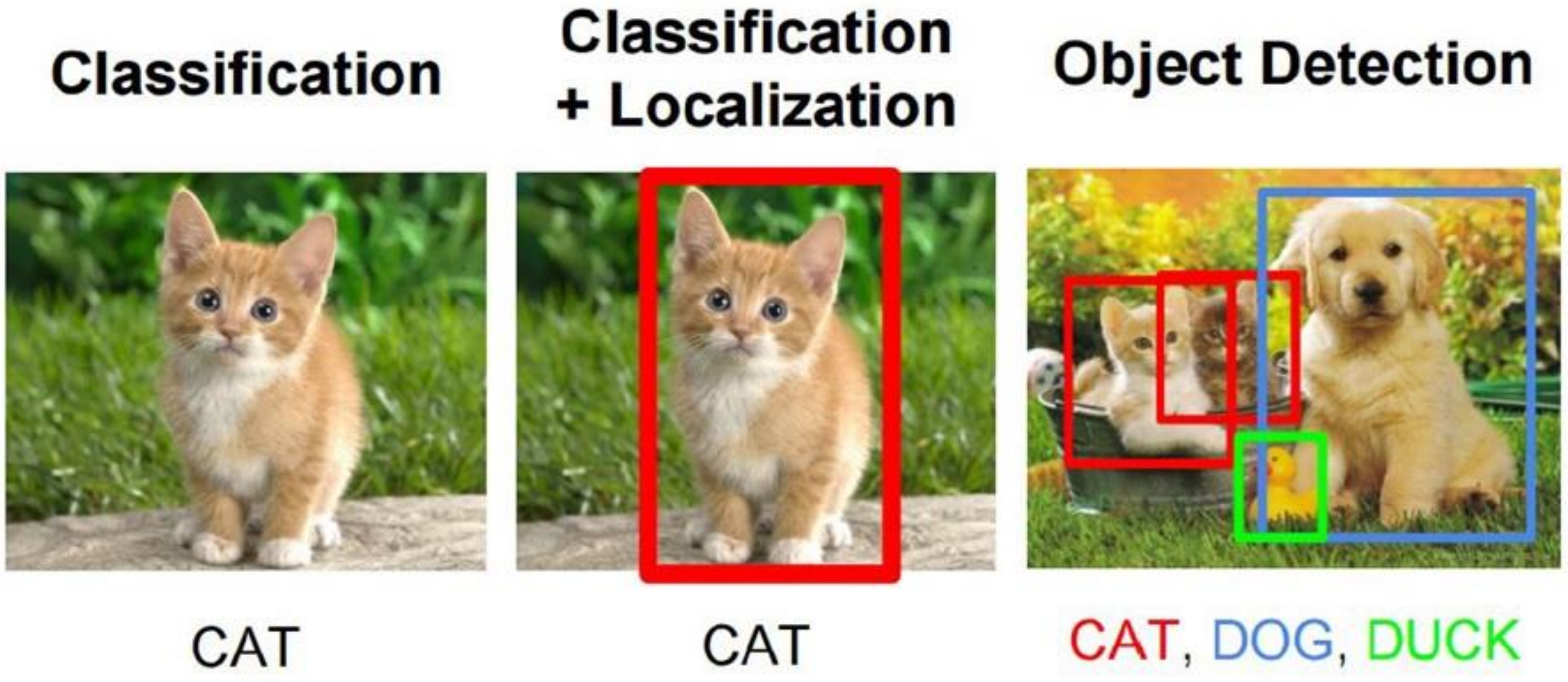
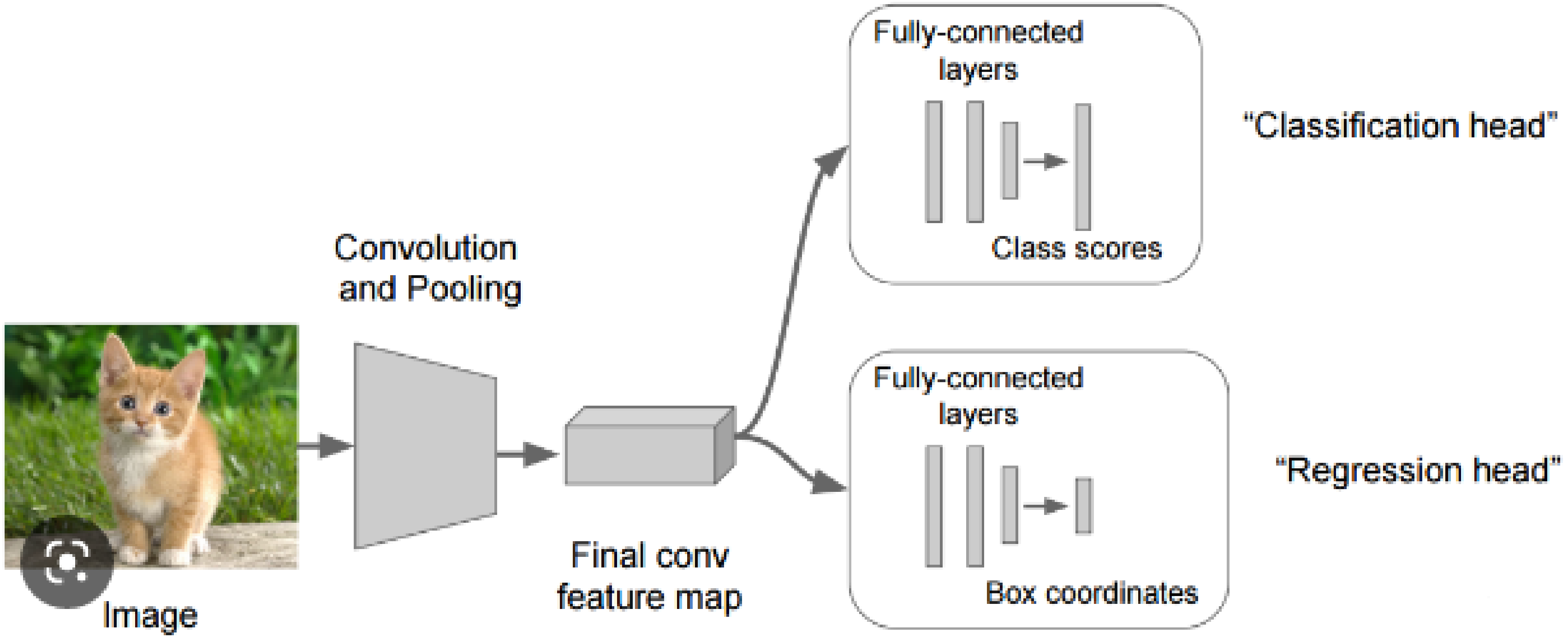
Object Detection

Object Detection and Localization

- Object detection aims for detecting, locating and classifying objects in an image




Convolution
and Pooling
Image
Final conv
feature map
Fully-connected
layers
Class scores
"Classification head"
Fully-connected
layers
Box coordinates
"Regression head"

Object Detection and Localization

- Widely used in computer vision tasks such as:
  - Vehicle detection
  - People counting
  - Number plate recognition
  - Autonomous driving
- Techniques for object detection are generally machine learning or deep learning based

Recent Research

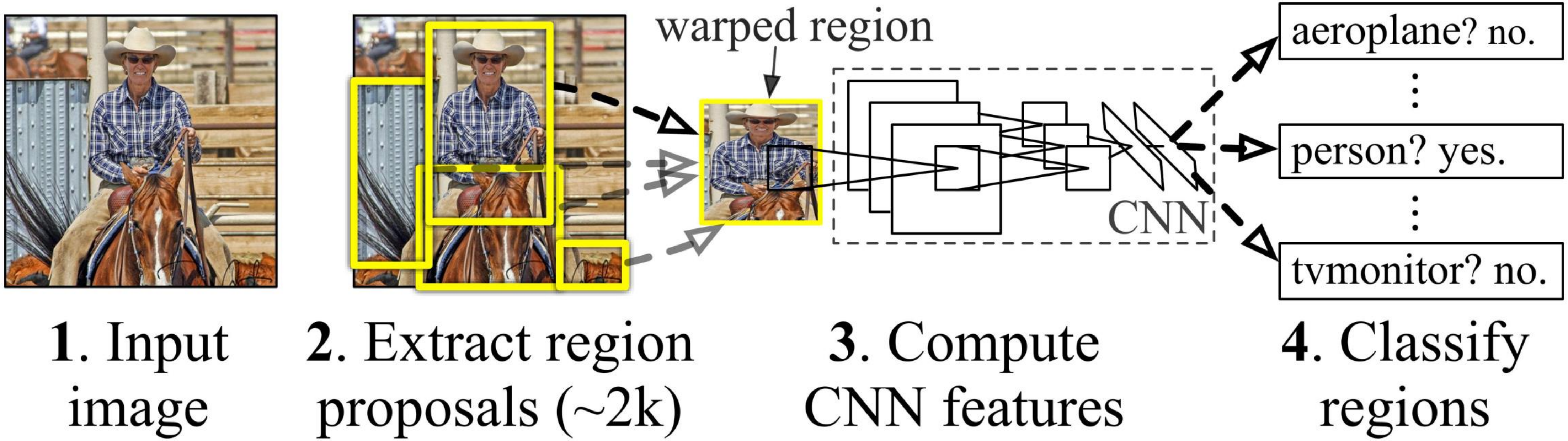
- Region Based Convolutional Network [Girshick et al. 2014]:
  - Used the sliding window approach with Selective Search
  - Still feeds a limited part of the image to the classifier
  - Drawbacks: Large pipeline, slow, too many false positives

- Fast and Faster R-CNN [Gavrilescu et al. 2018]:
  - Optimized parts of pipeline
  - Drawbacks: loses accuracy

R-CNN

- Comprised of three modules:
- **Region Proposal:** Generate and extract category independent region proposals, e.g. candidate bounding boxes
  - Propose candidate regions or bounding boxes of potential objects in image called "*selective search*"
  - Look at image through windows of different sizes and for each size, group together adjacent pixels by texture, colour, intensity to identify objects
- **Feature Extractor:** Extract feature from each candidate region, e.g. using a deep CNN like a pre-trained AlexNet
  - Output of CNN was a 4,096 element vector
- **Classifier:** Classify features as one of known class, e.g. linear SVM classifier

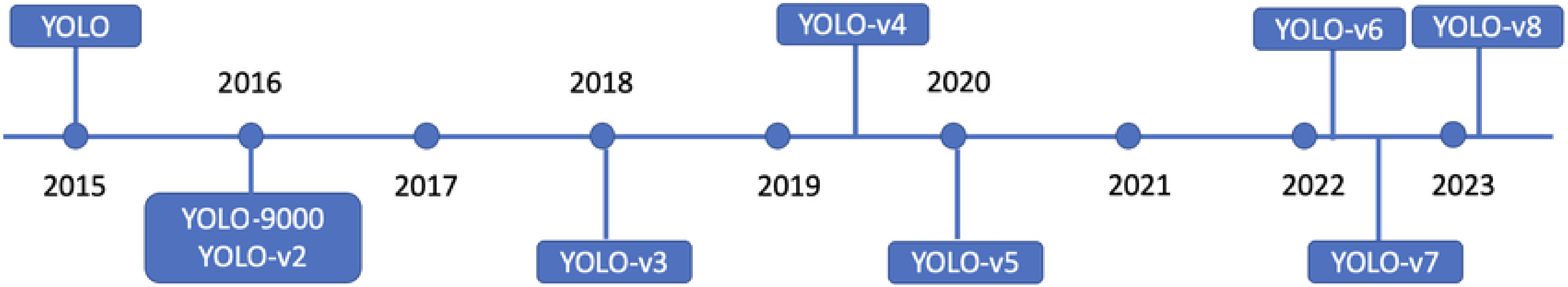
R-CNN: *Regions with CNN features*



YOLO – You Look Only Once

- First described by Joseph Redmon, et al. in the 2015
- State-of-art for real-time object detection algorithm
- Goal: perform object detection with speed and high accuracy
- Looks at whole image at test time: predictions influenced by global context in image
- Applies a single neural network to image:
  - Network divides image into regions and predicts bounding boxes and probabilities for each region
  - Bounding boxes weighted by predicted probabilities

YOLO Timeline



Comparison with State-of-art

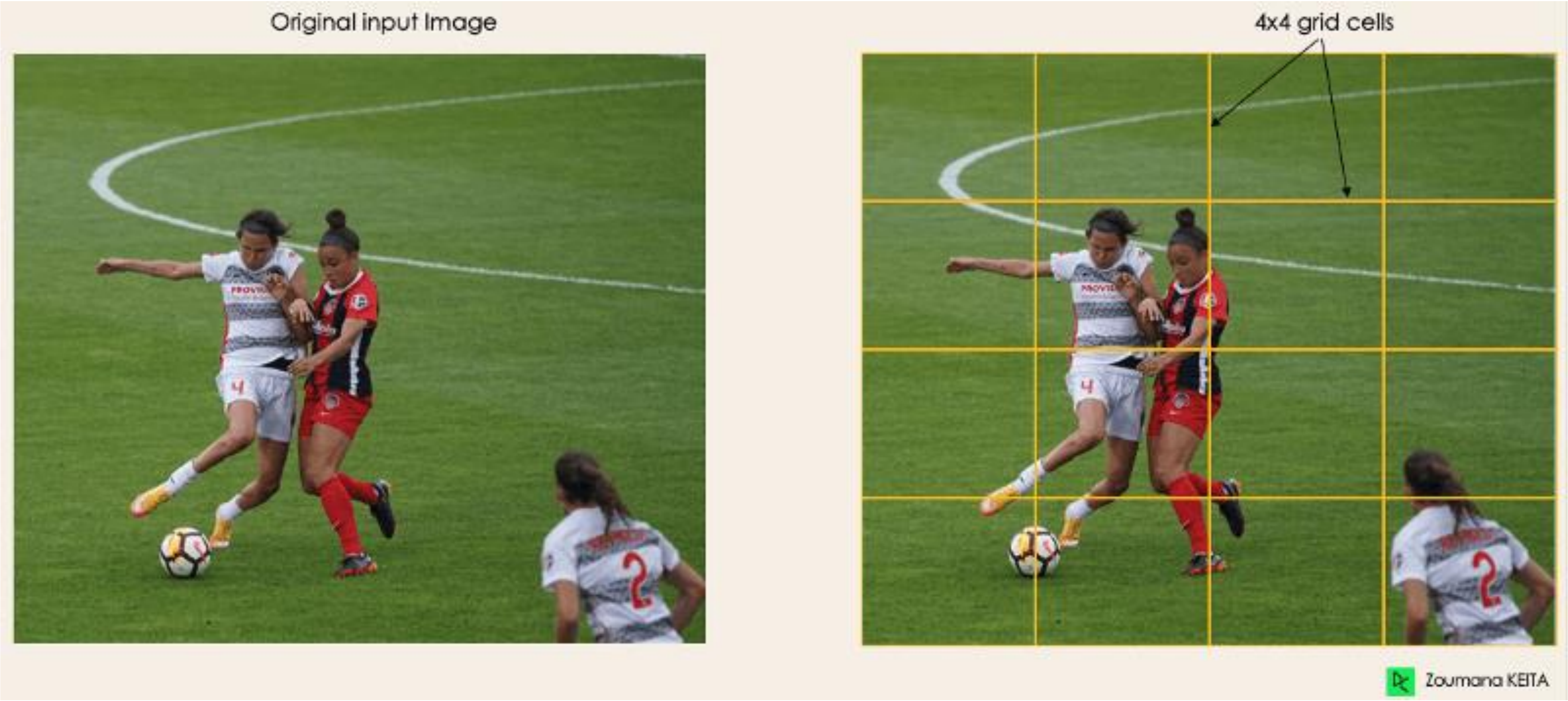
Previous Approaches	YOLO algorithm
Separate models for generating bounding boxes and for classification (more complicated model pipeline)	A single neural network for localization and for classification (less complicated pipeline)
Need to run classification many times (expensive computation)	Need to inference only once (efficient computation)
Looks at limited part of the image (lacks contextual information for detection)	Looks at the entire image each time leading to less false positives (has contextual information for detection)

Algorithm

- Algorithm works on following four approaches:
  - Residual blocks
  - Bounding box regression
  - Intersection Over Unions (IOU)
  - Non-Maximum Suppression

Residual Blocks

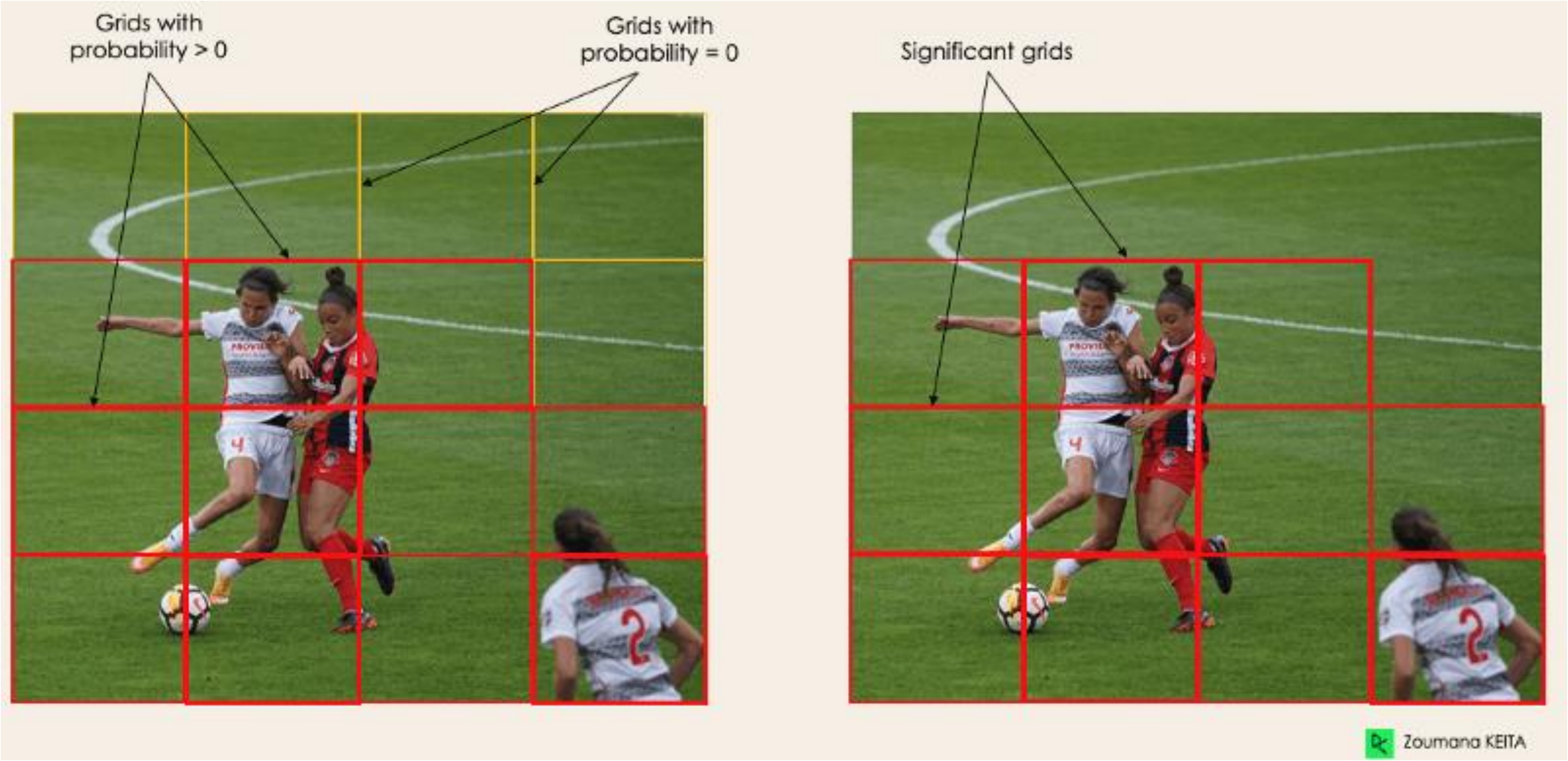
- Divide original image into S*S grid cells of equal shape
- Each cell in grid responsible for localizing and predicting class of object it covers, along with the probability/confidence value
  - If center of an object falls into a grid cell, that grid cell is responsible for detecting that object


Original input Image
4x4 grid cells
Zoumana KEITA

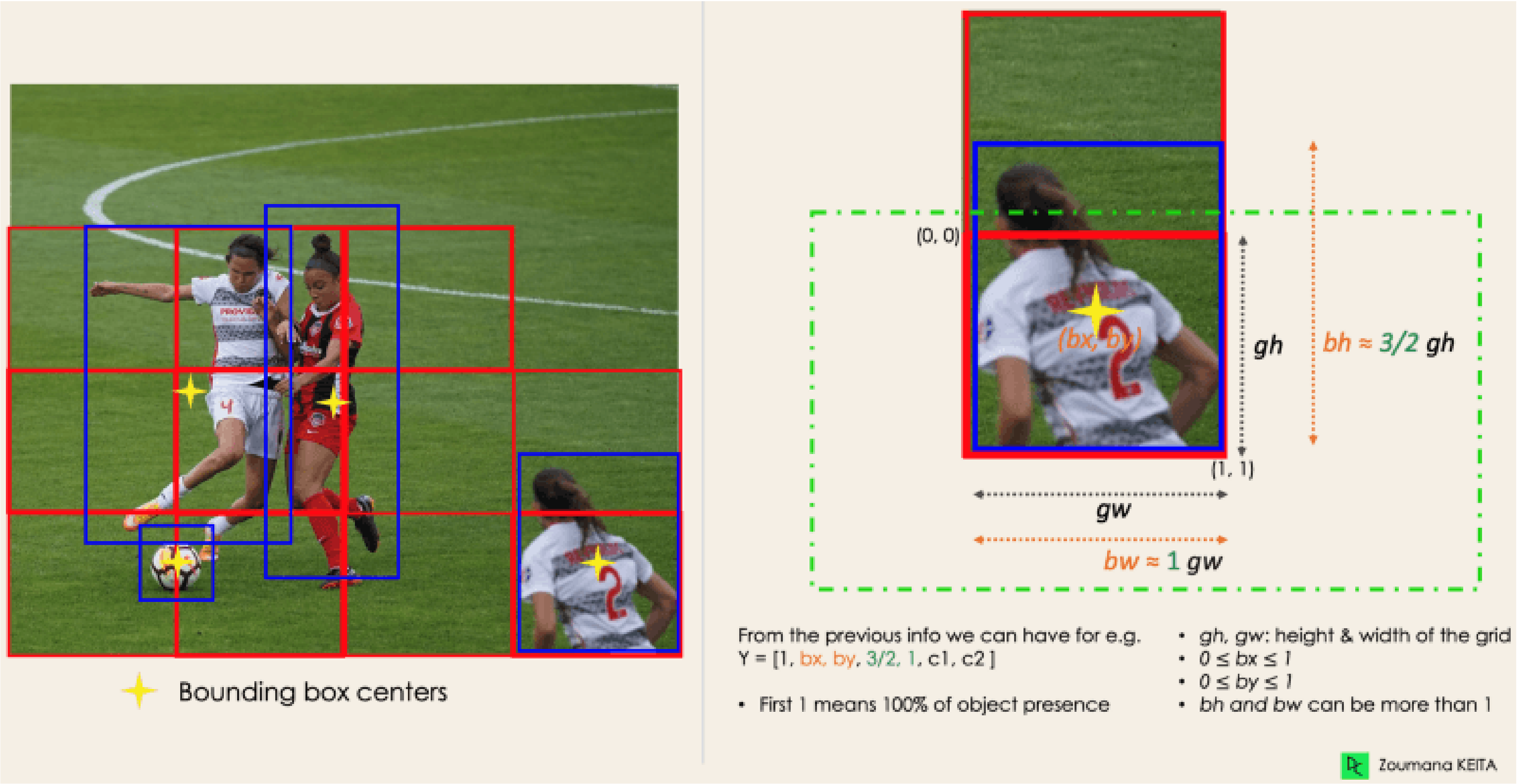
Bounding-box Regression

- Determine bounding boxes - correspond to rectangles highlighting all objects in image
- For each grid square, generate *B* bounding boxes
- For each bounding box, following predictions are made:
  - *pc*: Confidence score - Probability that bounding box has an object
  - *bx*, *by*: center coordinates of bounding box w.r.t. enveloping grid cell
  - *bw, bh*: width and height of bounding box w.r.t. enveloping grid cell
  - *c*1, *c*2: class of object in bounding box
- YOLO determines attributes of bounding boxes using a single regression module: *Y* is final vector representation for each bounding box

$$Y = [pc, bx, by, bh, bw, c1, c2]$$

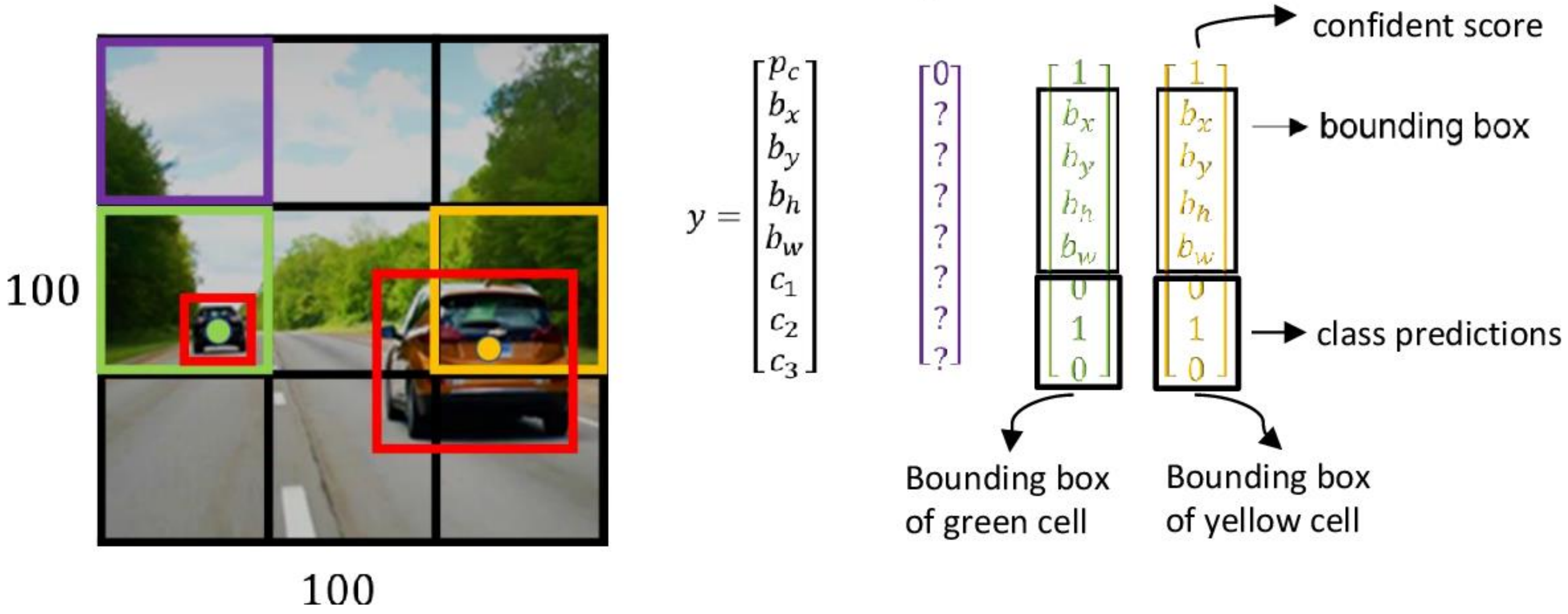


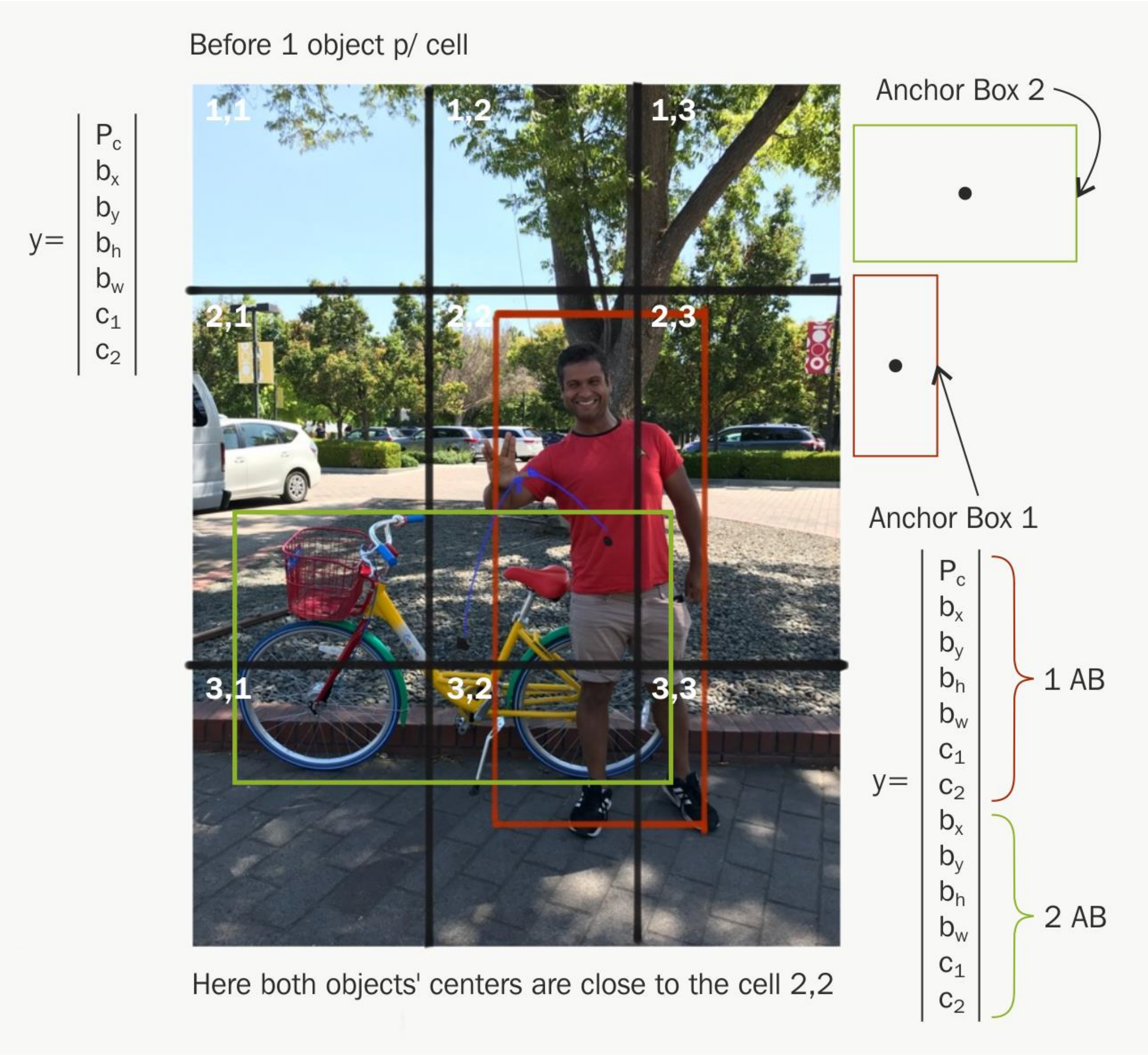
Ex., all grids in red will have a probability score higher than zero. Image on right is simplified version since probability of each yellow cell is zero (insignificant)



c1, c2: correspond to two classes Player and Ball; Can have as many classes as your use case requires

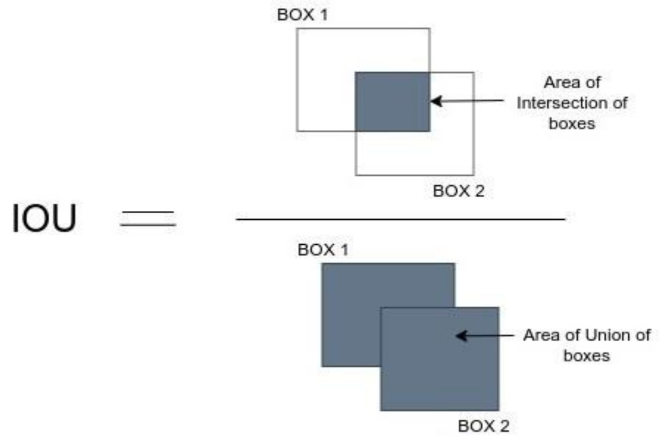
Labels for training for each grid cell:

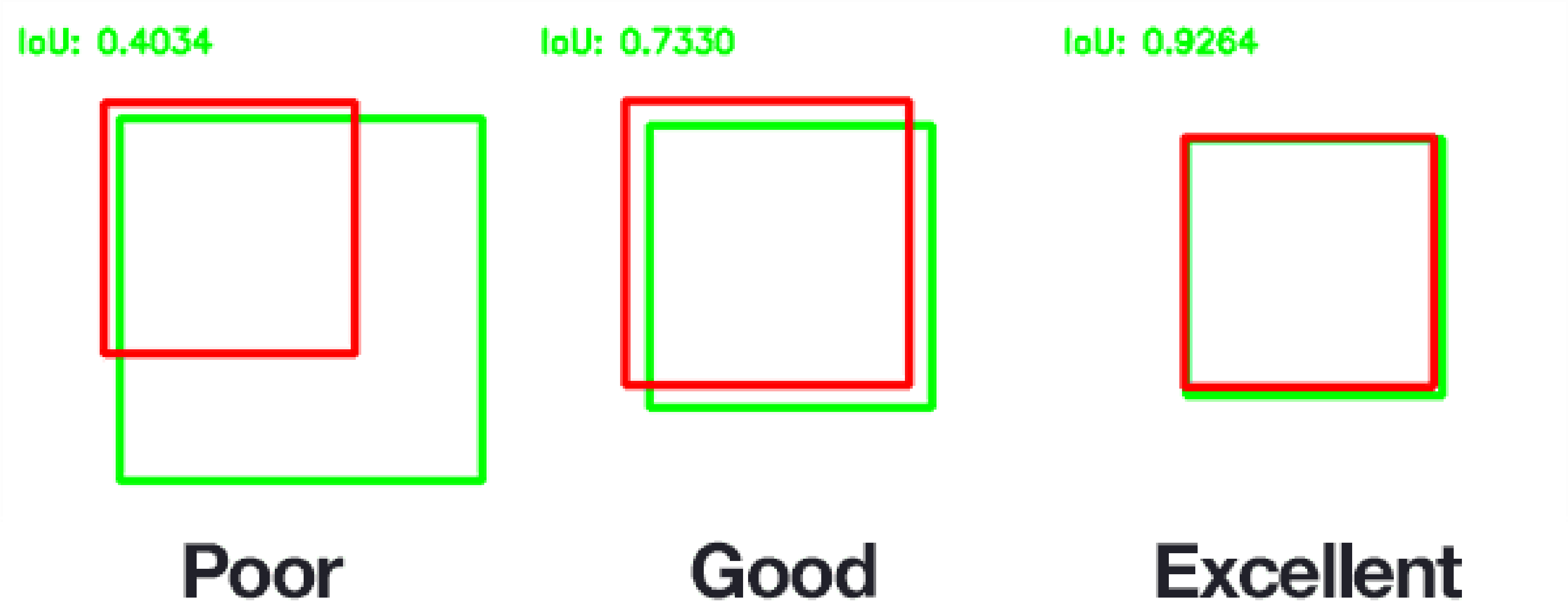



Before 1 object p/ cell
y= Pc bx by bh bw c1 c2
1,1
1,2
1,3
2,1
2,2
2,3
3,1
3,2
3,3
Anchor Box 2
Anchor Box 1
y= Pc bx by bh bw c1 c2 bx by bh bw c1 c2
1 AB
2 AB
Here both objects' centers are close to the cell 2,2

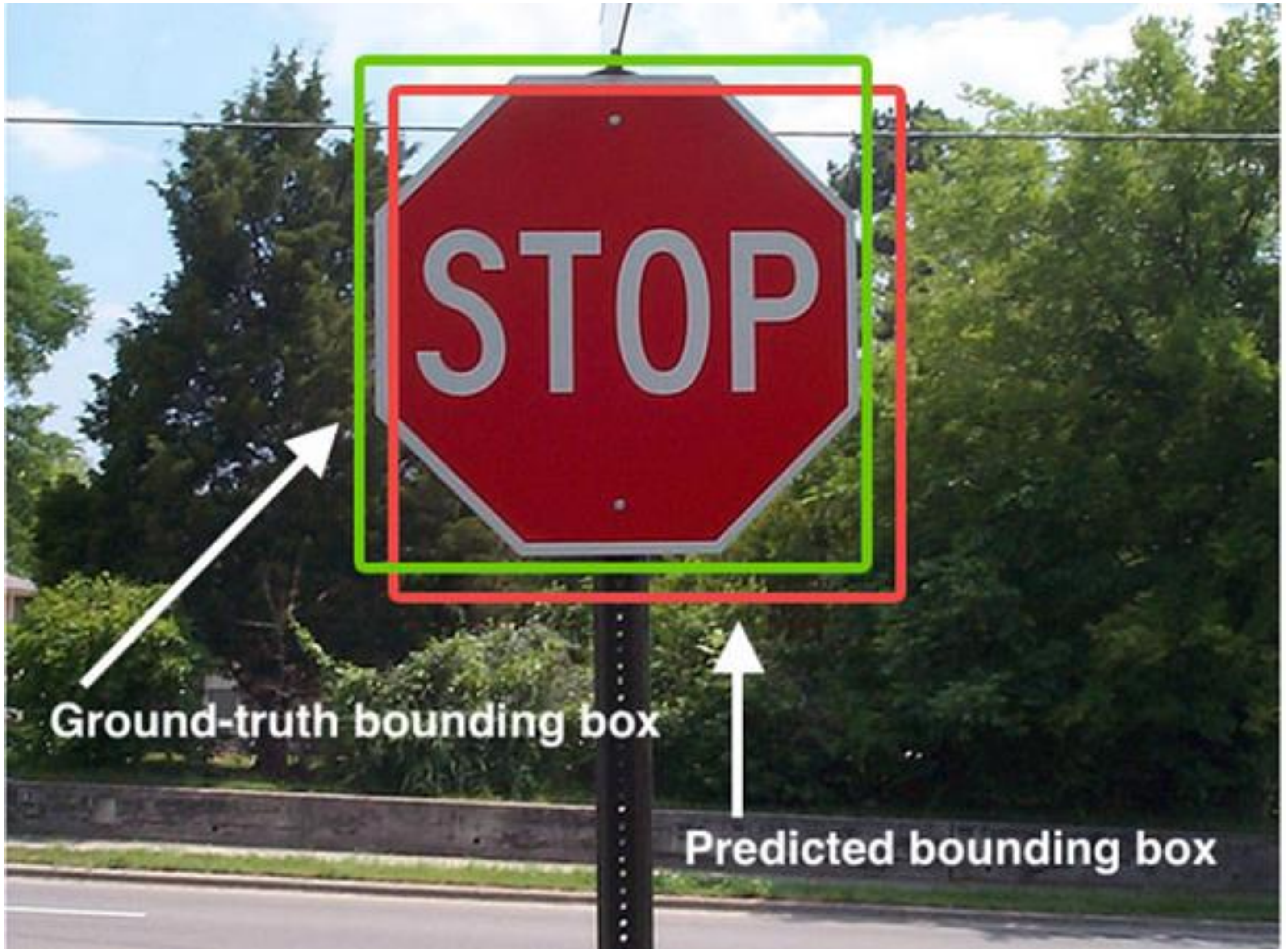
Intersection Over Unions (IoU)

- Used to describe extent of overlap of two boxes
  - Greater the region of overlap, greater the IoU
- Due to varying parameters of model, a complete and total match between predicted and ground-truth bounding boxes is unrealistic
- Need to define an evaluation metric that *rewards* predicted bounding boxes for heavily overlapping with the ground-truth





Predicted bounding boxes that heavily overlap with ground-truth bounding boxes have higher scores than those with less overlap; Makes IoU an excellent metric for evaluating custom object detector

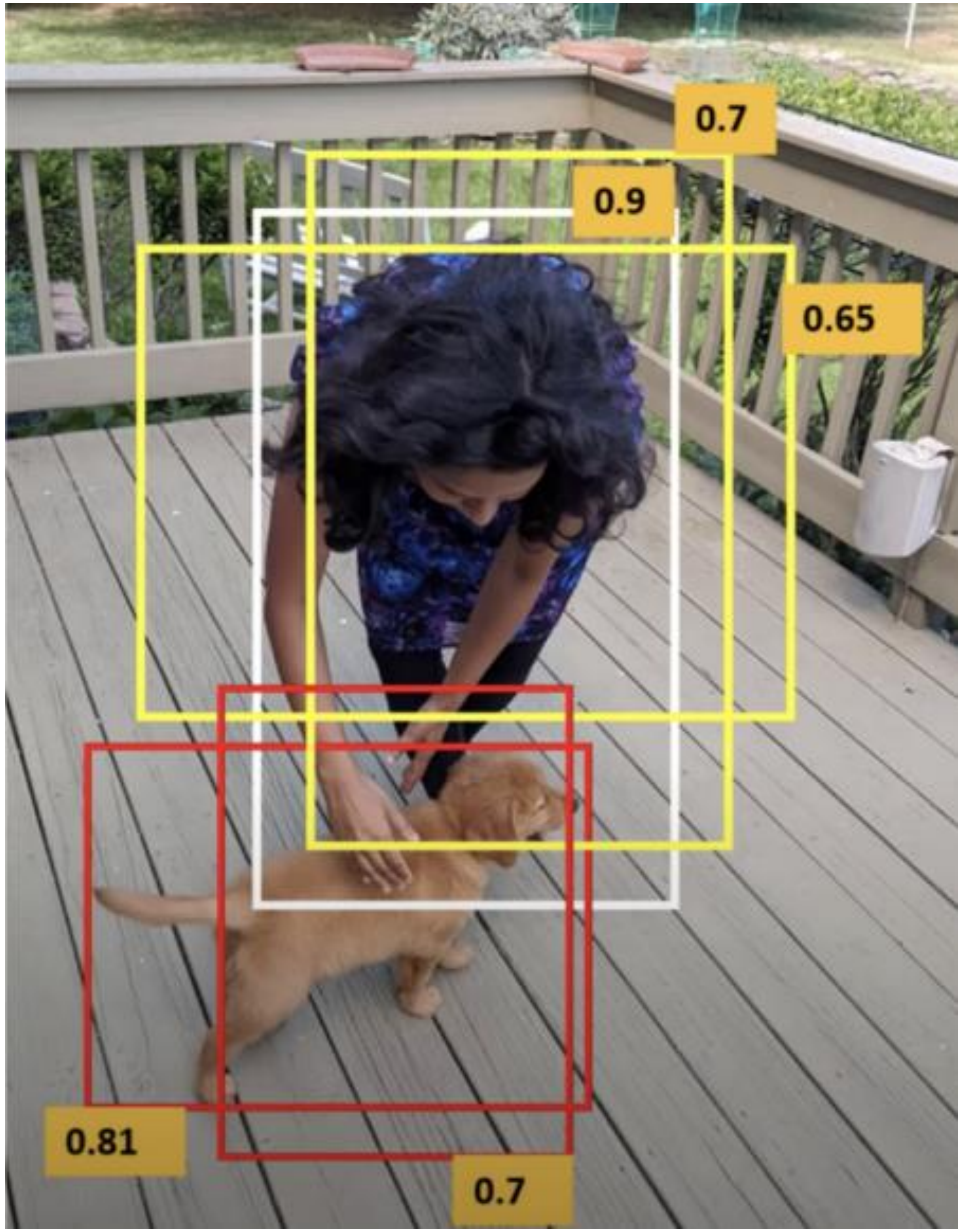

STOP
Ground-truth bounding box
Predicted bounding box

Non-Max Suppression (NMS)

- An object can have multiple boxes with IOU beyond threshold
  - These may overlap or be located at different positions, but all represent same object
  - Leaving all those boxes might include noise
- NMS used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in image
  - Keep only boxes with highest probability score of detection
- To remove duplicates:
  - Select the box with highest probability and output that as a prediction
  - Eliminate any bounding box with IoU > 0.5 (or any threshold value) with the predicted output

Non-Max Suppression (NMS)

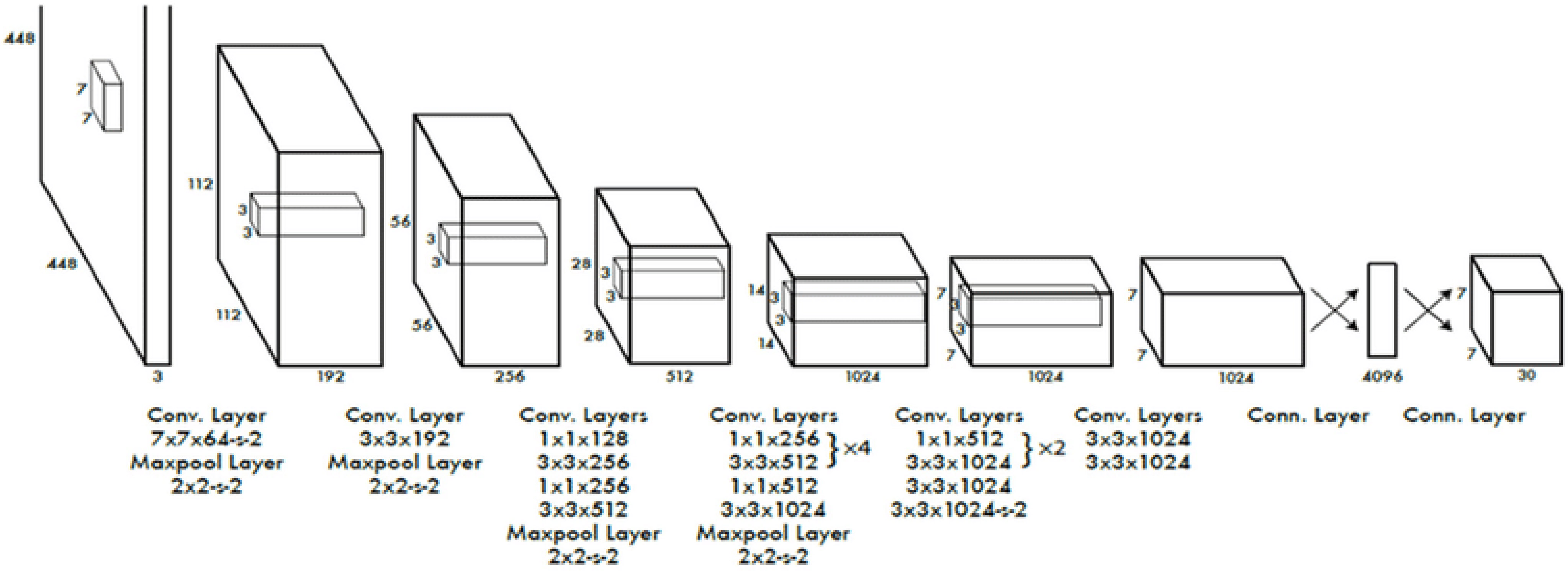
- Helps to remove duplicate bounding boxes for same object
  - Sort all predictions/objects in descending order of their confidence
  - If two bounding boxes are pointing to same object, their IoU would be high
  - In this case, choose box with higher confidence
  - If IoU is very low, this would possibly mean that the two boxes point to different objects of same class
- While training model, can choose a suitable minimum IoU score needed for a predicted box to be regarded as an accurate positive detection


0.7
0.9
0.65
0.81
0.7

YOLO

- YOLO is a regression algorithm
- Trained on PASCAL VOC dataset
  - Can detect 20 different classes
- Input *X* an image of *width* * *height* * *RGB* values
- *Y* is a tensor of size $S * S * (B * (5 + C))$
  - $B*(5+C)$ represents the 5 predictions and predicted class distribution for each bounding box of a grid block
- Ex.: Image size $416*416*3$ as input
  - Parameters: $S = 19$, $B = 3$, $C = 80$
  - Output is $S*S*(B*(5+C)) = 19 * 19 * (3 * (5 + 80)) = 19 * 19 * 255$

YOLO Architecture



Overall 24 convolutional layers, four max-pooling layers, and two fully connected layers

YOLO: Architecture

- First 20 convolution layers of model pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer
- This pre-trained model converted to perform detection
  - Adding convolution and connected layers to a pre-trained network improves performance
- Final fully connected layer predicts both class probabilities and bounding box coordinates

YOLO Architecture

- Resizes input image into 448*448 before going through convolutional network
- A 1*1 convolution is first applied to reduce number of channels, followed by a 3x3 convolution to generate a cuboidal output
- Activation function under the hood is ReLU, except for final layer, which uses a linear activation function
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting

YOLO: Limitations

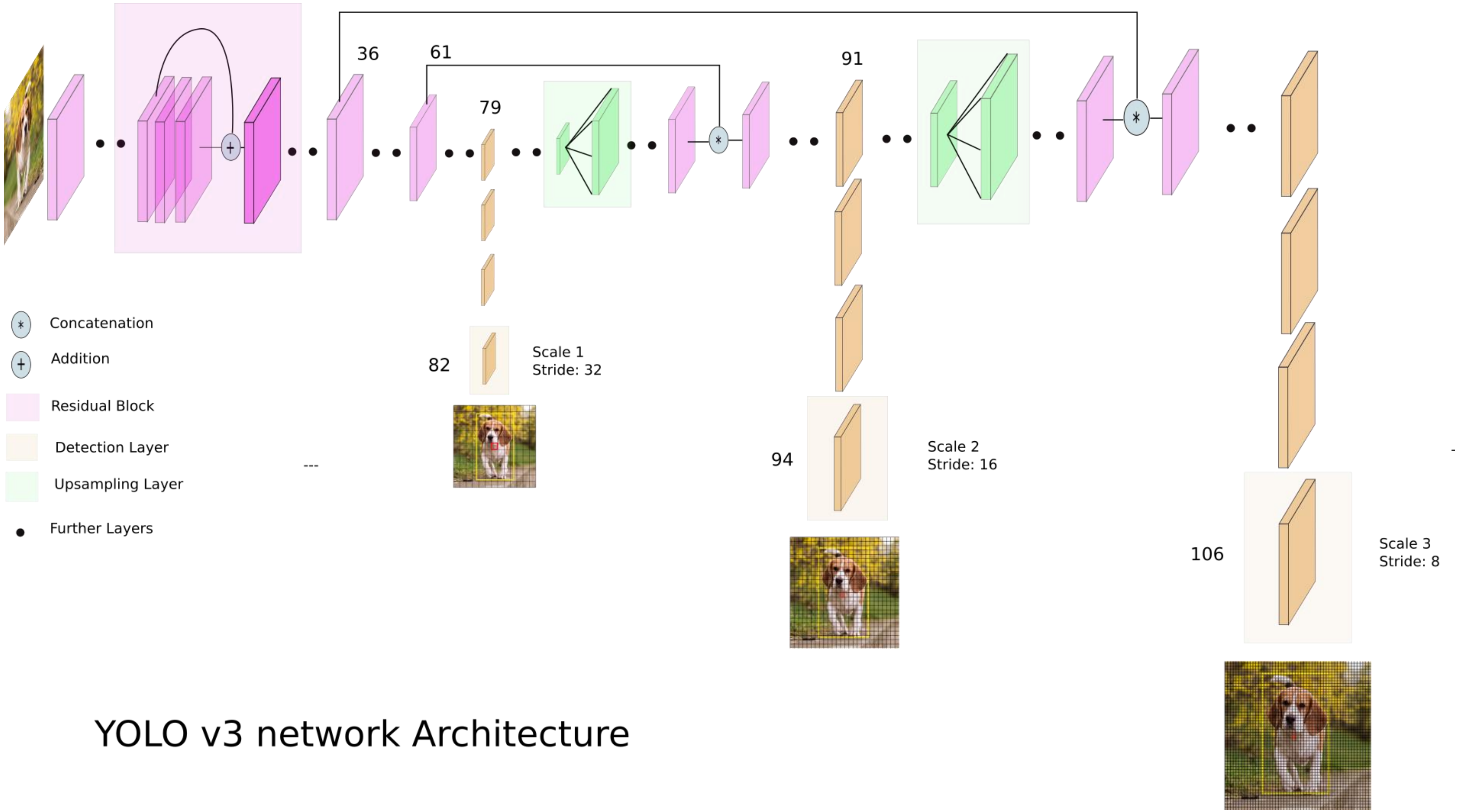
- Requires data to be labeled with bounding boxes, hard to collect for many classes
- May not be ideal for using niche models where large datasets can be hard to obtain

Models:

- YOLO v2: introduced in 2016
  - Uses a different CNN backbone called Darknet-19 - variant of VGGNet architecture
- Main improvements:
  - Use of anchor boxes - set of predefined bounding boxes of different aspect ratios and scales
  - Use of batch normalization - helps to improve accuracy and stability of model
  - Uses multi-scale training strategy - involves training model on images at multiple scales and averaging predictions - helps to improve detection performance of small objects
  - Introduces new loss function - based on sum of squared errors between predicted and ground truth bounding boxes and class probabilities

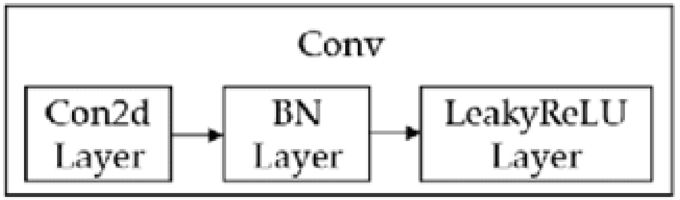
YOLO v3

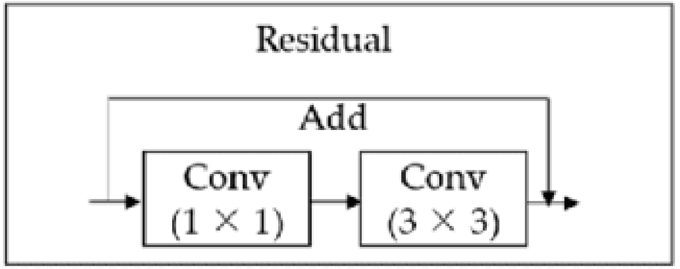
- Main improvements: use of new CNN architecture called Darknet-53
  - Variant of ResNet architecture, designed specifically for object detection tasks
- Uses anchor boxes with different scales and aspect ratios: In YOLO v2, anchor boxes were all the same size
- Introduces concept of "feature pyramid networks" (FPN)
  - FPNs are a CNN architecture used to detect objects at multiple scales
  - They construct a pyramid of feature maps - each level of pyramid being used to detect objects at a different scale; helps to improve detection performance on small objects
- Can handle a wider range of object sizes and aspect ratios



YOLO v3 network Architecture

Layer	Filters size	Repeat	Output size
Image			416 × 416
Conv	32 3 × 3/1	1	416 × 416
Conv	64 3 × 3/2	1	208 × 208
Conv Conv Residual	32 1 × 1/1 64 3 × 3/1	Conv Conv Residual × 1	208 × 208 208 × 208 208 × 208
Conv	128 3 × 3/2	1	104 × 104
Conv Conv Residual	64 1 × 1/1 128 3 × 3/1	Conv Conv Residual × 2	104 × 104 104 × 104 104 × 104
Conv	256 3 × 3/2	1	52 × 52
Conv Conv Residual	128 1 × 1/1 256 3 × 3/1	Conv Conv Residual × 8	52 × 52 52 × 52 52 × 52
Conv	512 3 × 3/2	1	26 × 26
Conv Conv Residual	256 1 × 1/1 512 3 × 3/1	Conv Conv Residual × 8	26 × 26 26 × 26 26 × 26
Conv	1024 3 × 3/2	1	13 × 13
Conv Conv Residual	512 1 × 1/1 1024 3 × 3/1	Conv Conv Residual × 4	13 × 13 13 × 13 13 × 13





YOLO v4

- Use of a new CNN architecture called CSPNet
  - Stands for "Cross Stage Partial Network"
  - Variant of the ResNet architecture designed for object detection tasks
  - Has a relatively shallow structure, with only 54 convolutional layers
- Introduces a new term called "GHM loss”
  - Variant of focal loss function
  - Designed to improve model’s performance on imbalanced datasets
- Improves architecture of FPNs used in YOLO v3

YOLO v5

- Uses more complex architecture called EfficientDet - based on EfficientNet network architecture
  - Allows to achieve higher accuracy and better generalization to object categories
- Trained on a larger and more diverse dataset called D5 - includes a total of 600 object categories - YOLO trained on PASCAL VOC dataset (20 classes)
- Uses new method for generating anchor boxes - "dynamic anchor boxes"
  - Involves using clustering algorithm to group ground truth bounding boxes into clusters and then using centroids of clusters as anchor boxes
  - Allows anchor boxes to be more aligned with detected objects' size and shape

YOLO v5

- Also introduces concept of "spatial pyramid pooling" (SPP) - type of pooling layer used to reduce spatial resolution of feature maps
  - Allows the model to see the objects at multiple scales
- YOLO v4 also uses SPP, but YOLO v5 includes several improvements to SPP architecture that allow it to achieve better results

YOLO v6

- Uses a variant of the EfficientNet architecture called EfficientNet-L2
  - More efficient architecture than EfficientDet, with fewer parameters and a higher computational efficiency
- Can achieve state-of-the-art results on various object detection benchmarks
- Introduces a new method for generating anchor boxes, called "dense anchor boxes"

YOLO v7

- Main improvements is use of anchor boxes
  - Uses nine anchor boxes
  - Allows to detect a wider range of object shapes and sizes compared to previous versions
  - Reduces number of false positives
- Has a higher resolution than previous versions
  - Processes images at a resolution of 608*608 pixels, which is higher than the 416*416 resolution used in YOLO v3
  - Allows to detect smaller objects and to have a higher accuracy overall

YOLO v8

- New API that will make training and inference much easier on both CPU and GPU devices and the framework will support previous YOLO versions
- New features and improved performance over its predecessors

Conclusion

- Object detection is the problem of detecting multiple objects in an image
- Almost real time object detection can make highly responsive robot systems without complex sensors
- Prior work relies on a large architecture with numerous parts to optimize
- YOLO proposes a unified architecture, which does all the tasks in one model and by one inference over the entire image
- They show enormous speed improvement and show that they can beat most other prior work in terms of mAPs

Example Code

- Google CoLab

References

- [Gavrilescu et al., 2018] Gavrilescu, R., Zet, C., Fos,al˘au, C., Skoczylas, M., and Cotovanu, D. (2018). Faster r-cnn:an approach to real-time object detection. In 2018 International Conference and Exposition on Electrical And Power Engineering (EPE), pages 0165–0168.
- [Girshick et al., 2016] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1):142–158.
- [Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, Computer Vision – ECCV 2016, pages 21–37, Cham. Springer International Publishing.
- [Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. pages 779–788.