Computer Vision: Fall 2022 — Lecture 15

Dr. Karthik Mohan

Univ. of Washington, Seattle

November 19, 2022

References

Generic ML/DL

1. Good Book for Machine Learning Concepts
2. Deep Learning Reference

CNN

1. Convolutional Neural Networks for Visual Recognition
2. Convolutional Neural Net Tutorial
3. CNN Transfer Learning
4. PyTorch Transfer Learning Tutorial

CNN Publication References

CNN surveys

1. Convolutional Neural Networks: A comprehensive survey, 2019
2. A survey of Convolutional Neural Networks: Analysis, Applications, and Prospects, 2021

CNN Archs

1. GoogLeNet
2. Top models on ImageNet
3. ResNet ILSVRC paper

Object Detection and Image Segmentation References

Object Detection

1. A survey of modern deep learning based object detection methods
2. YOLO Survey ✓
3. YOLO Original Paper ✓

Product of UW → ALI FARHADI

Learnings from the Mini-Project - Breakout Session!

Breakout and Discuss - Peer Learning (5 mins)

Breakout and discuss in your zoom room - What were your key learnings from the mini-project? What strategies worked and what didn't? How much did hyper-param tuning play a role in the result? Did you get to build your intuition with the models you tested?

Last Lecture

1. Introduction to Object Detection and Instance Segmentation
2. R-CNN model and metrics for Object Detection

& data sets

Today

1. Object Detection Recap
2. R-CNN variants - Fast and Faster R-CNN
3. YOLO - Single Stage Object Detection
4. Results and Benchmarking on the data sets

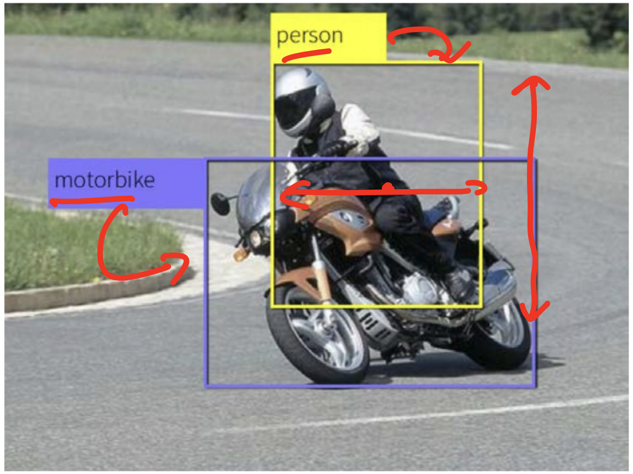
Today

1. **Object Detection Recap**
2. R-CNN variant - Fast R-CNN
3. YOLO - Single Stage Object Detection
4. Results and Benchmarking on the data sets

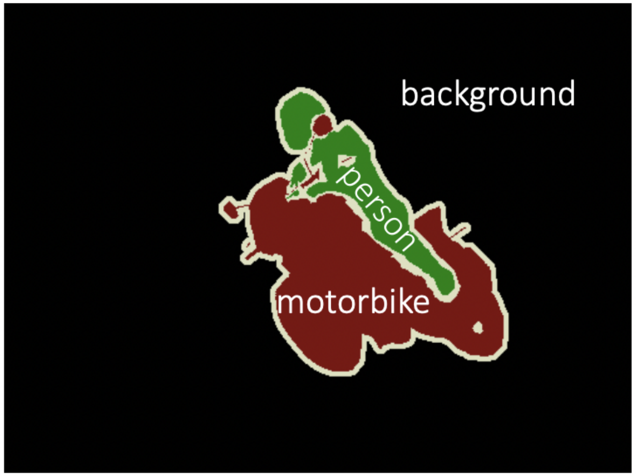
Object Detection vs Image Segmentation

input image

object detection

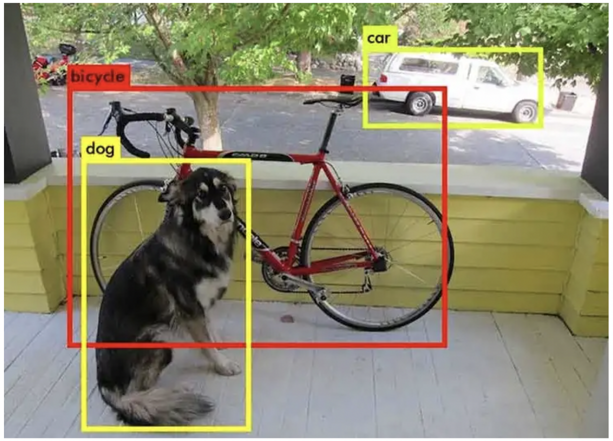


segmentation



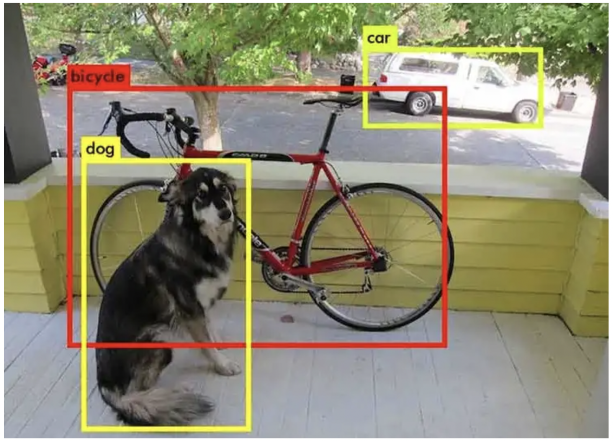
(c_x, c_y, w, h), clas

Object Detection History



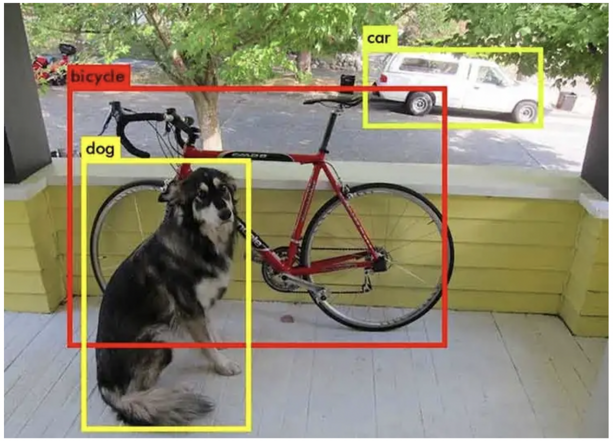
1. Has been an uphill task until 2012

Object Detection History



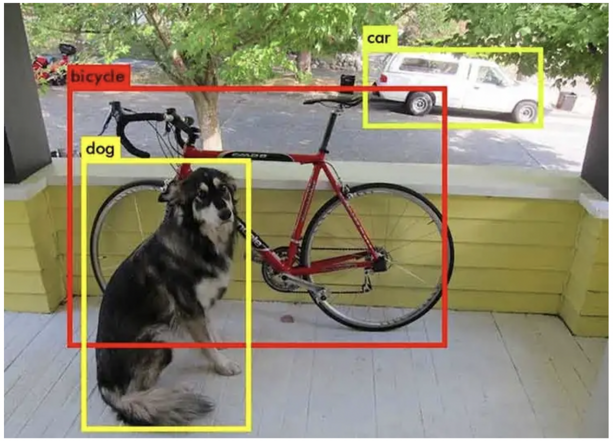
1. Has been an uphill task until 2012
2. Early detectors for objects - Ensemble of hand-crafted ones

Object Detection History



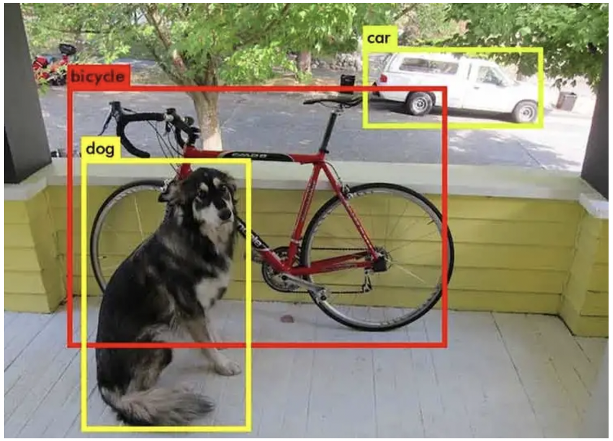
1. Has been an uphill task until 2012
2. Early detectors for objects - Ensemble of hand-crafted ones
3. Early detectors: Low accuracy and cumbersome/time-consuming

Object Detection History



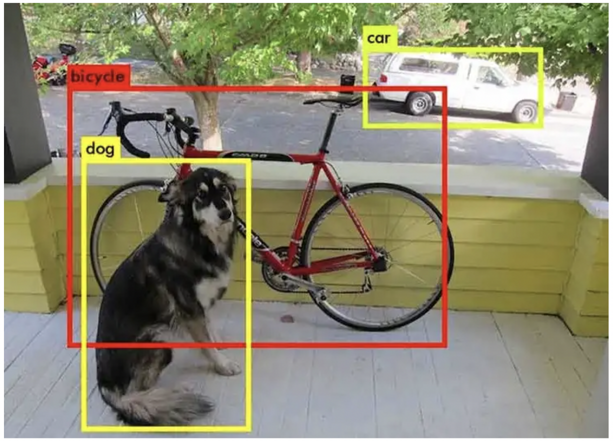
1. Has been an uphill task until 2012
2. Early detectors for objects - Ensemble of hand-crafted ones
3. Early detectors: Low accuracy and cumbersome/time-consuming
4. CNN changed the landscape - Better Accuracy, faster train, generalizability

Object Detection History



1. Has been an uphill task until 2012
2. Early detectors for objects - Ensemble of hand-crafted ones
3. Early detectors: Low accuracy and cumbersome/time-consuming
4. CNN changed the landscape - Better Accuracy, faster train, generalizability
5. AlexNet (2012) - First CNN archs to be applied to Obj. Detection

Object Detection History



1. Has been an uphill task until 2012
2. Early detectors for objects - Ensemble of hand-crafted ones
3. Early detectors: Low accuracy and cumbersome/time-consuming
4. CNN changed the landscape - Better Accuracy, faster train, generalizability
5. AlexNet (2012) - First CNN archs to be applied to Obj. Detection
6. **Real world application:** Self-driving cars

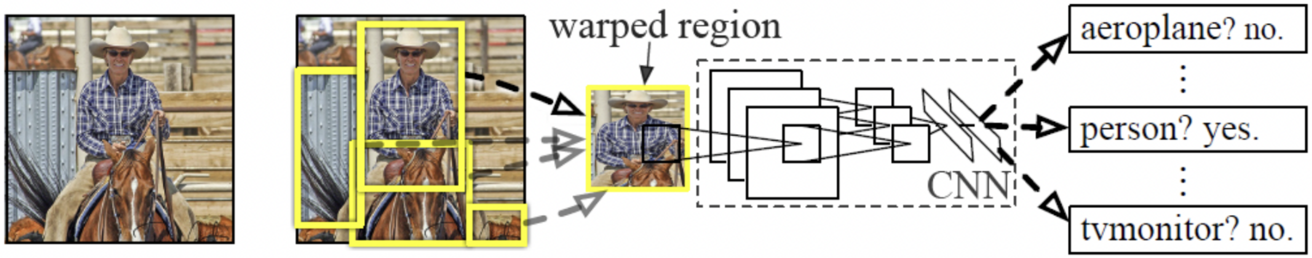
Multi-class Classification vs Object Detection

Multi-Class Classification

Object Detection

- **Localization**
- **Multi-class Classification!**

Multi-label Classification vs Object Detection



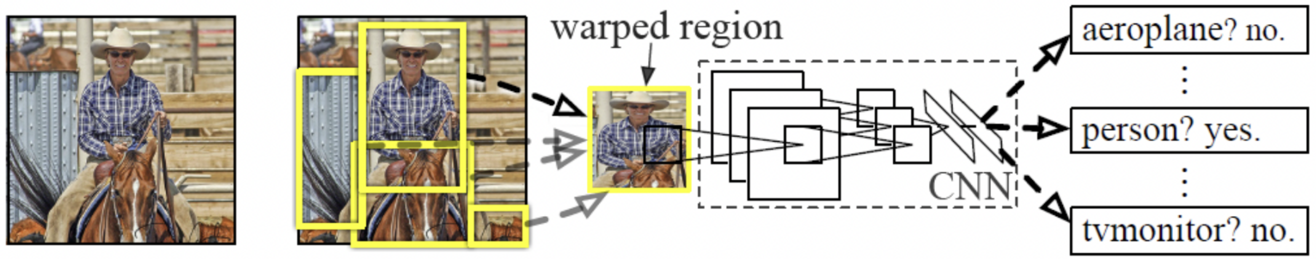
Multi-Class Classification

Object Detection

Localization

Multi-class Classification!

Multi-class Classification vs Object Detection



Multi-Class Classification

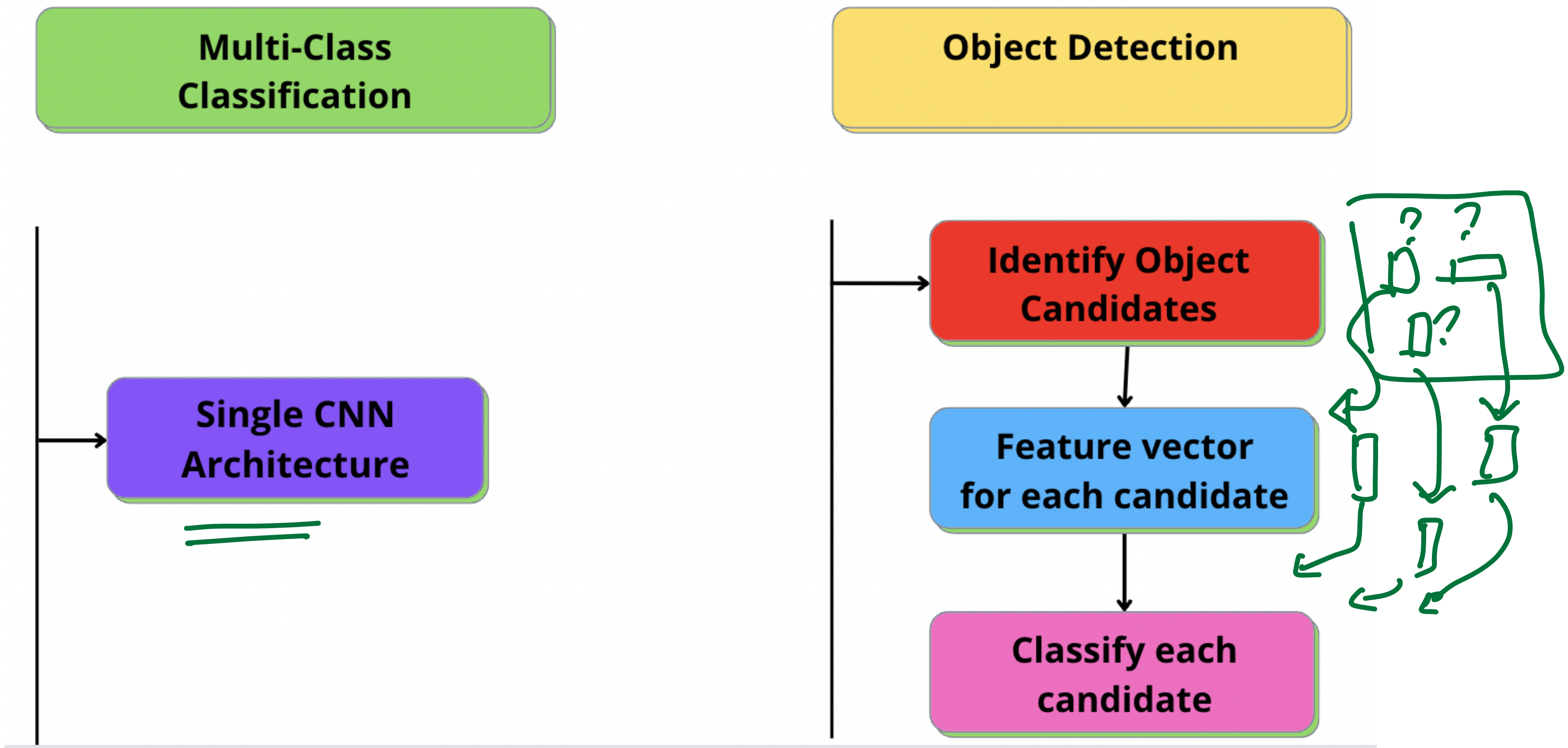
Object Detection

Localization

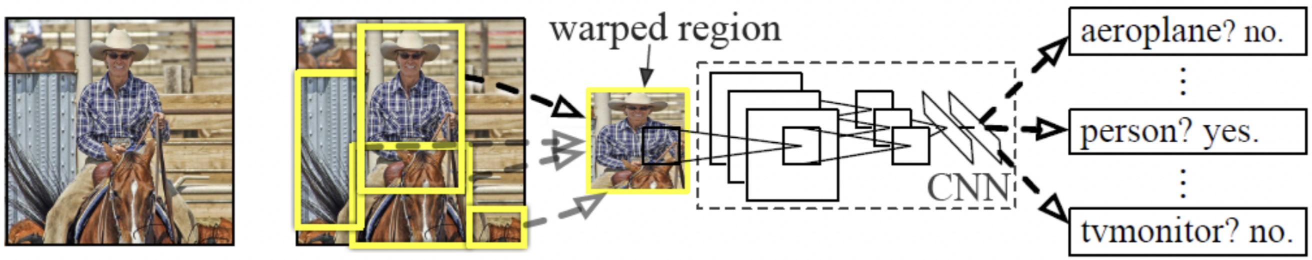
Multi-label Classificaiton

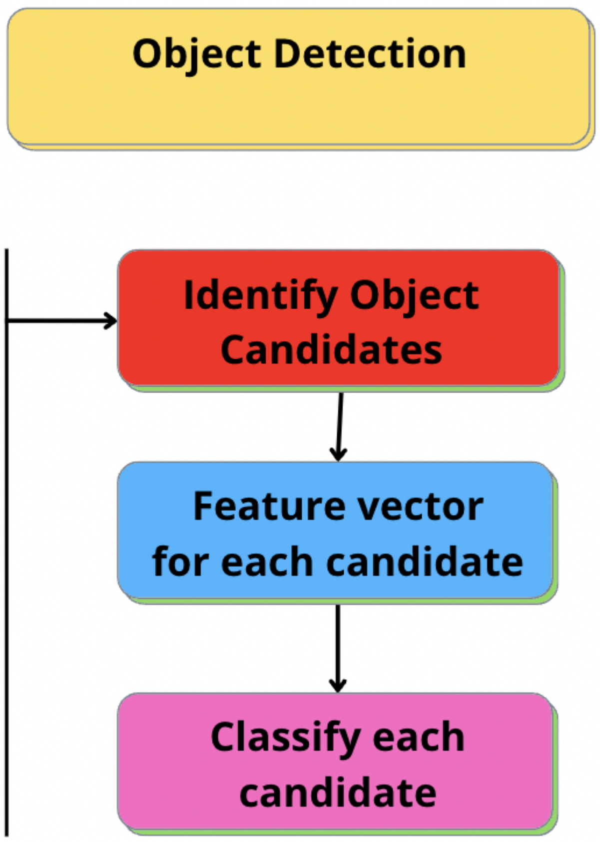
Multi-class Classification!

Multi-class Classification vs Object Detection

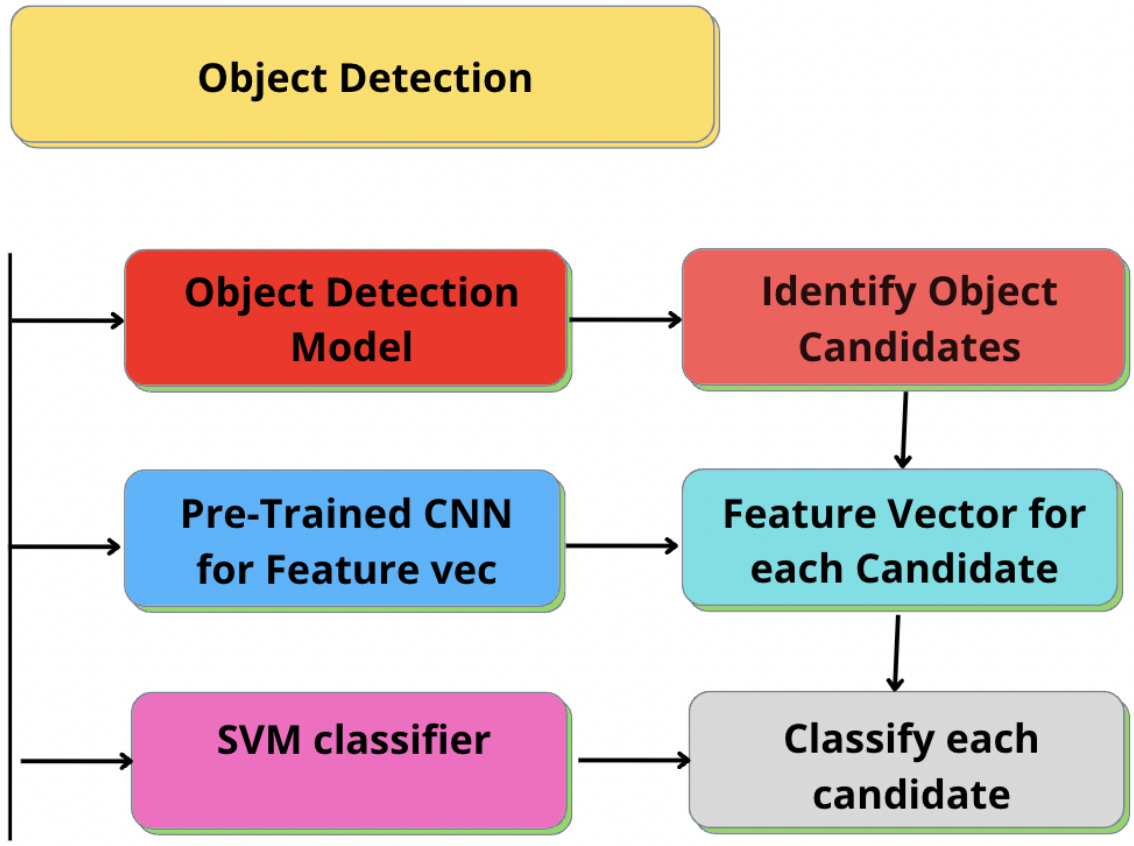


Object Detection Intuition

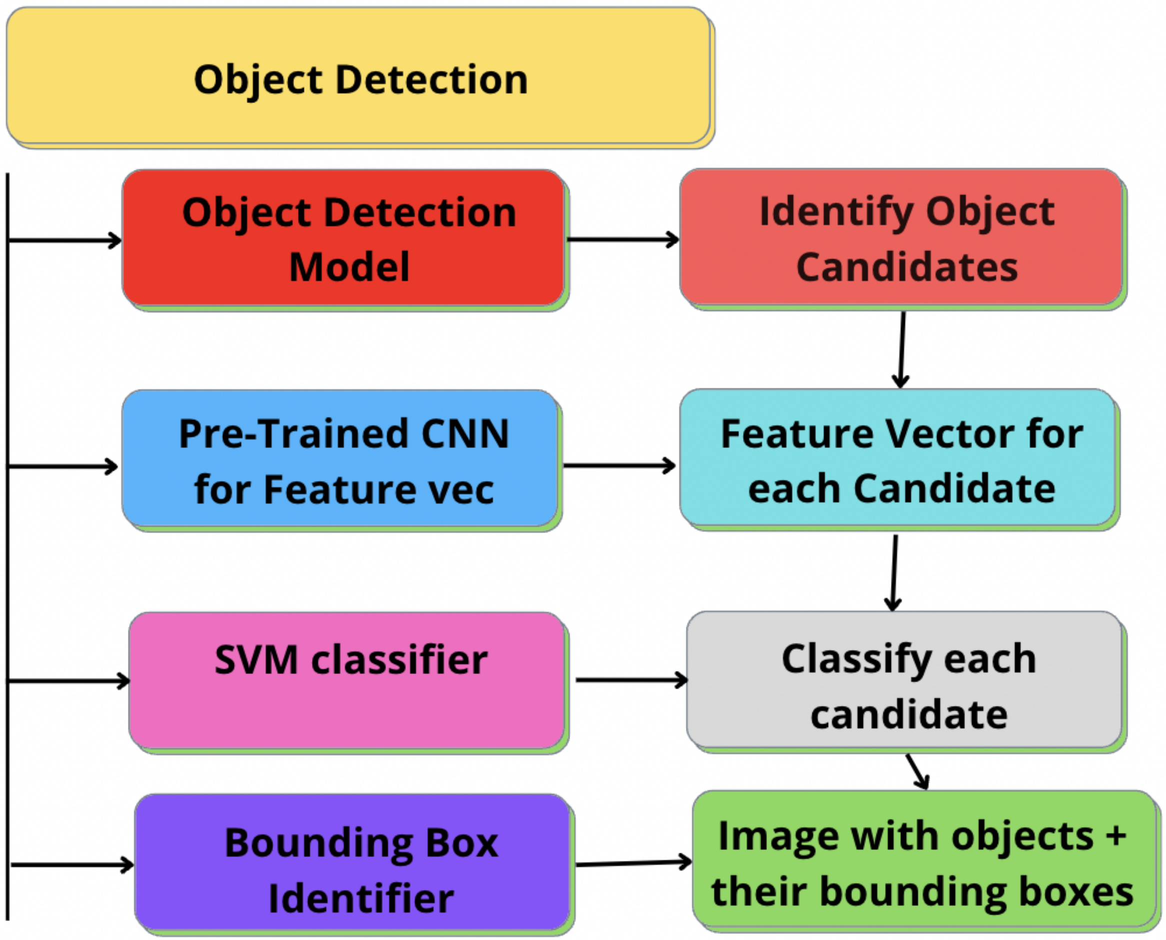




Object Detection Model Framework



Object Detection Model Framework

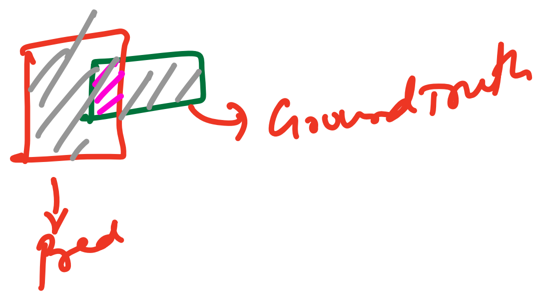


R-CNN
↳ Region

Metrics for Classification

IOU

Intersection over Union: Is the ratio of the area of the *intersection between the predicted bounding box and the ground truth bounding box* **over** *the union of the area between the predicted bounding box and the ground truth bounding box*



Metrics for Classification

IOU

Intersection over Union: Is the ratio of the area of the *intersection between the predicted bounding box and the ground truth bounding box* **over** *the union of the area between the predicted bounding box and the ground truth bounding box*

MAP @ 0.5 IOU

Average Precision (AP) @ 0.5 IOU: If

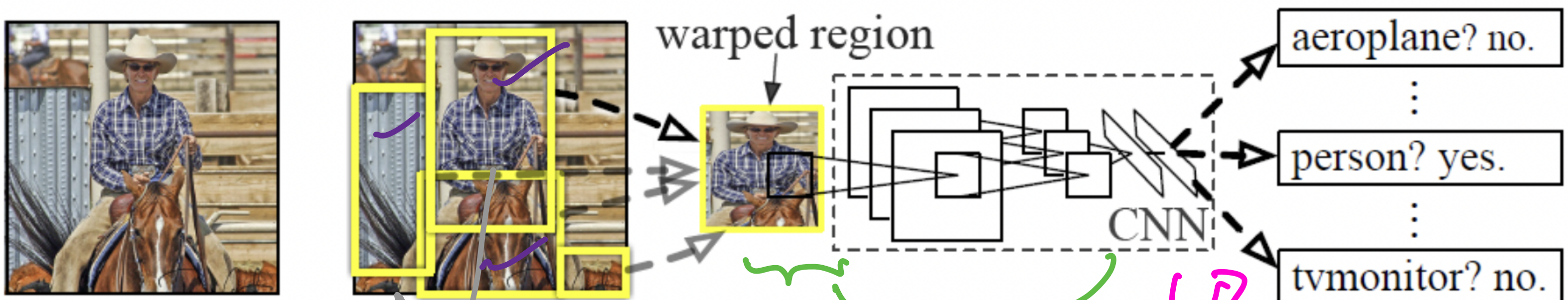
$$\text{IOU} > 0.5$$

$\rightarrow t$ (0.6, 0.7, ...)

across examples of a given class, count the precision as 1, else 0. Average Precision is the average of all the precisions in a given class.

First CNN Model for Object Detection: R-CNN model

Region of Interest:



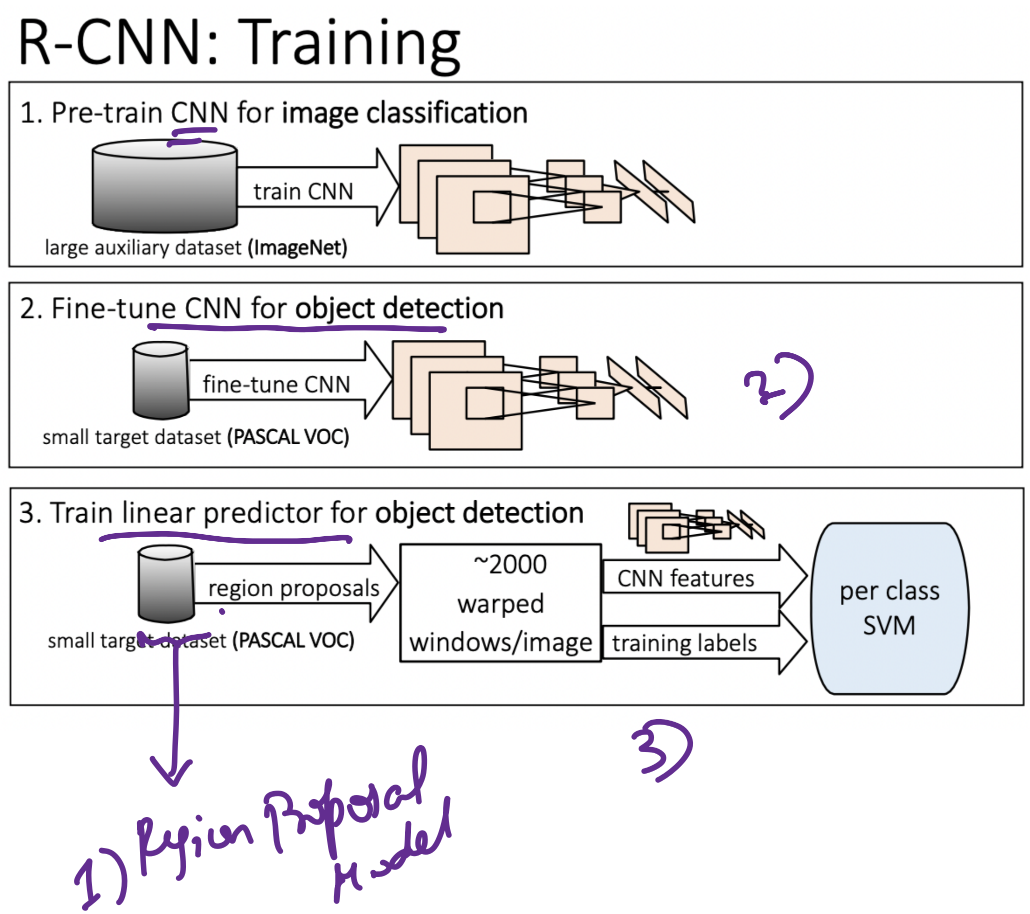
1) Identify candidate Bounding Boxes

2) Get a feature vec for candidate

3) classify each candidate

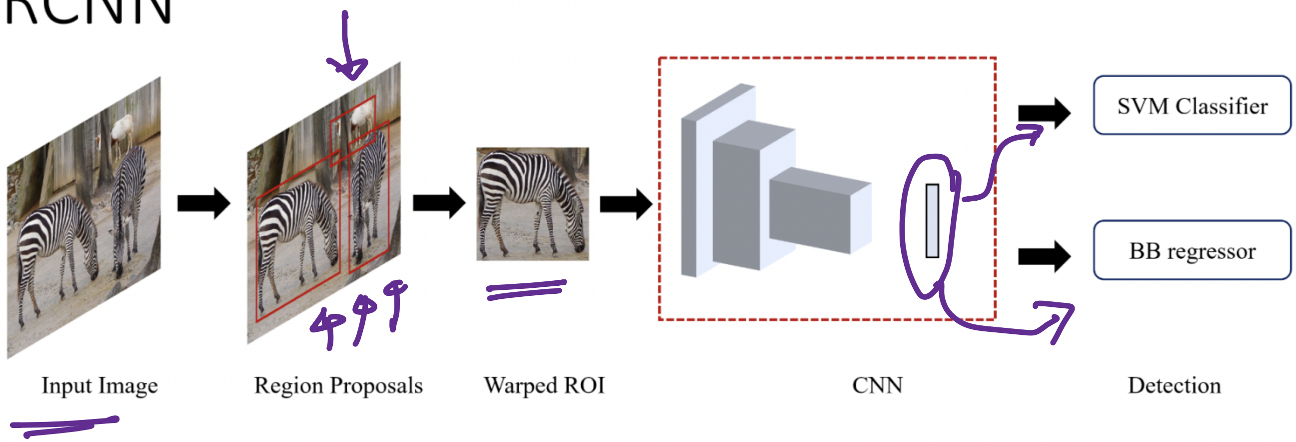
4) Prune candidates & return results

First CNN Model for Object Detection: R-CNN model

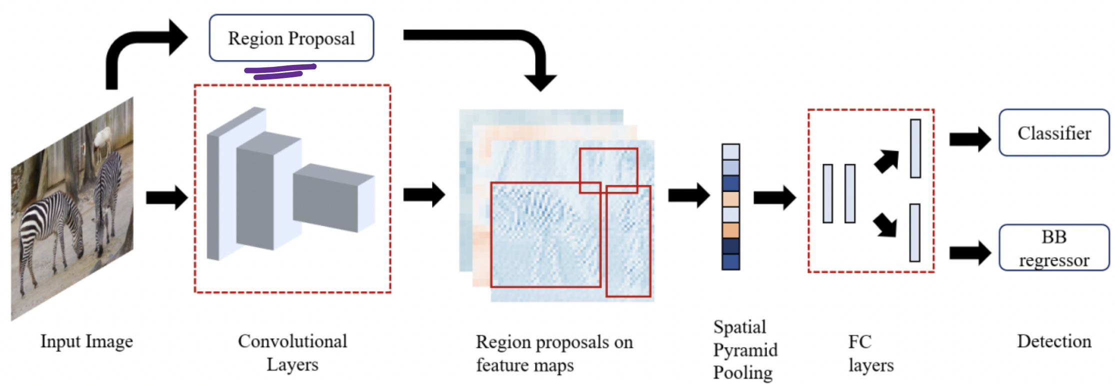


R-CNN variant

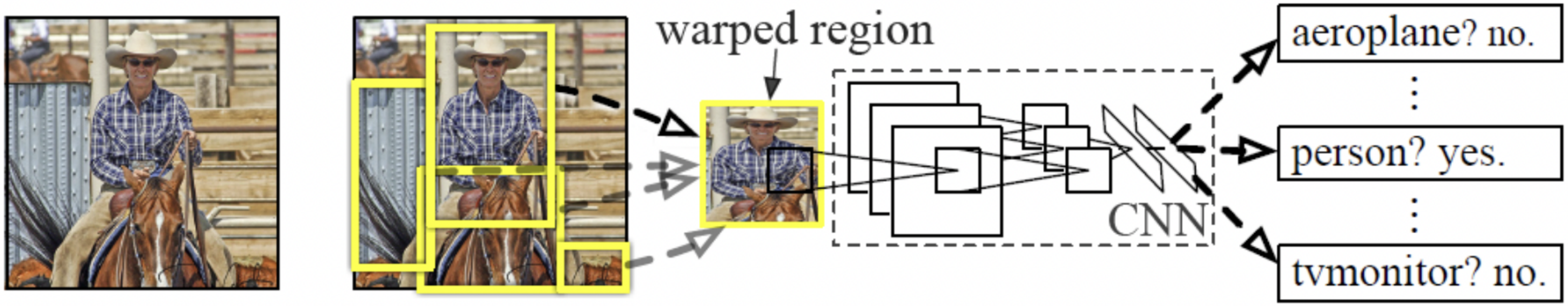
RCNN



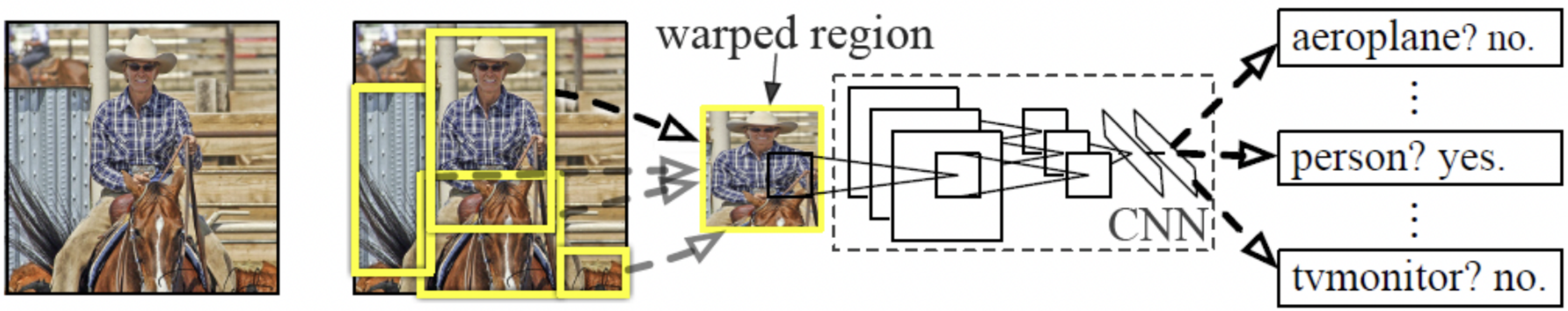
Fast RCNN



Drawbacks of Two Stage Detection



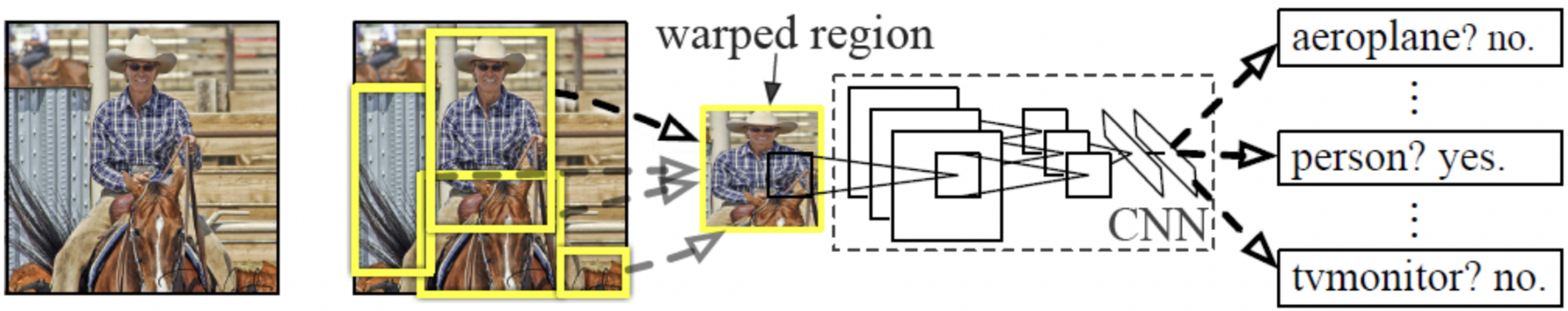
1. **R-CNN variants:** a) Generate Bounding Boxes b) Classify these boxes and c) Merge bounding boxes to eliminate duplicates - 3 steps, 3 models to train!



1. **R-CNN variants:** a) Generate Bounding Boxes b) Classify these boxes and c) Merge bounding boxes to eliminate duplicates - 3 steps, 3 models to train!
2. **Inference Time Taken:** FPS can be low - E.g. even Faster R-CNN has a speed of 5 FPS at inference time

Slow for video processing!

Drawbacks of Two Stage Detection



1. **R-CNN variants:** a) Generate Bounding Boxes b) Classify these boxes and c) Merge bounding boxes to eliminate duplicates - 3 steps, 3 models to train!
2. **Inference Time Taken:** FPS can be low - E.g. even Faster R-CNN has a speed of 5 FPS at inference time
3. **Train Time:** 3 separate models implies more time to train

Single Stage Detection

Simplify!

1 Model

What if we could simplify the detection process and also make training and inference more efficient in the process? Enter **YOLO**

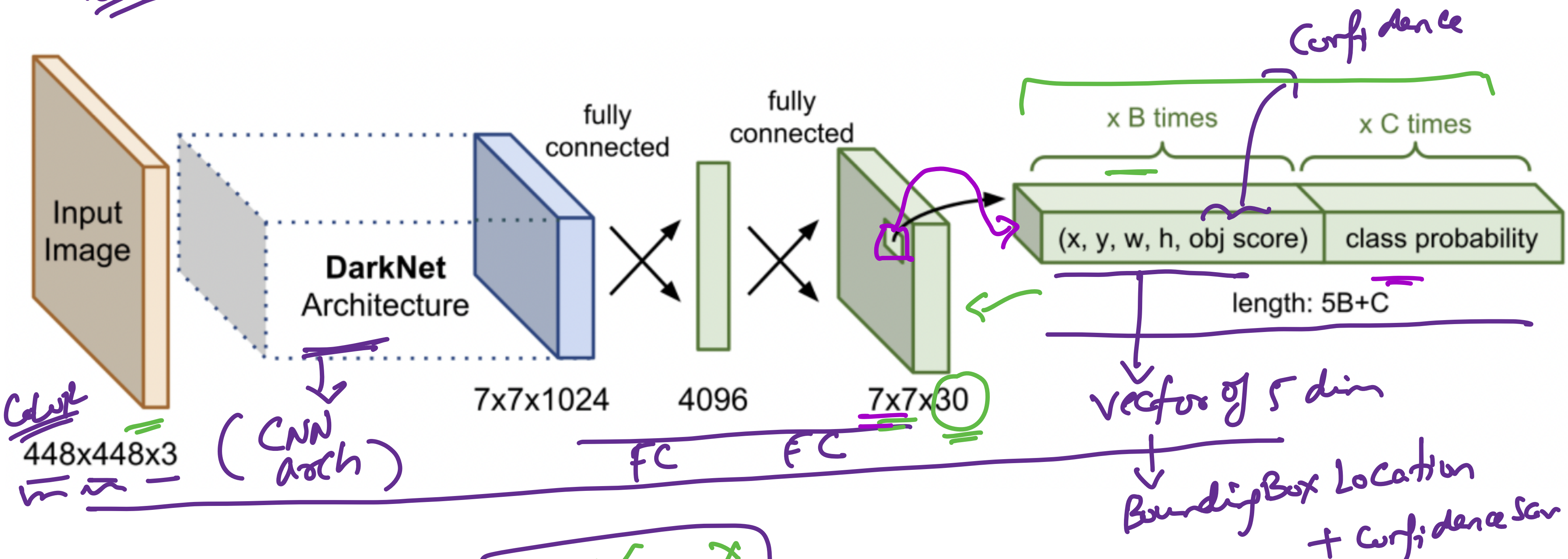
Today

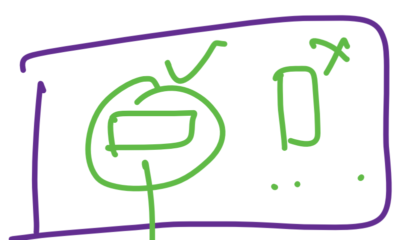
1. Object Detection Recap
2. R-CNN variants - Fast and Faster R-CNN
3. **YOLO - Single Stage Object Detection**
4. Results and Benchmarking on the data sets

YOLO - Single Stage Detection

2016 | UW

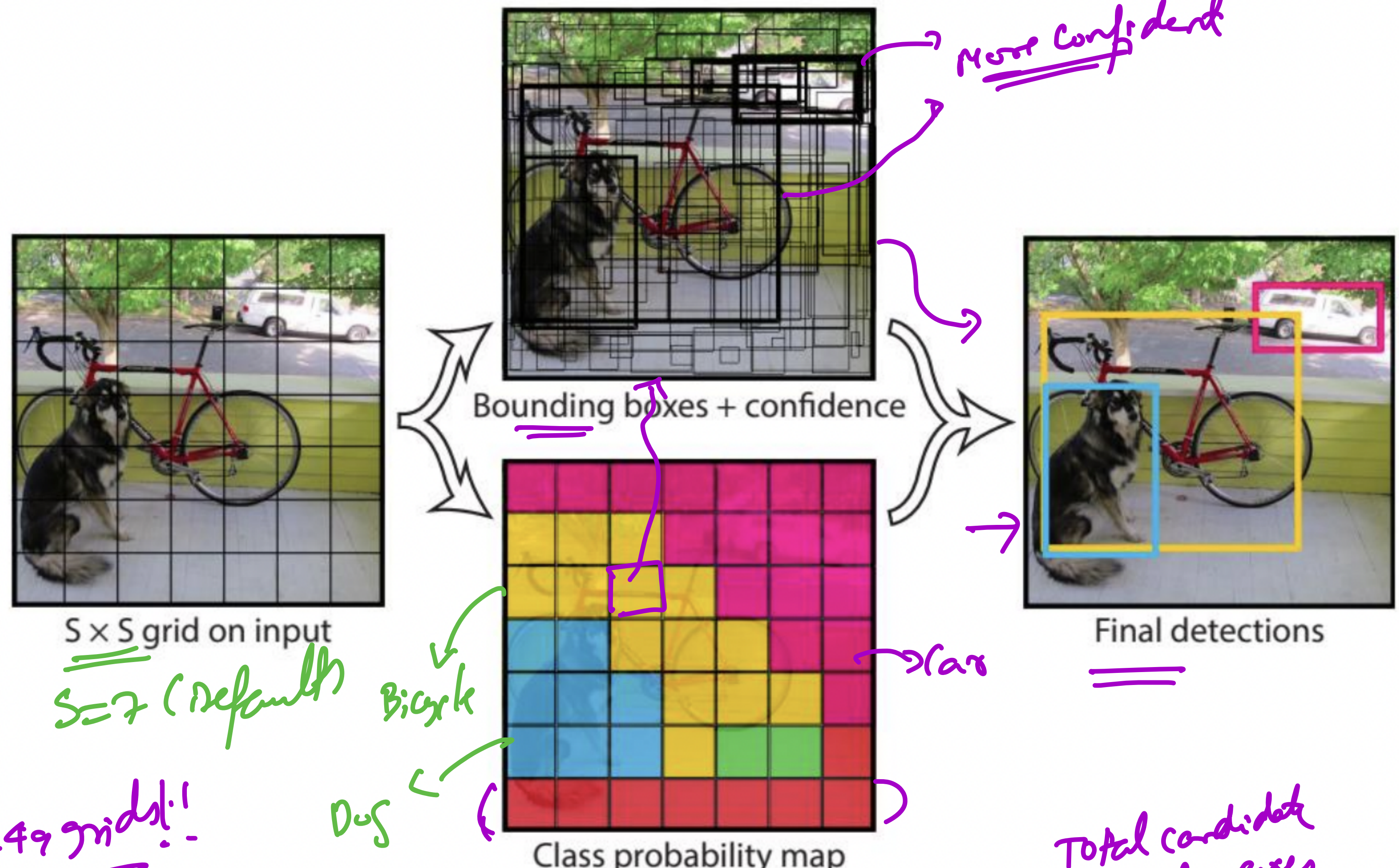
YOLO – YOU ONLY LOOK ONCE
Y O L O





0.3 / 0.9
Dog
(cx, cy, w, h), confidence, class

YOLO Breakdown



More confident

S=7 (Default)

Bicycle

Dog

Car

7x7=49 grids!!

B=3 ⇒ Total candidate Bounding Boxes $= 7 \times 7 \times 3 = 147$

YOLO v3 Video

YOLO Real-time Detection

ICE #1

YOLO breakdown

YOLO implicitly divides the input into 7x7 regions and makes bounding box predictions for each of the 49 grids. How would this be different from 'explicitly' breaking up the input into 49 grids and passing each grid through an object detection method?

1. It would be the same thing and give similar results
2. It would be different
3. Surrounding context doesn't get encoded in the latter and impacts accuracy of object detection
4. The explicit breakup would yield higher accuracy on the IOU metric as compared to the implicit breakup

YOLO benefits

1. **Single pass**

Less clunky

YOLO benefits

1. **Single pass**
2. **Fast**

YOLO benefits

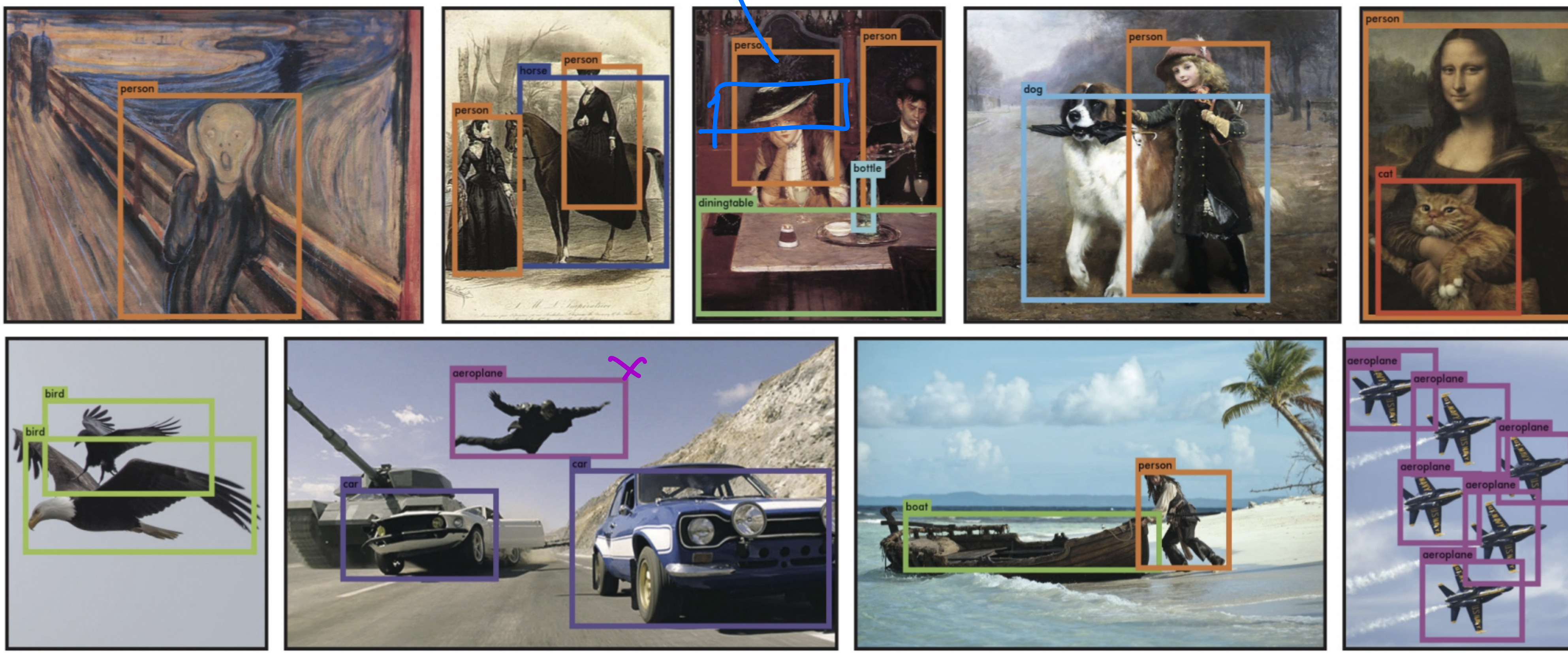
1. **Single pass**
2. **Fast**
3. **Global reasoning** (Keeps context of surrounding objects inform object detection)

vs (Local reasoning) in R-CNN

YOLO benefits

1. **Single pass**
2. **Fast**
3. **Global reasoning**
4. **More generalized representations**

YOLO examples



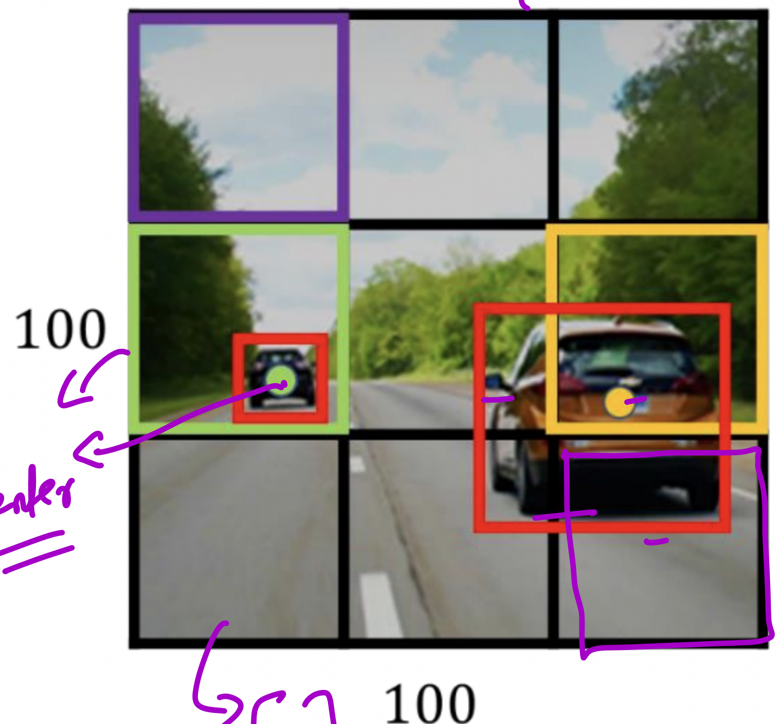
YOLO labels

49x8 = 392

labels

5+C

=3



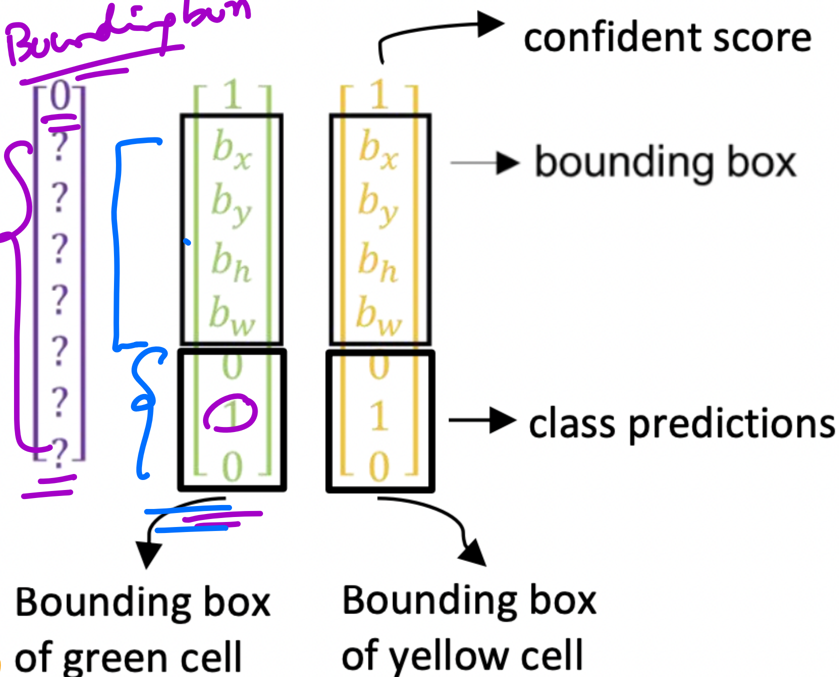
Center

Confidence

Labels for training for each grid cell:

Bounding box

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$



Class Labels

— One Hot encoding

only one component non-zero

ICE #2

5 → $[c_x, c_y, w, h, \text{confidence}]_{5\times1}$

B bounding boxes ↳ 5 dim vector +
→ Prob. for C classes → C dim vector

Label Dimensions (2 mins)

Consider a data set of Images, let's say a subset of the MS Coco data set. Let's look at 3 different but related ML problem formulations on this data set: Multi-Class Classification, Multi-Label Classification, and Object Detection (YOLO style). Let's say there are 100 classes to pick from. For object detection, assume that YOLO uses a 9x9 grid with each grid producing 3 different bounding boxes. What's the dimension of output vector for these 3 problem formulations?

total dimension

1. 100, 300, and 8100
2. 100, 100, and 9315
3. 100, 8100, and 9315
4. 100, 100 and 8100

multi-class

$$\begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \end{bmatrix}_{3\times1}$$

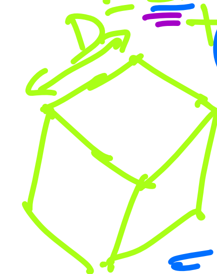
1

multi-label

$$\begin{bmatrix} 0.5 \\ 0.7 \\ 0.9 \end{bmatrix}_{3\times1}$$

→ Each is a prob.

D = 3×5 + 100

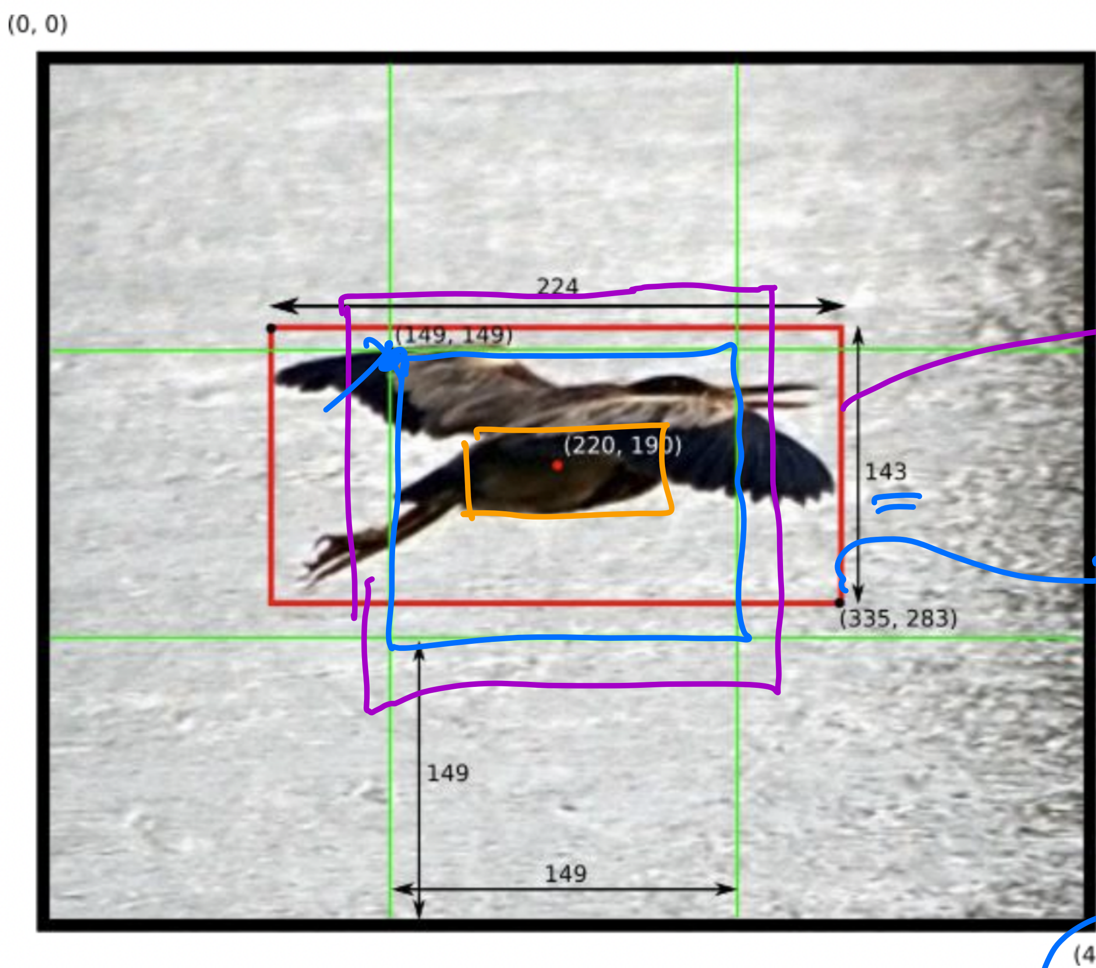


9×9

YOLO Bounding Box

Blue Grid
$\Rightarrow$ 3x 5dim (confidence + Location)
$\hookrightarrow$ Bounding Box

5dim vector



x = (220-149) / 149 = 0.48

y = (190-149) / 149 = 0.28

w = 224 / 448 = 0.50

h = 143 / 448 = 0.32

$0 < x < 1$

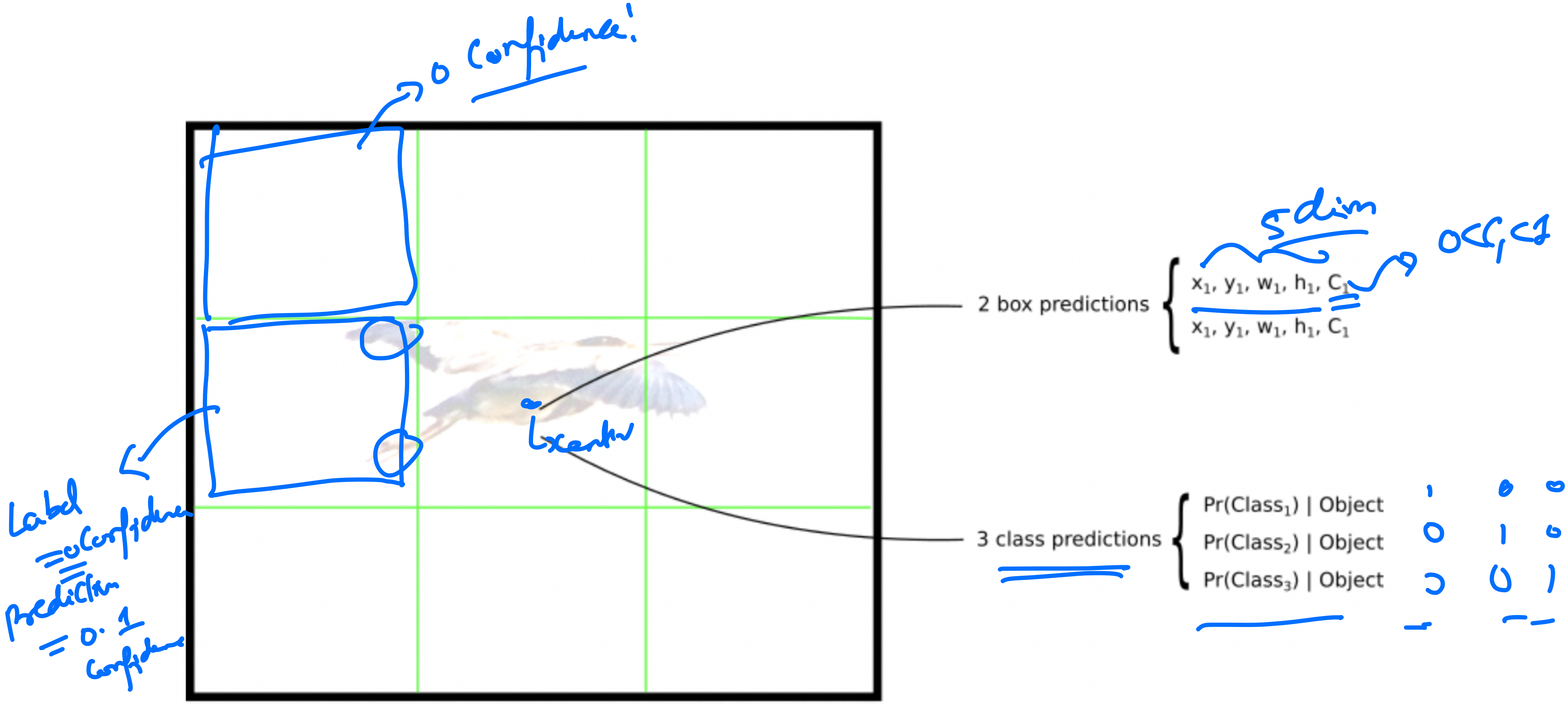
$0 < y < 1$

$0 < w < 1$

$0 < h < 1$

Key pre-processing step:- Normalize x, y, w, h (Labels)

YOLO Bounding Box 2



YOLO and Regression

YOLO Loss Function - Regression!

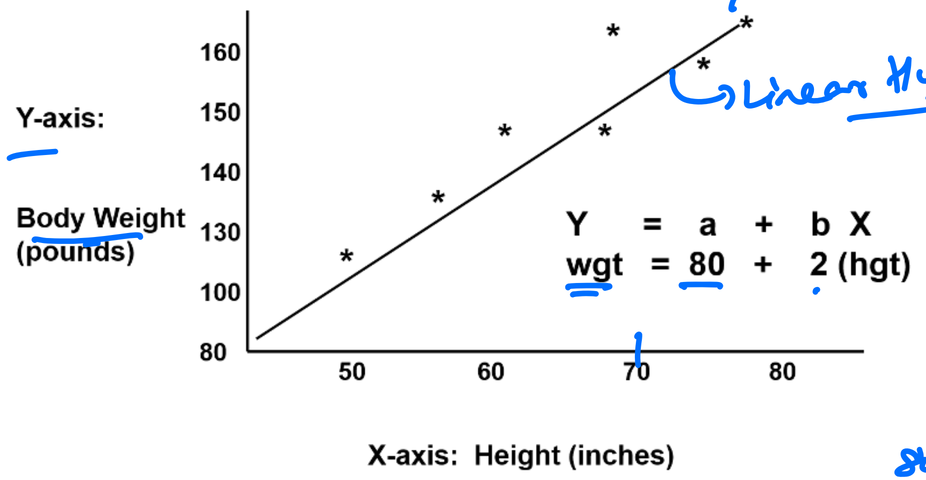
YOLO loss function turns out to be just like a Regression Loss! Why Regression?

YOLO and Regression

YOLO Loss Function - Regression!

YOLO loss function turns out to be just like a Regression Loss! Why Regression?

Linear Regression Classic Example



Linear Regression between 2 variables/ 2 features

Linear Hyper-plane

80+2×70 = 220 pounds

ICE #3

Regression (2 mins)

You want to predict the 'sharpness' of an image when the input is an image. Sharpness for this exercise is defined on a continuous scale between 0 and 1. The training data looks like $\{Image, Sharpness\}$ where Image is the input and Sharpness (on a continuous scale) is the output. You devise an ingenious loss function as follows: Take the prediction $\hat{y}_i$ of the sharpness, subtracts it from the ground truth sharpness y_i, and obtain the error, e_i. Define the loss, $L = \sum_i e_i$. You then minimize the loss as you hope a good model for sharpness would give zero errors and hence a close to zero loss. Optimizing the loss function:

1. Will help you train a good model for sharpness
2. Is a good idea but may have to watch out for overfitting
3. Would not be a good idea
4. Could result in a model with overall zero error but poor individual predictions

YOLO Loss Function

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

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

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2$$

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

Today

1. Object Detection Recap
2. R-CNN variants - Fast and Faster R-CNN
3. YOLO - Single Stage Object Detection
4. **Results and Benchmarking on the data sets**

Data Sets

Dataset	Classes	Train			Validation			Test
		Images	Objects	Objects/Image	Images	Objects	Objects/Image	
PASCAL VOC 12	20	5,717	13,609	2.38	5,823	13,841	2.37	10,991
MS-COCO	80	118,287	860,001	7.27	5,000	36,781	7.35	40,670
ILSVRC	200	456,567	478,807	1.05	20,121	55,501	2.76	40,152
OpenImage	600	1,743,042	14,610,229	8.38	41,620	204,621	4.92	125,436

Results for Object Detection

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	~0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	~0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	~0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	~3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	~4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
Swin-L	2021	HTC++	-	57.70%	-	-

[a]Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.Rows colored gray are real-time detectors (>30 FPS).

v5! → 2020

Breakout for Takeaways!

Discuss Takeaways (5 mins)

From today's lecture in your zoom group

Next Lecture

1. Newer Variants of YOLO
2. Object Detection vs Instance Segmentation
3. Image Captioning Models