resnet
SSD512
Faster R-CNN
Fast R-CNN
R-CNN
SSD300
YOLO
YOLOv2
0
30
50
100
Frames Per Second

YOLO v2 Architecture

- YOLO v2 is trained on different architectures such as VGG-16 and GoogleNet, and uses the Darknet-19 architecture

- The reason for choosing the Darknet architecture is its lower processing requirement than other architectures

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Darknet-19 architecture

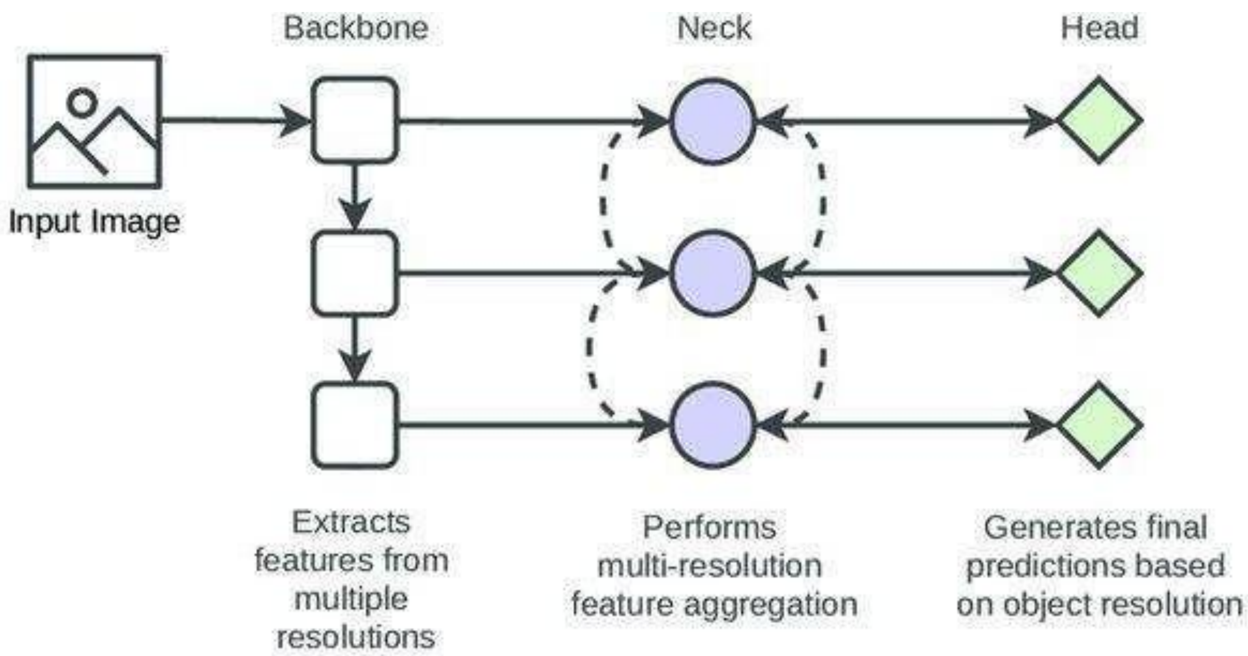
YOLO v7

- YOLO v7 was released in July 2022 in the paper-Trained bag-of-freebies sets new state-of-the-art for real-time object detectors
- YOLO v7 made a significant move in the field of object detection, and surpassed all the previous models in terms of accuracy and speed
- It achieved this by making two changes. One at the architectural level and another at the Trainable bag-of-freebies level

Changes YOLOv7 made

- Architectural level:
  - YOLO v7 reformed its architecture by integrating the Extended Efficient Layer Aggregation Network (E-ELAN) which allows the model to learn more diverse features
  - YOLOv7 also scales its architecture by concatenating the architecture of the models it is derived from such as YOLO v4, Scaled YOLO v4, and YOLO-R
- Trainable bag-of-freebies:
  - The term bag-of-freebies refers to improving the model's accuracy without increasing the training cost
  - This is the reason why YOLOv7 increased not only the inference speed but also the detection accuracy

Model	Type	Backbone	Neck	Head	Loss Function	Framework
YOLOv3	Fully convolution	Darknet-53	FPN	YOLOv3 head	Binary cross entropy	Darknet
YOLOv4	Fully convolution	CSPDarknet53	SPP and PANet	YOLOv3 head	Binary cross entropy	Darknet
YOLOv5	Fully convolution	CSPDarknet53	PANet	YOLOv3 head	Binary cross entropy and Logits loss	Pytorch
YOLOv6	Fully convolution	EfficientRep	Rep-PANet	Decoupled head	Varifocal Loss and Distribution Focal Loss	Pytorch
YOLOv7	Fully convolution	E-ELAN	FPN and PANet	Multiple heads	Binary cross entropy with Focal loss	Pytorch



Layers	YOLOv3	YOLOv4	YOLOv5	YOLOv7	YOLOv6
Backbone	**Darknet53**	**CSPDarknet53** (CSPNet in Darknet)	**CSPDarkent53** Focus structure	**EELAN**	**RepVGG** and **CSPRepStack**
Neck	**FPN** (Feature Pyramid Network)	**SPP** (Spatial Pyramid Pooling) and **PANet** (Path Aggregation Network)	**PANet**	**PANet**	**RepPAN**
Head	**B x (5 + C)** output layer **B**: No. of bounding boxes **C**: Class score	Same as Yolo v3	Same as Yolo v3	**Lead Head** for final output, **Auxiliary Head** for middle layer outputs	**Decoupled Classification** and **Detection Head**
Loss Function	Binary Cross Entropy	Binary Cross Entropy	Binary Cross Entropy and Logit Loss Function	**BCE with Focal Loss** for Classification, **IoU loss** for Detection	**Varifocal Loss for Classification** and **Distribution Focal Loss for Detection**

Thank you