DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

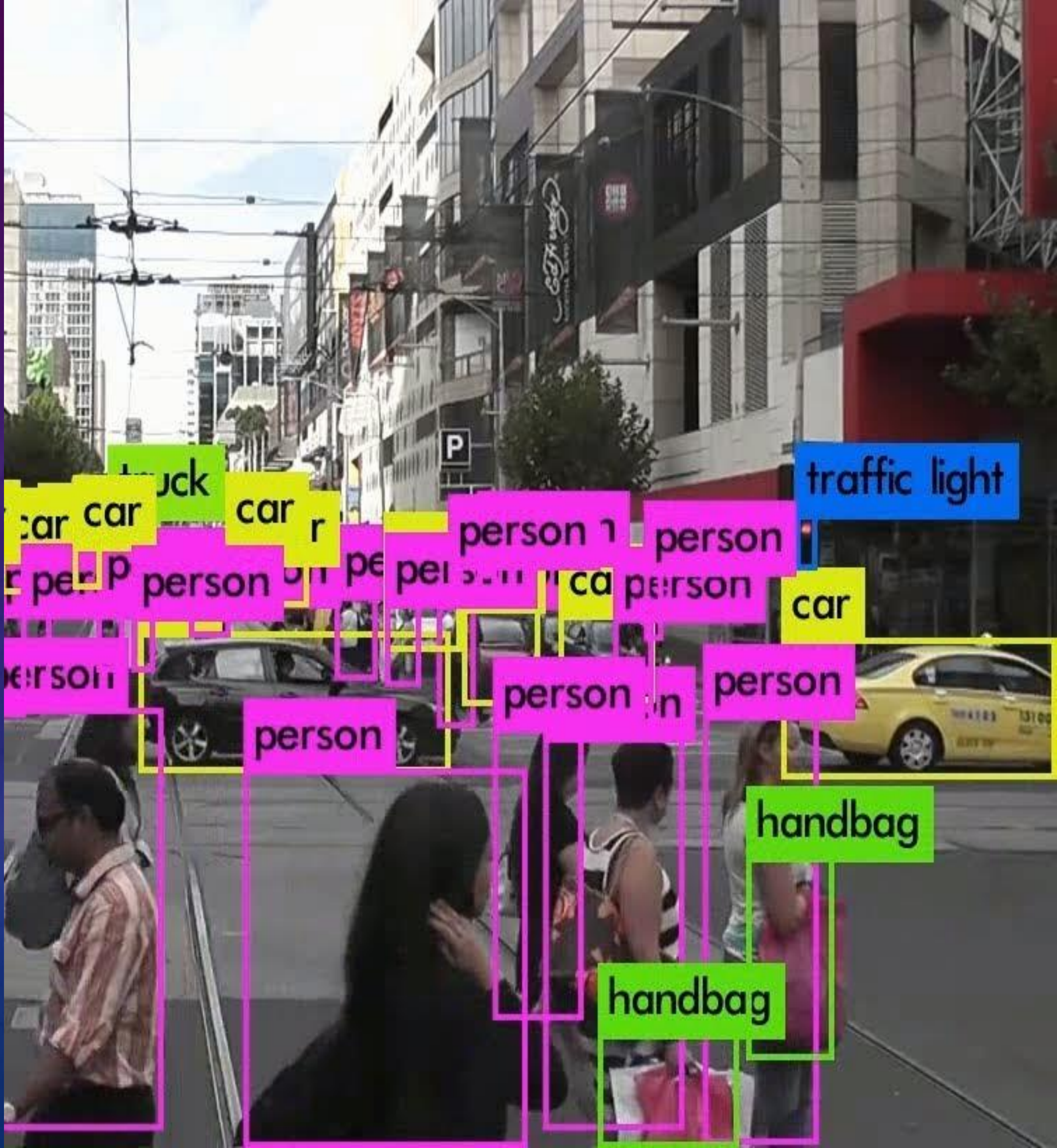
CH5

OBJECT DETECTION WITH YOLO

徐繼聖

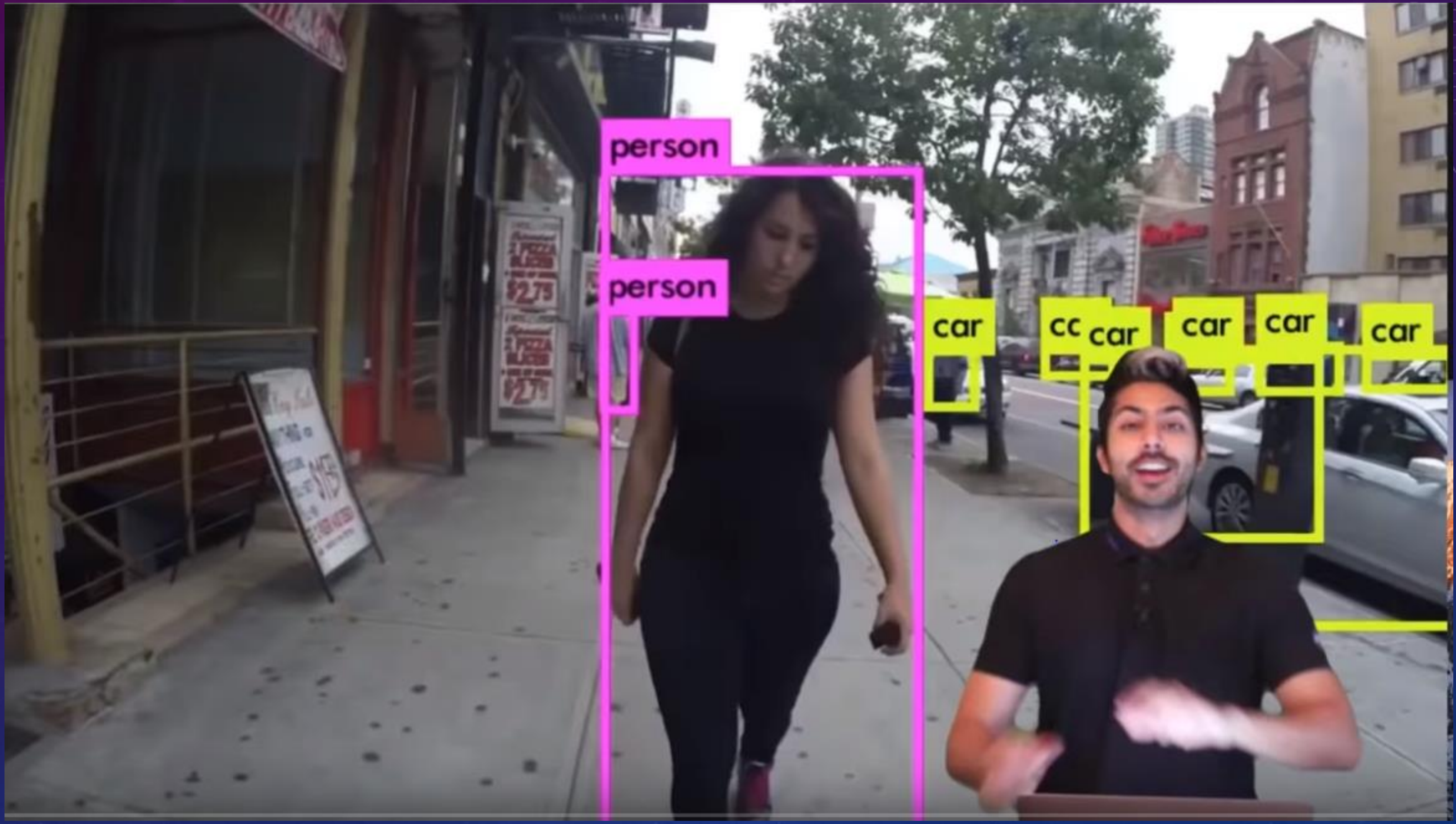
Gee-Sern Jison Hsu

National Taiwan University of Science and Technology



Contents

YOLO Object Detection (TensorFlow tutorial)



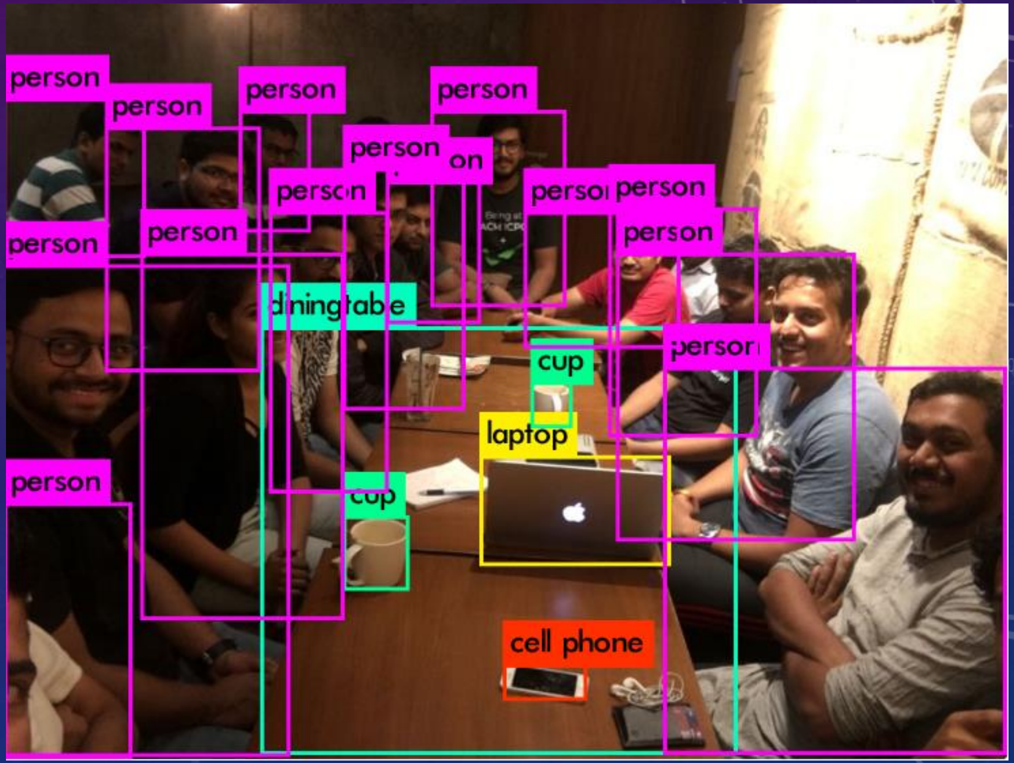
https://www.youtube.com/watch?v=4eIBisqx9_g [21:50]Code tutorial starts at[17:06]

You Only Look Once: Unified, Real-Time Object Detection

https://www.youtube.com/watch?v=NM6lrxy0bxs&feature=youtu.be&ab_channel=ComputerVisionFoundationVideos[13:06]

What is YOLO ?

- YOLO is an object detection algorithm, able to detect multiple objects by creating a labeled bounding box for each object.
- YOLO combines the two stages for detection and classification into a single stage.
- YOLO brings a unified neural network architecture to the table, single architecture which does bounding box prediction and gives class probabilities.



YOLO History

- YOLOv1:

  YOLO v1 was introduced in May 2016 by Joseph Redmon with the paper "You Only Look Once: Unified, Real-Time Object Detection." (CVPR 2016) This was one of the biggest evolutions in real-time object detection.

- YOLOv2:

  In December 2016, Joseph introduced another version of YOLO with paper "YOLO9000: Better, Faster, Stronger." (CVPR 2017) it was also known as YOLO 9000.

- YOLOv3:

  After a year in April 2018, the most popular and stable version of YOLO was introduced. Joseph had a partner this time and they released YOLOv3 with paper "YOLOv3: An Incremental Improvement". (CVPR2018)

https://medium.com/towards-artificial-intelligence/yolo-v5-is-here-custom-object-detection-tutorial-with-yolo-v5-12666ee1774e

YOLO History

- YOLOv4:

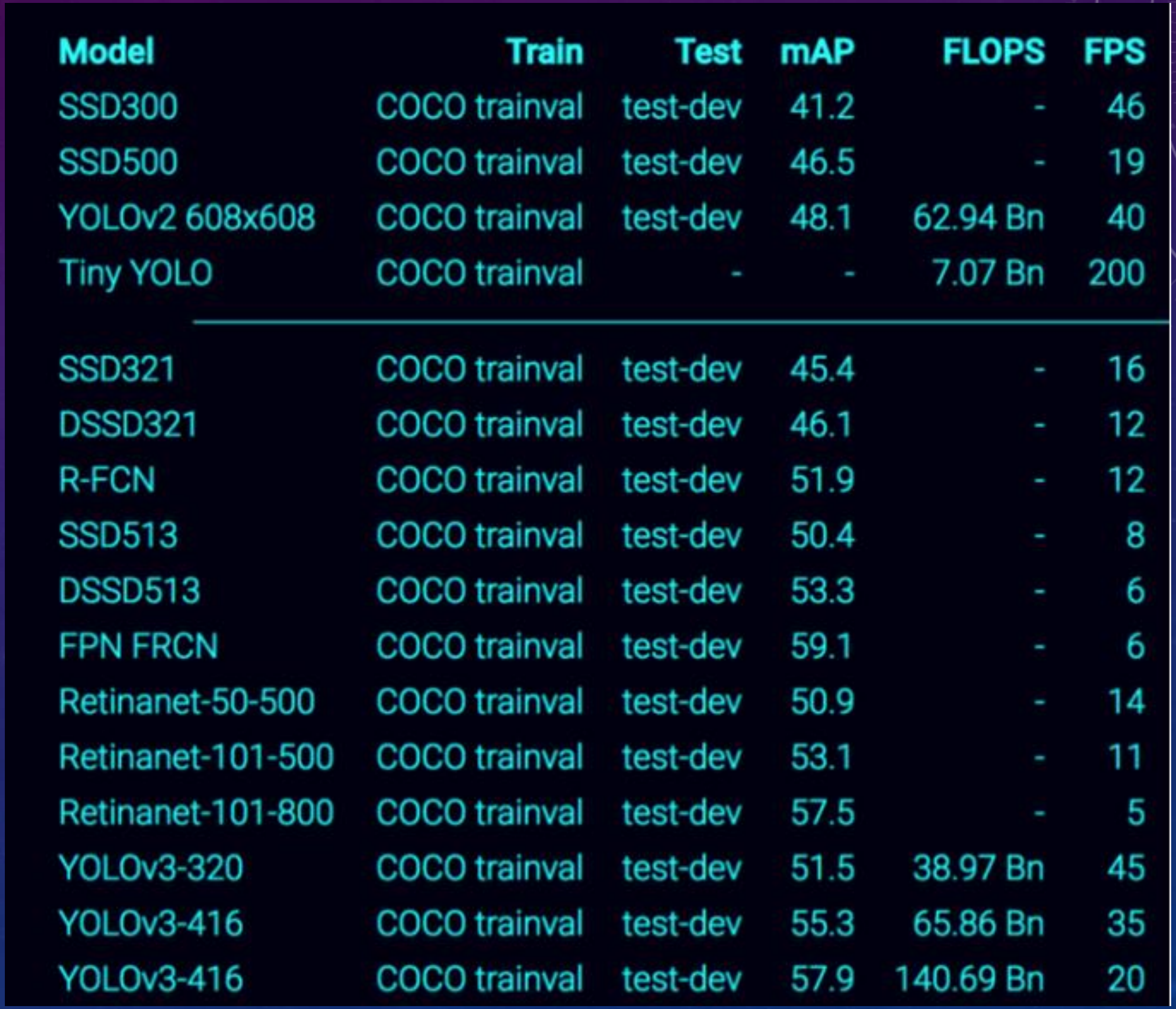
In April 2020, Alexey Bochkovskiy introduced YOLOv4 with the paper "YOLOv4: Optimal Speed and Accuracy of Object Detection" Alexey is not the official author of previous versions of YOLO but Joseph and Ali took a step back from YOLO someone has to handle the era. YOLOv4 was introduced with some astounding new things, It outperformed YOLOv3 with a high margin and also has a significant amount of average precision when compared to EfficientDet Family.

- Features of YOLOv4 include:
  - Novel Backbone: CSPDarknet53 with Spatial pyramid pooling
  - PANet Path-Aggregation Network
  - YOLOv3 detection head

https://medium.com/towards-artificial-intelligence/yolo-v5-is-here-custom-object-detection-tutorial-with-yolo-v5-12666ee1774e

How good is YOLO in comparison with others?

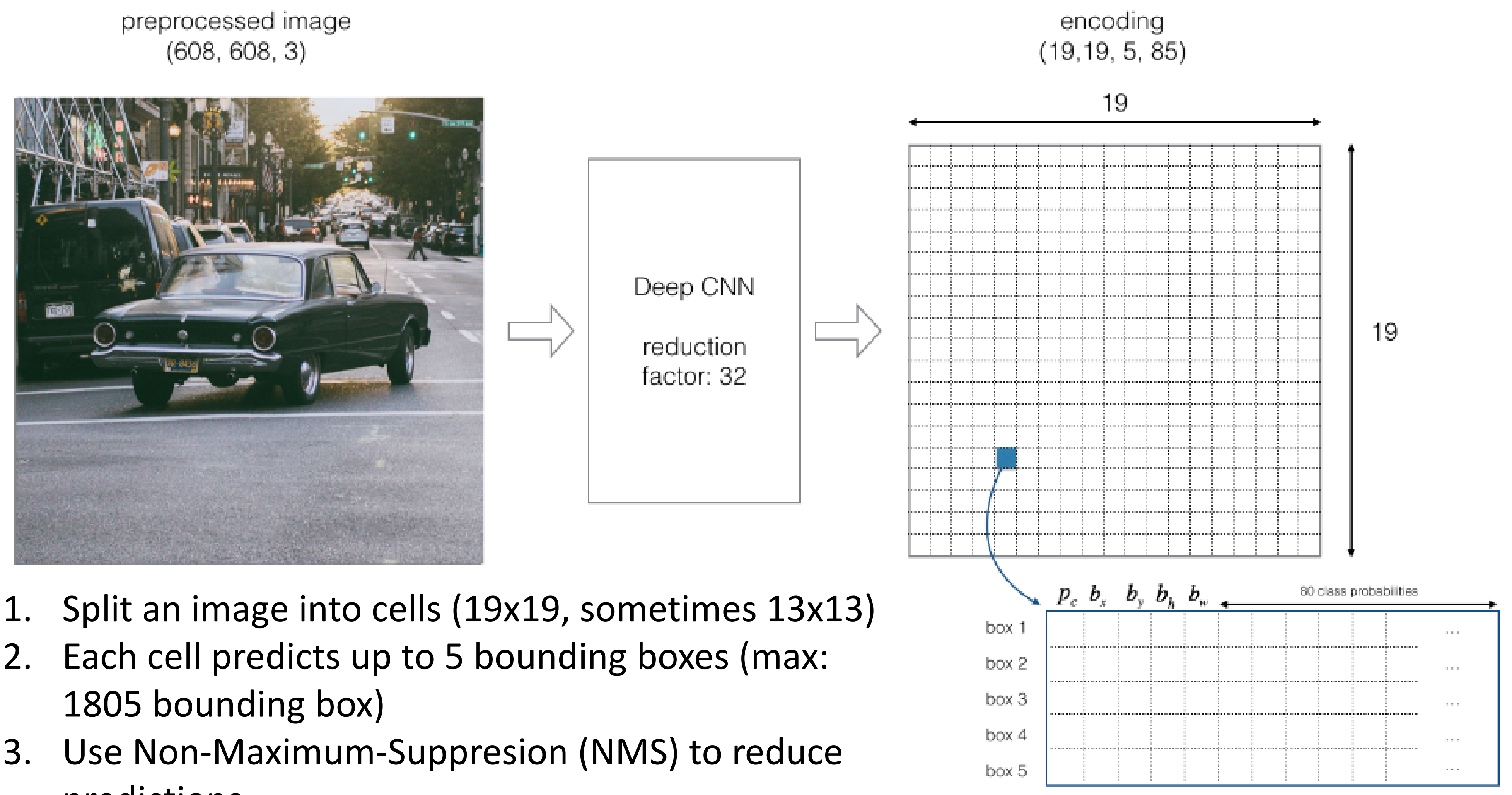
- YOLO is famous for its Real-time capabilities and should be in the repertoire of every AI engineer.



Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	-	-	7.07 Bn	200
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-416	COCO trainval	test-dev	57.9	140.69 Bn	20

https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

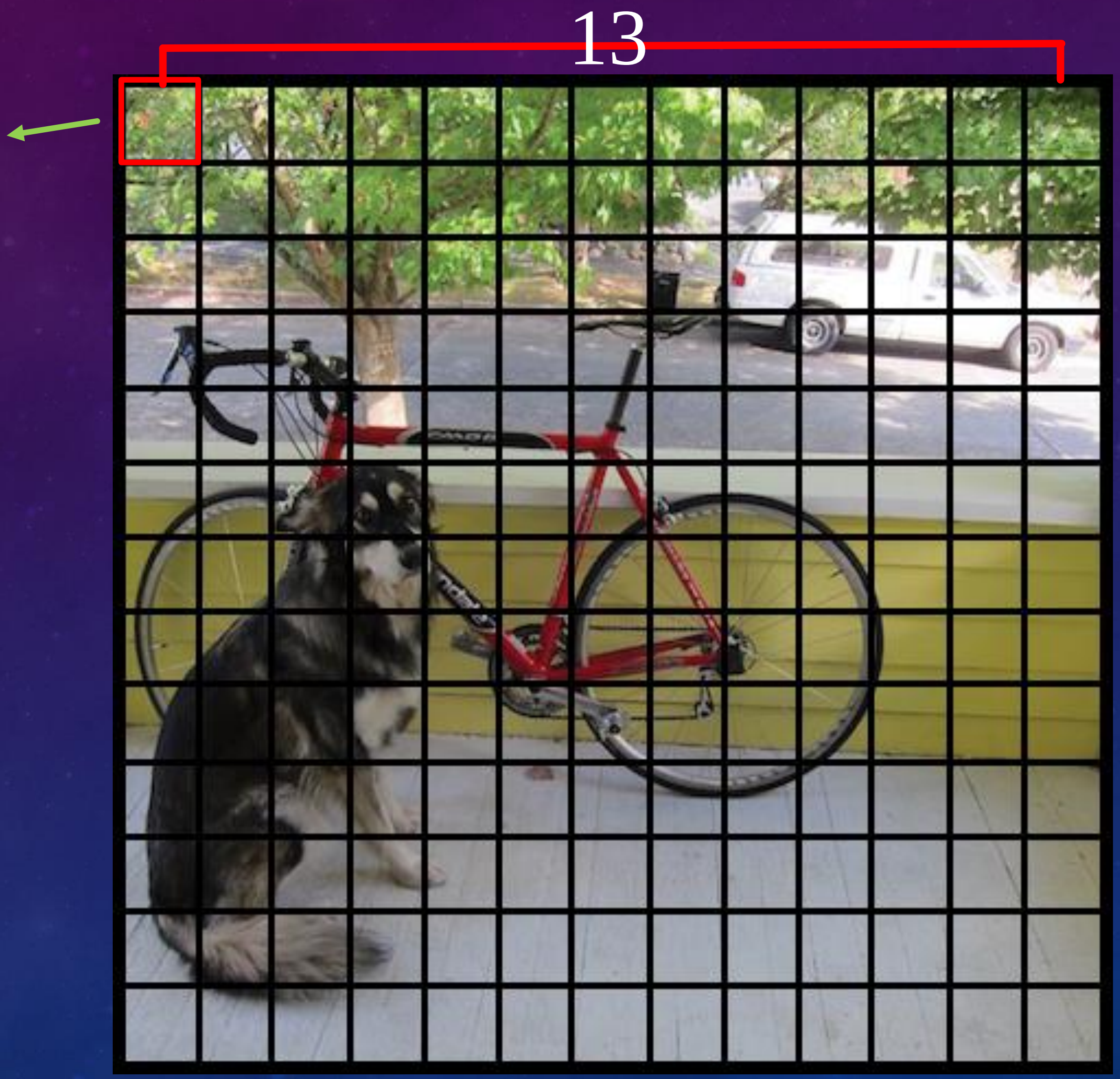
Overview of general YOLO strategy



1. Split an image into cells (19x19, sometimes 13x13)
2. Each cell predicts up to 5 bounding boxes (max: 1805 bounding box)
3. Use Non-Maximum-Suppresion (NMS) to reduce predictions

https://appsilon.com/object-detection-yolo-algorithm/

Grid Cell

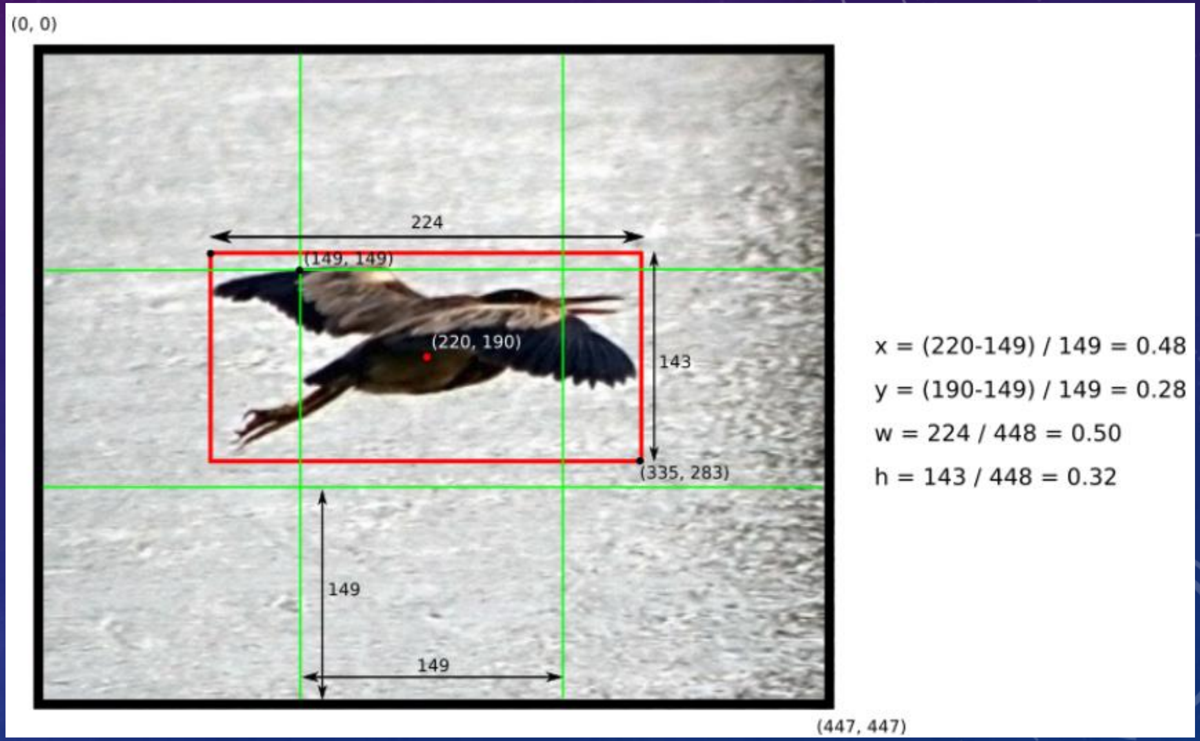


Resize picture to 448 x 448 and a grid cell size is 448/13

YOLO Bounding Box

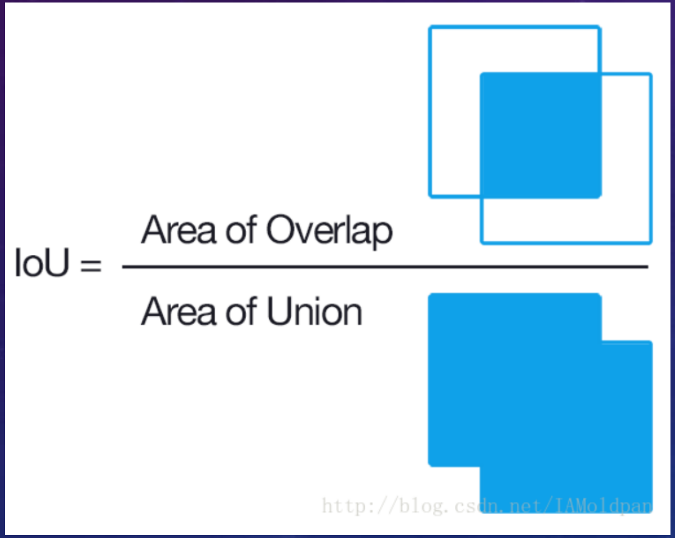
Each grid cell predicts 5 bounding boxes. The bounding box prediction has 5 components: (*x*, *y*, *w*, *h*, *confidence*).

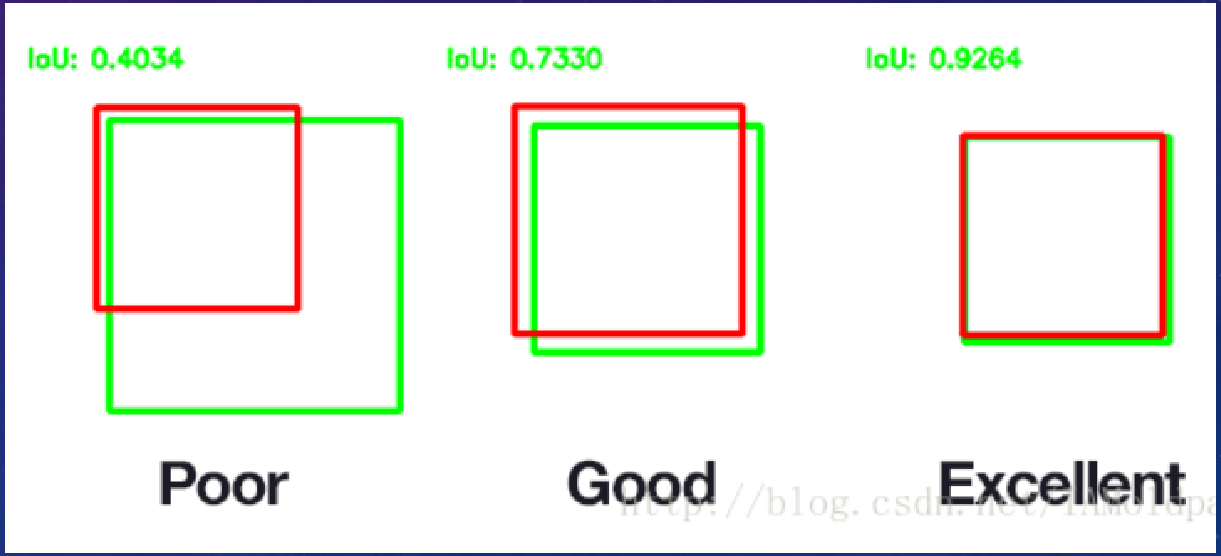
- (*x*, *y*) coordinates represent the center of the box, relative to the grid cell location. These coordinates are normalized to 1.
- (*w*, *h*) coordinates represent the bounding box's weight and height and box dimensions are also normalized to [0, 1], relative to the image size.



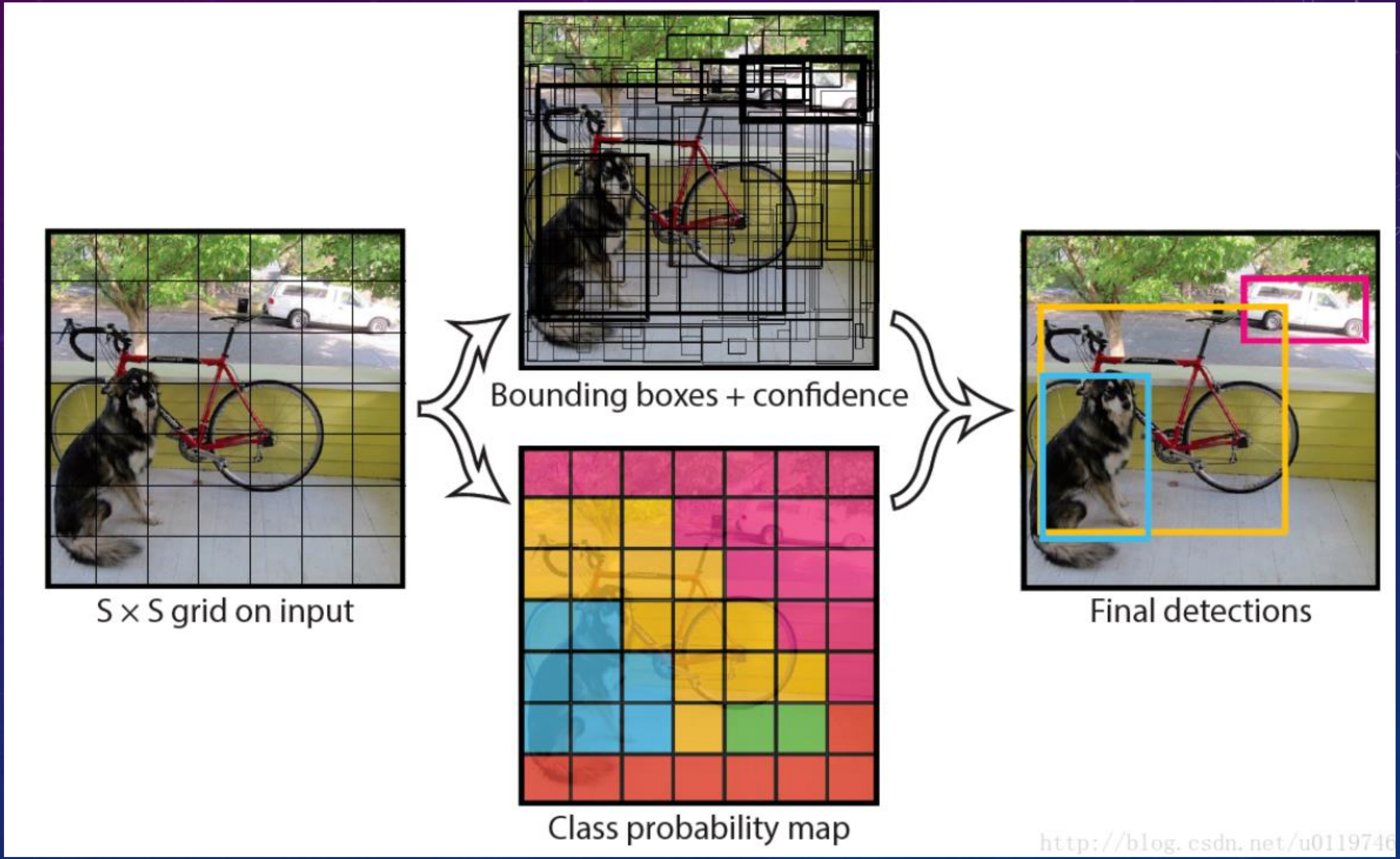
YOLO Bounding Box

- We define confidence as *Pr*(Object) * *IOU*(pred, truth) . If no object exists in that cell, the confidence score should be zero. Otherwise we want the confidence score to be equal to the intersection over union (IOU) between the predicted box and the ground truth.
- IOU

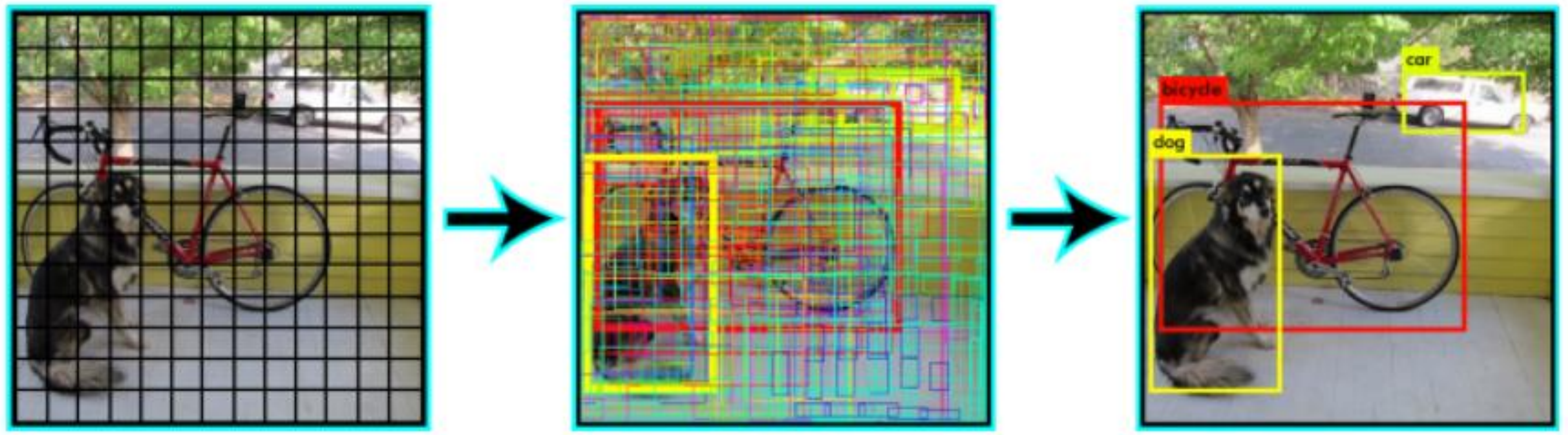




YOLO Object Detection Process



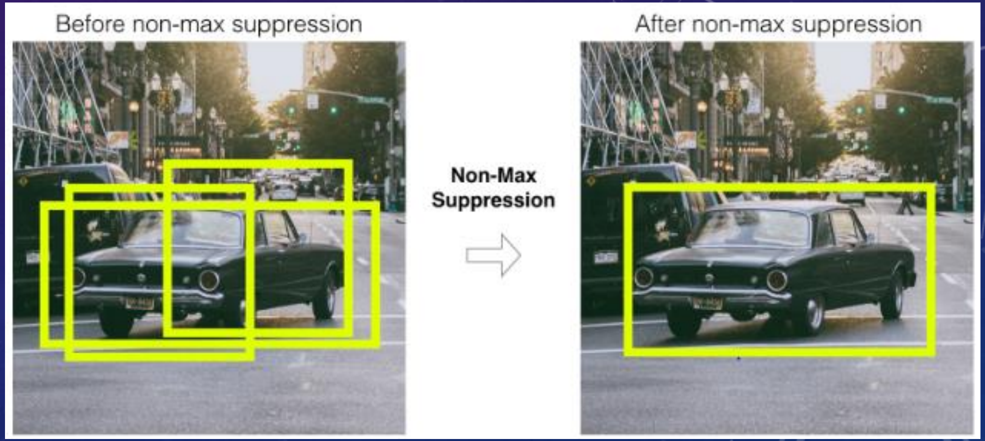
How to make sense of thousands of predictions (middle) ?



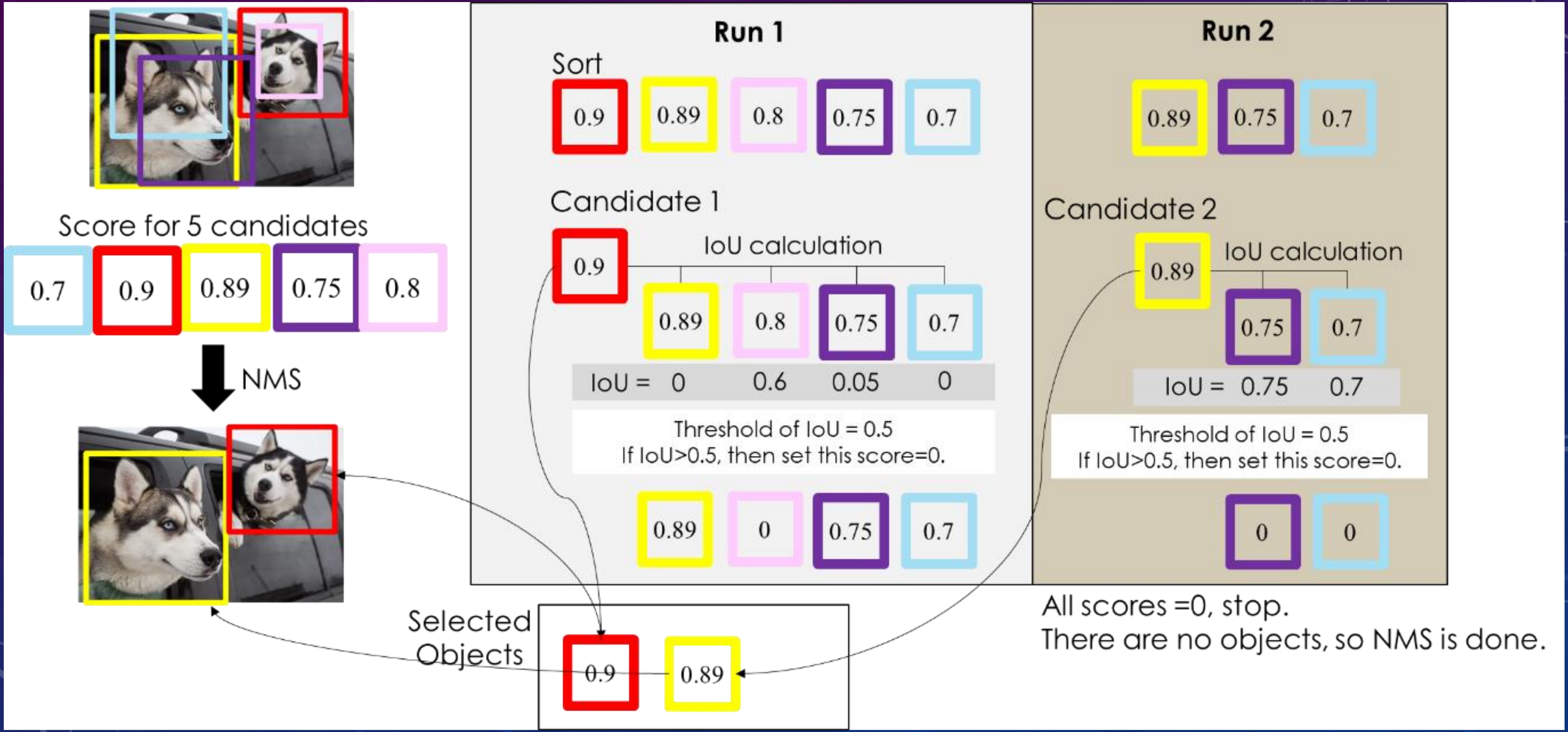
Source

Non-Maximum Suppression (NMS)

- After the regression Network, we have up to 1800 Bounding Boxes (Proposals)
- To come to our final predictions, we input a list of Proposal Boxes B with their confidence scores S and overlap threshold N into NMS, which will output a list of filtered proposals D:
    1. Select Proposal with highest confidence score, remove from B and add it to D
    2. Compare this proposal in D with all proposals in B. Calculate IoU with every other proposal. If IoU is greater than the threshold N, remove that proposal from B
    3. Again take proposal with highest confidence from remaining proposals in B and add it to D
    4. Once again, calc. IoU of this proposal with all proposals in B and eliminate boxes which have higher IoU than threshold
    5. Repeat until no more proposals in B.

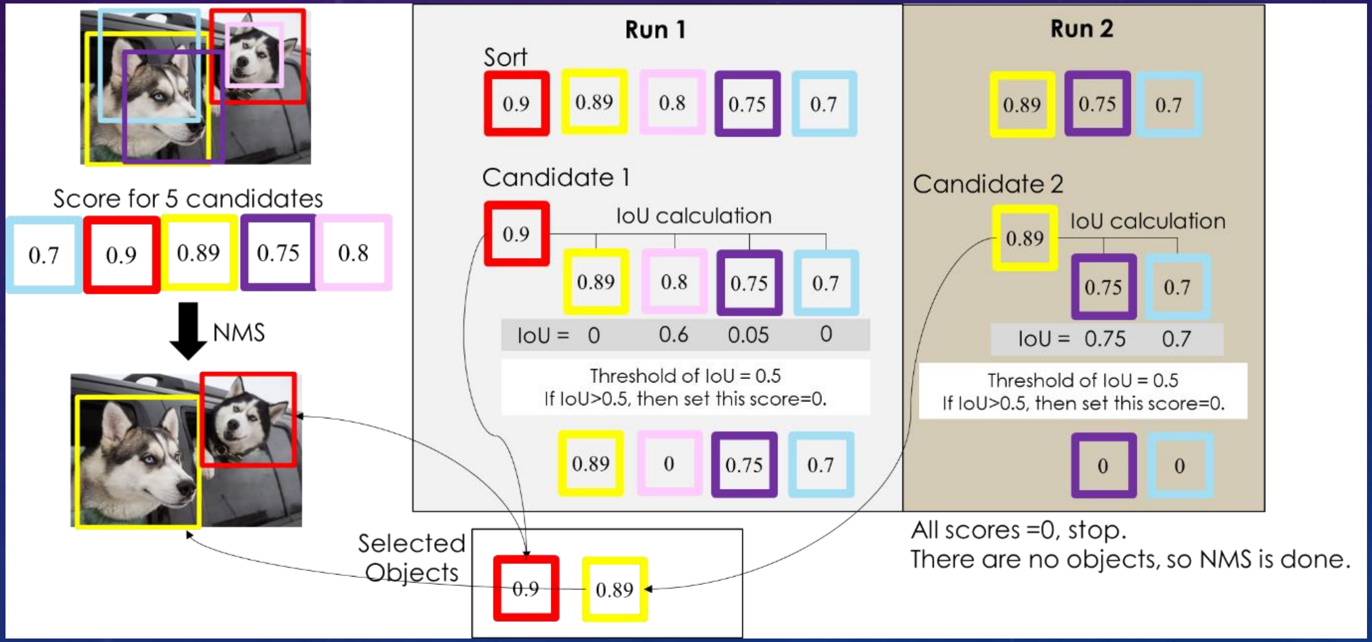


Example: Non-Maximum Suppression (NMS)



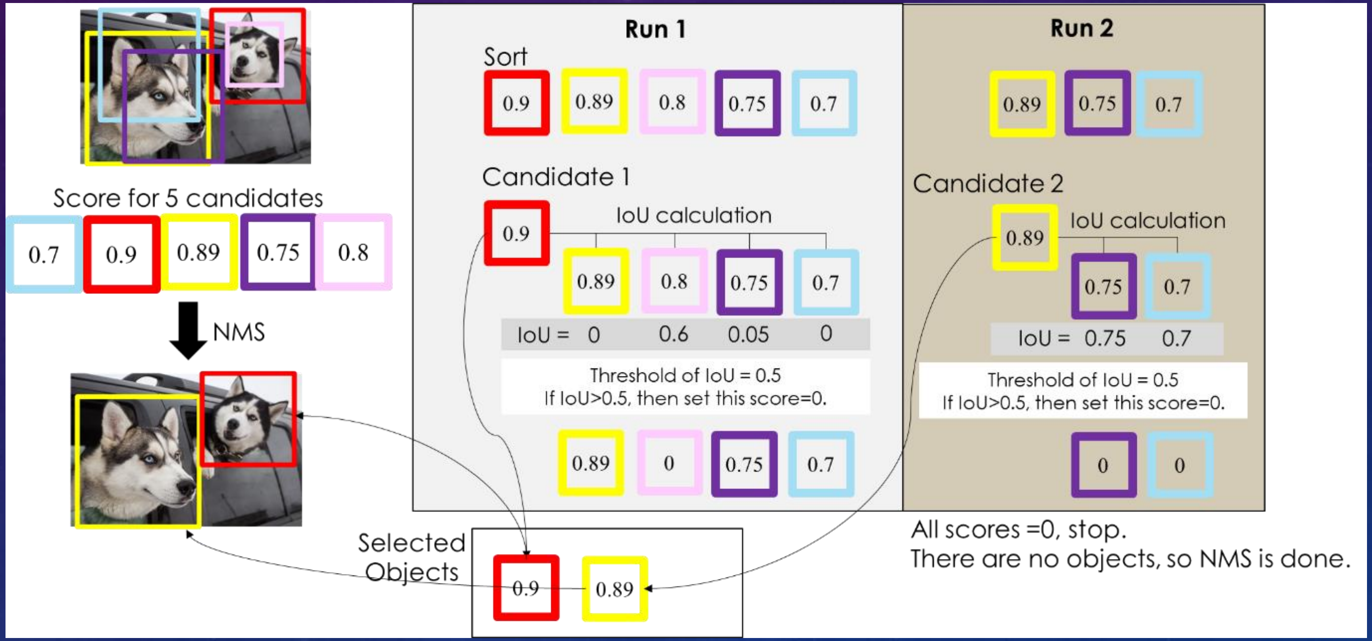
Example: Non-Maximum Suppression (NMS) Run 1

- First BBox confidence in accordance with the degree of ordering, the highest degree of confidence BBox (red) will be elected to "determine the set of objects" within other BBox will see this step to select the best BBox via IoU calculation, if the pink IoU we set greater than 0.5 to 0.6, so the pink BBox confidence is set to 0.



Example: Non-Maximum Suppression (NMS) Run 2

- The highest confidence score in the list is 0.89, yellow box.
- Performing the IoU calculation.
- Since IoU is bigger than our threshold, 0.5, we can eliminate remaining BBox.
- Result are the red and yellow BBox.



See Coding Manual, Chapter 5, Page 2.

Loss Function

- Classification loss

  If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c\in classes}^{n} (p_i(c) - \widehat{p}_i(c))^2$$

Where

1_i^{obj}=1 if an object appears in cell i, otherwise 0.

$\widehat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

Loss Function

- Localization loss

  The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

Where

1_i^{obj}=1 if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Loss Function

- Confidence loss

If an object is detected in the box, the confidence loss (measuring the objectness of the box) is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \widehat{C}_i)^2$$

$\widehat{C}_i$ is the box confidence score of the box j in cell i.

1_i^{obj}=1 if the j th boundary in cell i is responsible for the detection of the object, otherwise 0.

If an object is not detected in the box, the confidence loss is:

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \widehat{C}_i)^2$$

1_{ij}^{noobj} is the complement of 1_{ij}^{obj}.

$\widehat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

Loss Function

- Final loss

The final loss adds localization, confidence and classification losses together.:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2]$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C_i})^2$$

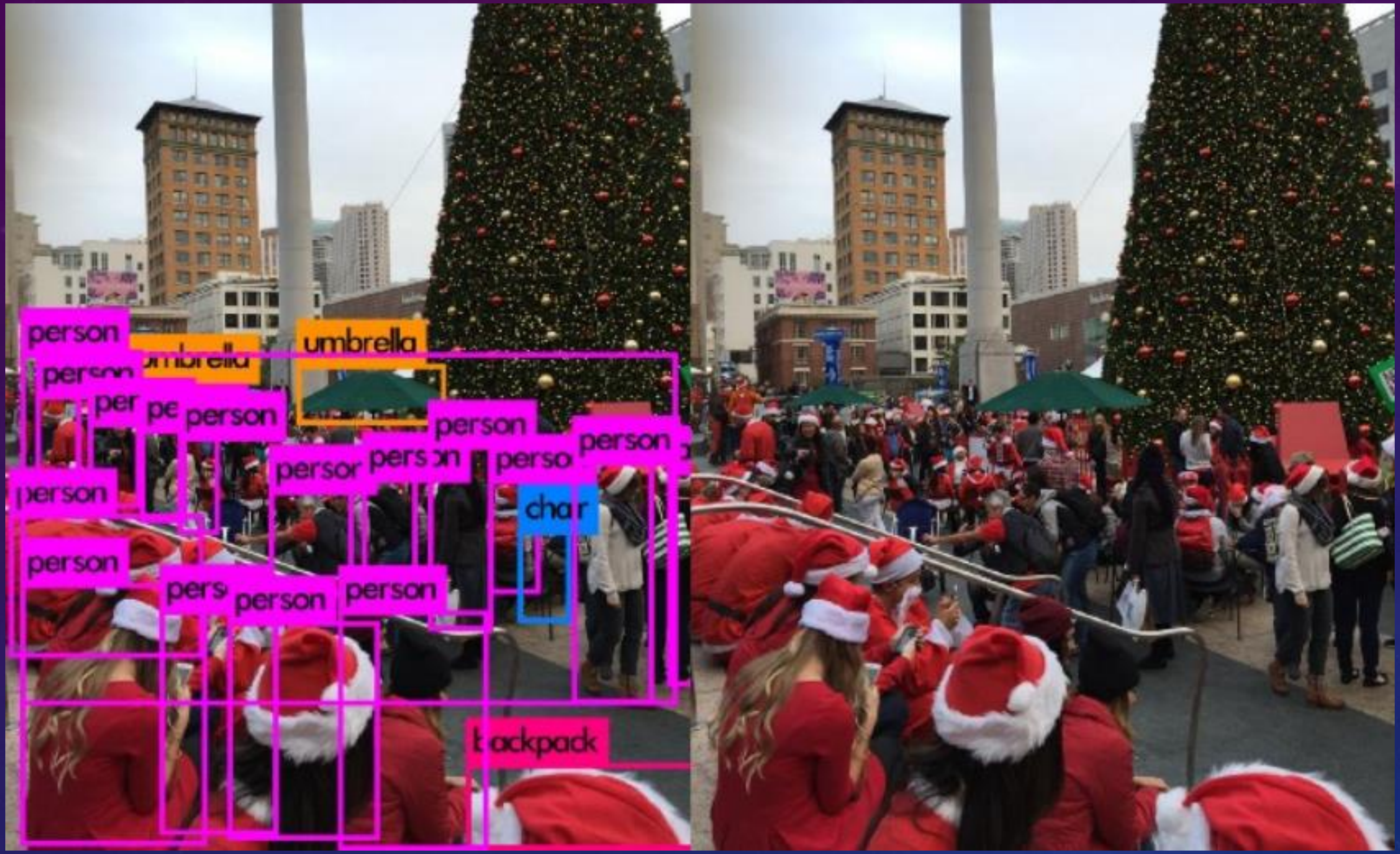
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C_i})^2$$

$$+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes}^{n} (p_i(c) - \hat{p_i}(c))^2$$

Advantages of YOLO

- Fast. Good for real-time processing.
- Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
- YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork.
- Region proposal methods limit the classifier to the specific region. YOLO accesses to the whole image in predicting boundaries. With the additional context, YOLO demonstrates fewer false positives in background areas.
- YOLO detects one object per grid cell. It enforces spatial diversity in making predictions

Limitations of YOLO



- The one object-rule per Grid Cell limits the YOLO detector to detect all objects at close distance
- Have a look at the bottom left of the picture on the left. There are 9 Santas, but only 5 are detected by YOLO

YOLO v2

https://www.youtube.com/watch?v=VOC3huqHrss&feature=youtu.be&ab_channel=YOLOObjectDetection [1:57]

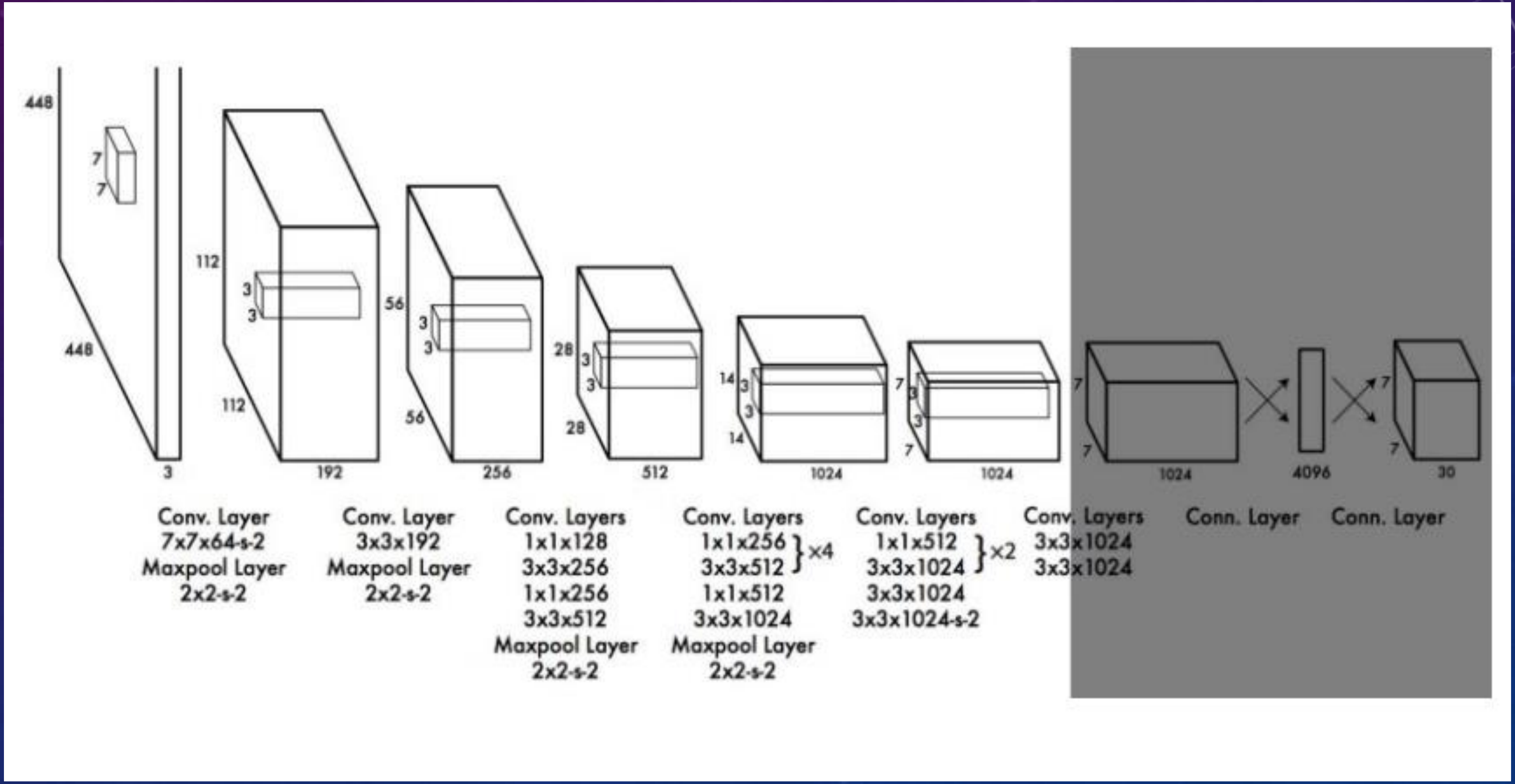
YOLOv2 Improvement

- Network Design
- Batch Normalization (BN)
- High Resolution Classifier
- Convolutions with Anchor Boxes
- Dimension Clusters
- Direct Location Prediction
- Fine-Grained Features

YOLOv2 Improvement

- **YOLOv2 Network Design**

  Remove the fully connected layers responsible for predicting the boundary box.



https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLOv2 Improvement

- Batch Normalization (BN)
  - BN is used on all convolutional layers in YOLOv2.
  - 2% improvement in mAP.
- High Resolution Classifier
  - After trained by 224×224 images, YOLOv2 also uses 448×448 images for fine-tuning the classification network for 10 epochs on ImageNet.
  - 4% increase in mAP.

YOLOv2 Improvement

- Convolutions with Anchor Boxes
  - YOLOv2 removes all fully connected layers and uses anchor boxes to predict bounding boxes.
  - One pooling layer is removed to increase the resolution of output.
  - And 416×416 images are used for training the detection network now.
  - And 13×13 feature map output is obtained, i.e. 32× downsampled.
  - Without anchor boxes, the intermediate model got 69.5% mAP and recall of 81%.
  - With anchor boxes, 69.2% mAP and recall of 88% are obtained. Though mAP is dropped a little, recall is increased by large margin.

YOLOv2 Improvement

- Dimension Clusters
  - The sizes and scales of Anchor boxes were pre-defined without getting any prior information, just like the one in Faster R-CNN.
  - Using standard Euclidean distance based k-means clustering is not good enough because larger boxes generate more error than smaller boxes
  - YOLOv2 uses k-means clustering which leads to good IOU scores:

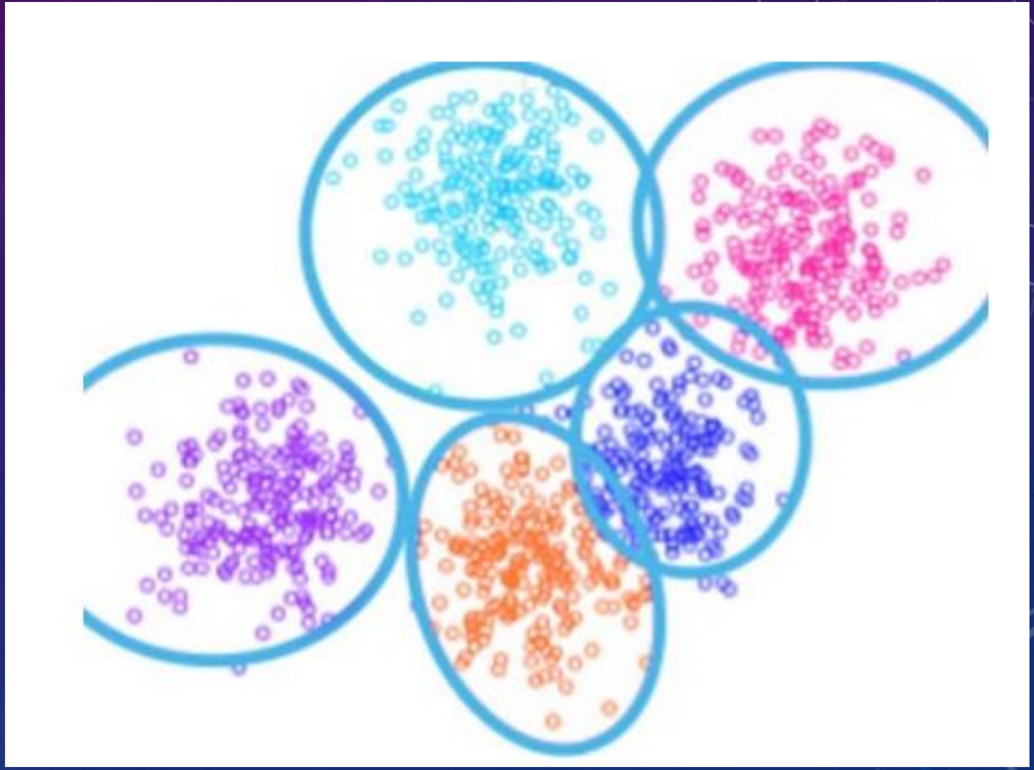
$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

Cluster IOU

  - k = 5 is the best value with good tradeoff between model complexity and high recall.
  - IOU based clustering with 5 anchor boxes (61.0%) has similar results with the one in Faster R-CNN with 9 anchor boxes (60.9%).
  - IOU based clustering with 9 anchor boxes got 67.2%.

YOLOv2 Improvement

- Dimension Clusters
  - In many problem domains, the boundary boxes have strong patterns. For example, in the autonomous driving, the 2 most common boundary boxes will be cars and pedestrians at different distances. To identify the top-K boundary boxes that have the best coverage for the training data, we run K-means clustering on the training data to locate the centroids of the top-K clusters.

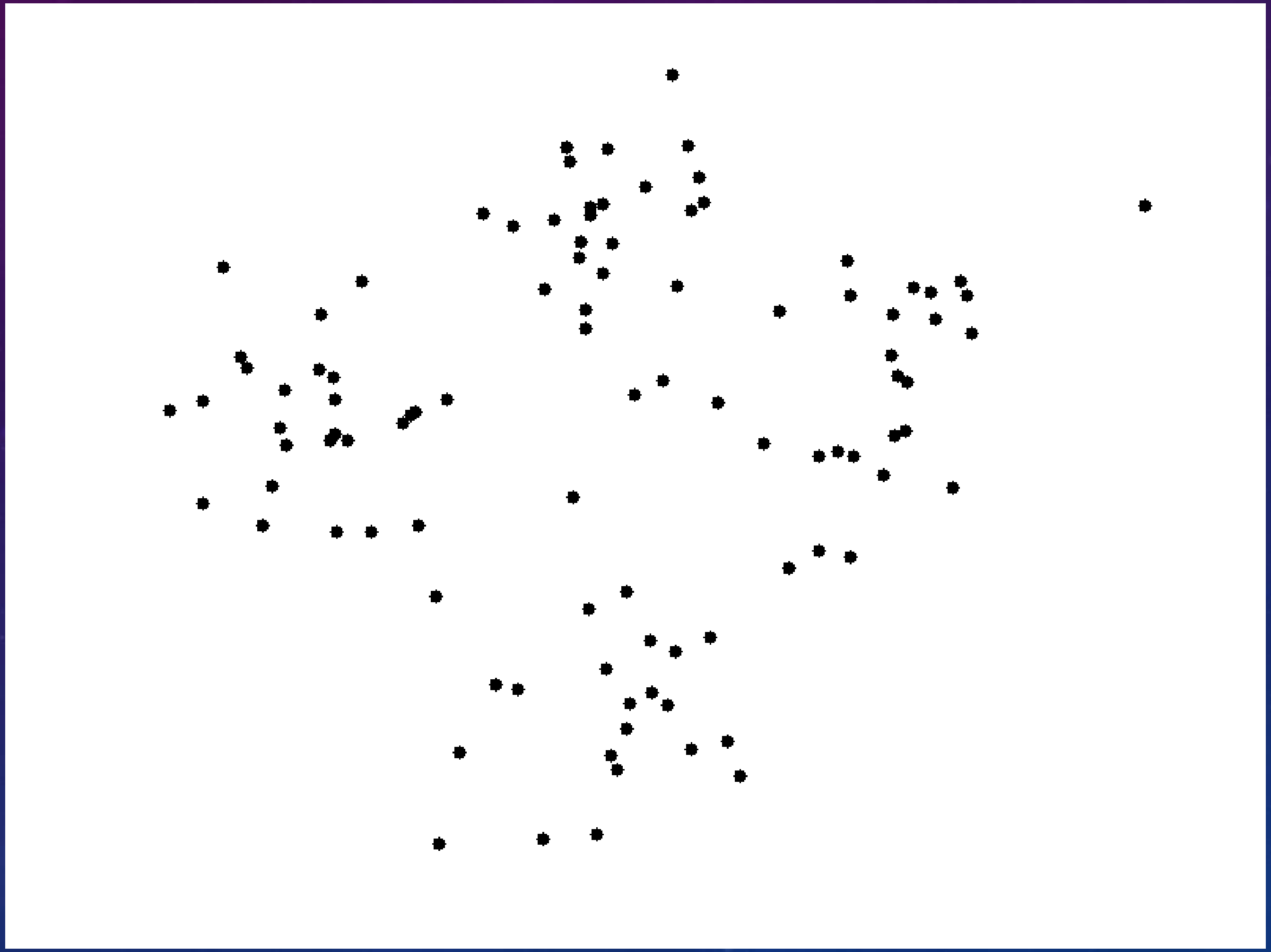


YOLOv2 Improvement

- K-Means is one of the simplest unsupervised learning algorithms that solve the clustering problems.
- The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters).
- The main idea is to define k centers, one for each cluster.

YOLOv2 Improvement

K-Means

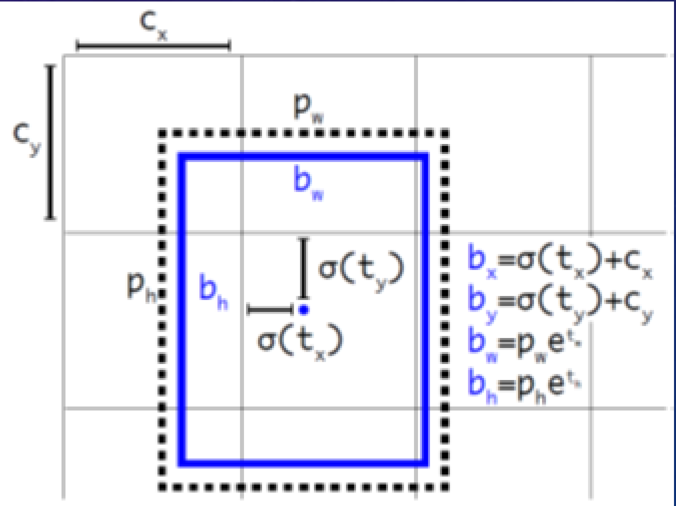


See Coding Manual, Chapter 5, Page 4.

YOLOv2 Improvement

- Direct Location Prediction
  - YOLOv1 does not have constraints on location prediction which makes the model unstable at early iterations. The predicted bounding box can be far from the original grid location.
  - YOLOv2 bounds the location using logistic activation σ, which makes the value between 0 and 1:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$



YOLOv2 Improvement

- Fine-Grained Features
  - The 13×13 feature map output is sufficient for detecting large object.
  - To detect small objects well, the 26×26×512 feature maps from earlier layer is mapped into 13×13×2048 feature map, then concatenated with the original 13×13 feature maps for detection.
- Multi-Scale Training
  - For every 10 batches, new image dimensions are randomly chosen.
  - The image dimensions are {320, 352, …, 608}.
  - The network is resized and continue training.

YOLO v3

https://www.youtube.com/watch?v=MPU2HistivI [3:45]

Introduction into YOLO v3

https://www.youtube.com/watch?v=vRqSO6RsptU&ab_channel=ValentynSichkar [26:55]

YOLOv3 Improvement

- Class Prediction
  - YOLO applies a softmax function to convert scores into probabilities that sum up to one.
  - YOLOv3 uses multi-label classification. For example, the output labels may be “pedestrian” and “child” which are not non-exclusive. (the sum of output can be greater than 1 now.)
  - YOLOv3 replaces the softmax function with independent logistic classifiers to calculate the likeliness of the input belongs to a specific label. Instead of using mean square error in calculating the classification loss, YOLOv3 uses binary cross-entropy loss for each label.
- Bounding box prediction & Cost function calculation
- Feature Pyramid Networks (FPN) like Feature Pyramid

YOLOv3 Improvement

- Bounding box prediction & cost function calculation
  - YOLOv3 predicts an objectness score for each bounding box using logistic regression.
  - YOLOv3 changes the way in calculating the cost function. If the bounding box prior (anchor) overlaps a ground truth object more than others, the corresponding objectness score should be 1.
  - For other priors with overlap greater than a predefined threshold (default 0.5), they incur no cost.
  - Each ground truth object is associated with one boundary box prior only. If a bounding box prior is not assigned, it incurs no classification and localization lost, just confidence loss on objectness.
  - We use tx and ty (instead of bx and by) to compute the loss.

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

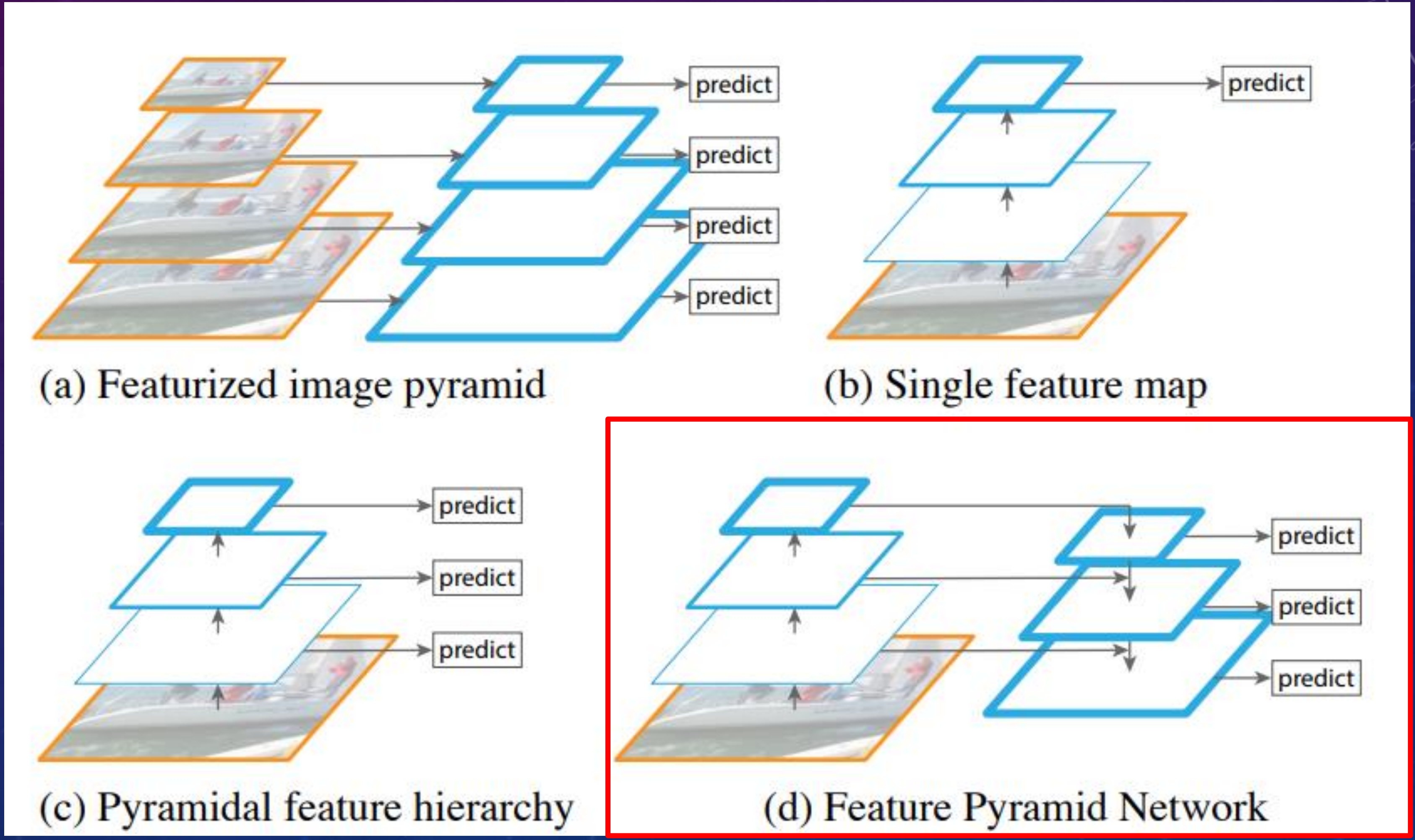
YOLOv3 Improvement

- Feature Pyramid Networks (FPN) like Feature Pyramid
  - YOLOv3 makes 3 predictions per location. Each prediction composes of a boundary box, an objectness and 80 class scores, i.e. $N \times N \times [3 \times (4 + 1 + 80)]$ predictions.
  - YOLOv3 makes predictions at 3 different scales (similar to the FPN):

1. In the last feature map layer.
2. Then it goes back 2 layers back and upsamples it by 2. YOLOv3 then takes a feature map with higher resolution and merge it with the upsampled feature map using element-wise addition. YOLOv3 apply convolutional filters on the merged map to make the second set of predictions.
3. Repeat 2 again so the resulted feature map layer has good high-level structure (semantic) information and good resolution spatial information on object locations.

YOLOv3 Improvement

- Feature Pyramid Networks (FPN)



See Coding Manual, Chapter 5, Page 10.