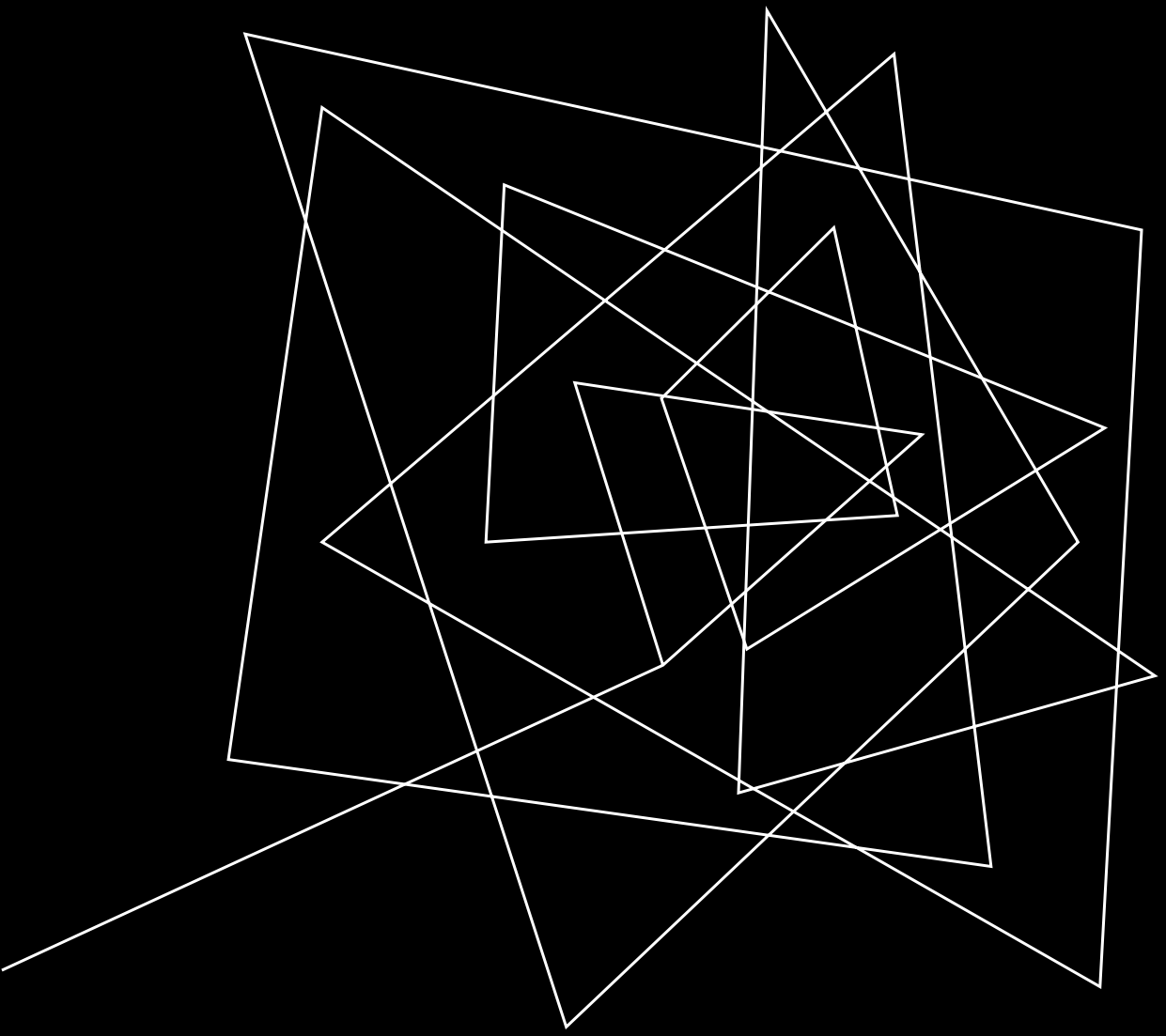
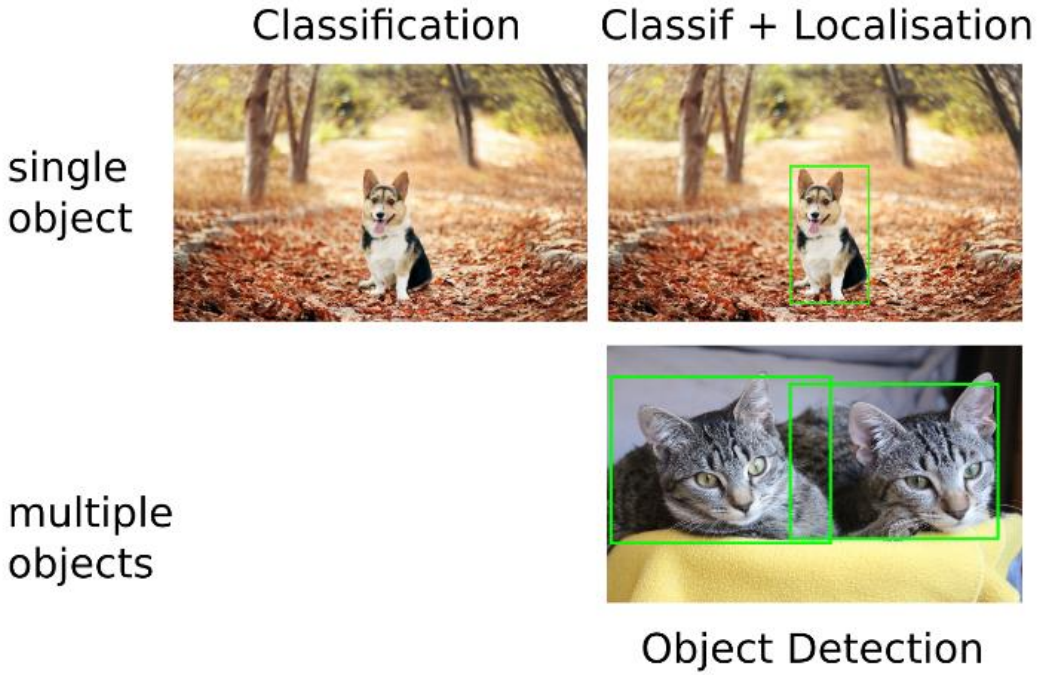


YOLO FOR OBJECT DETECTION

YOLO and its variants for object detection

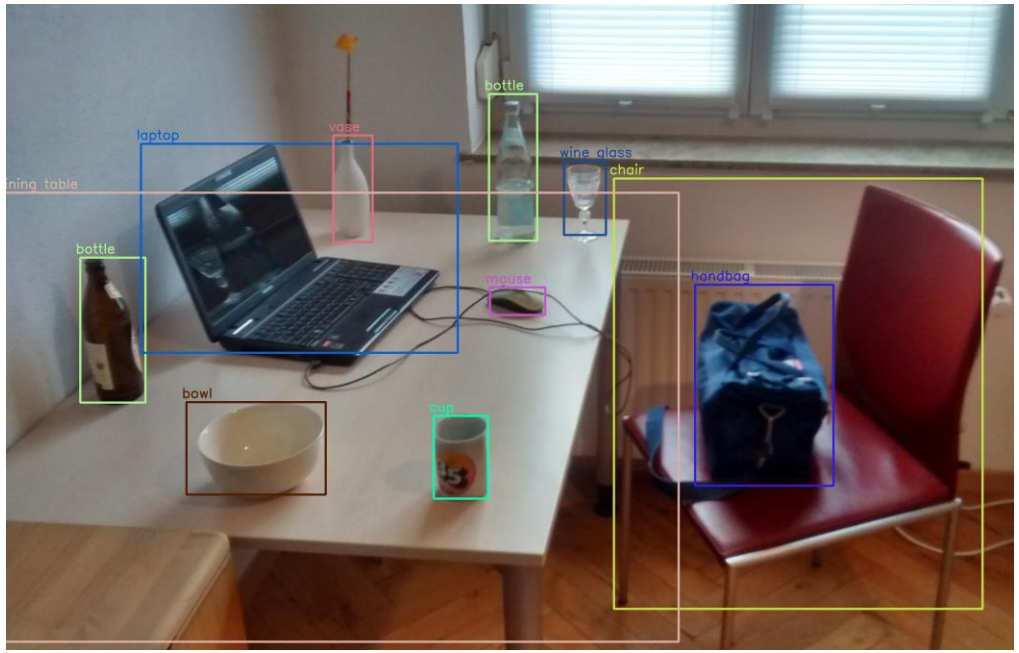
Introduction: Beyond Image Classification

- **Previous lecture:** Image classification using CNNs
- **Limitations**
  - Mostly on centered images
  - Only a single object per image
  - Not enough for many real-world vision tasks



Introduction: What is Object Detection?

- Object detection is a computer vision technique for locating and identifying instances of objects in **images** or **videos**



Introduction: Object Detection

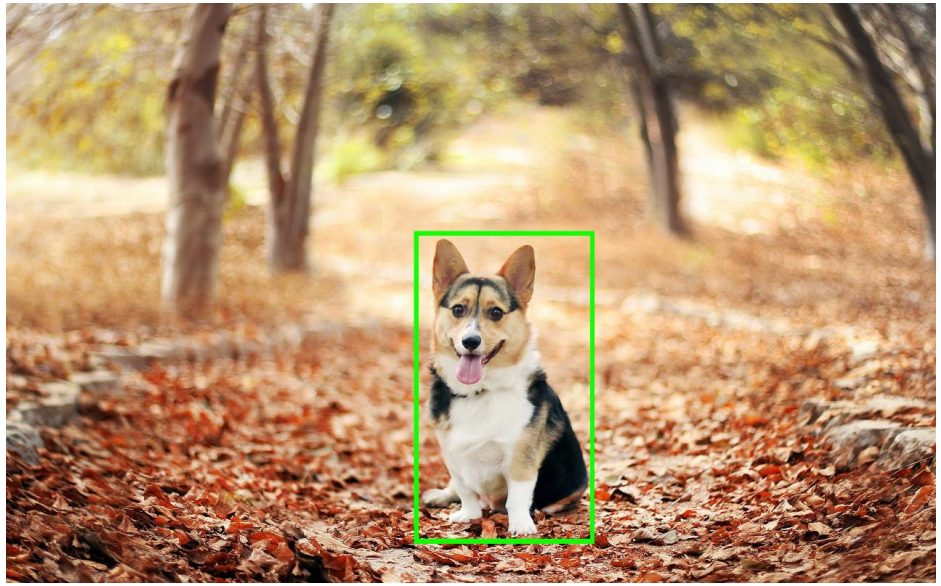
- **Object detection** is a technique used in computer vision for the identification and localization of objects within an image or a video.
- **Image Localization** is the process of identifying the correct location of one or multiple objects using bounding boxes, which correspond to rectangular shapes around the objects.

This process is sometimes confused with image classification or image recognition, which aims to predict the class of an image or an object within an image into one of the categories or classes.

Source: YOLO Object Detection Explained

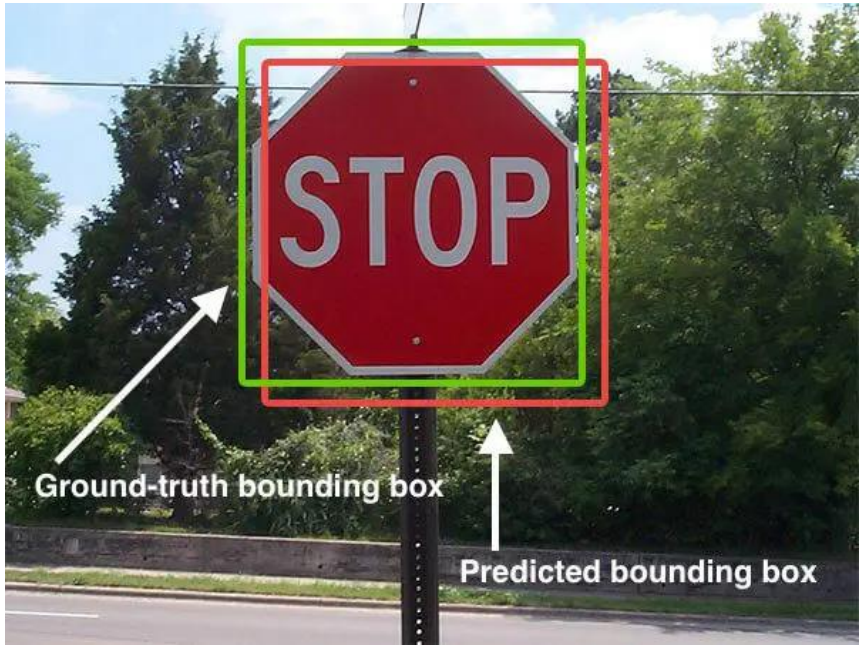
Introduction: Localization

- Single object per image
- Predict coordinates of a bounding box (x, y, w, h)
- Evaluate via Intersection over Union (IoU)

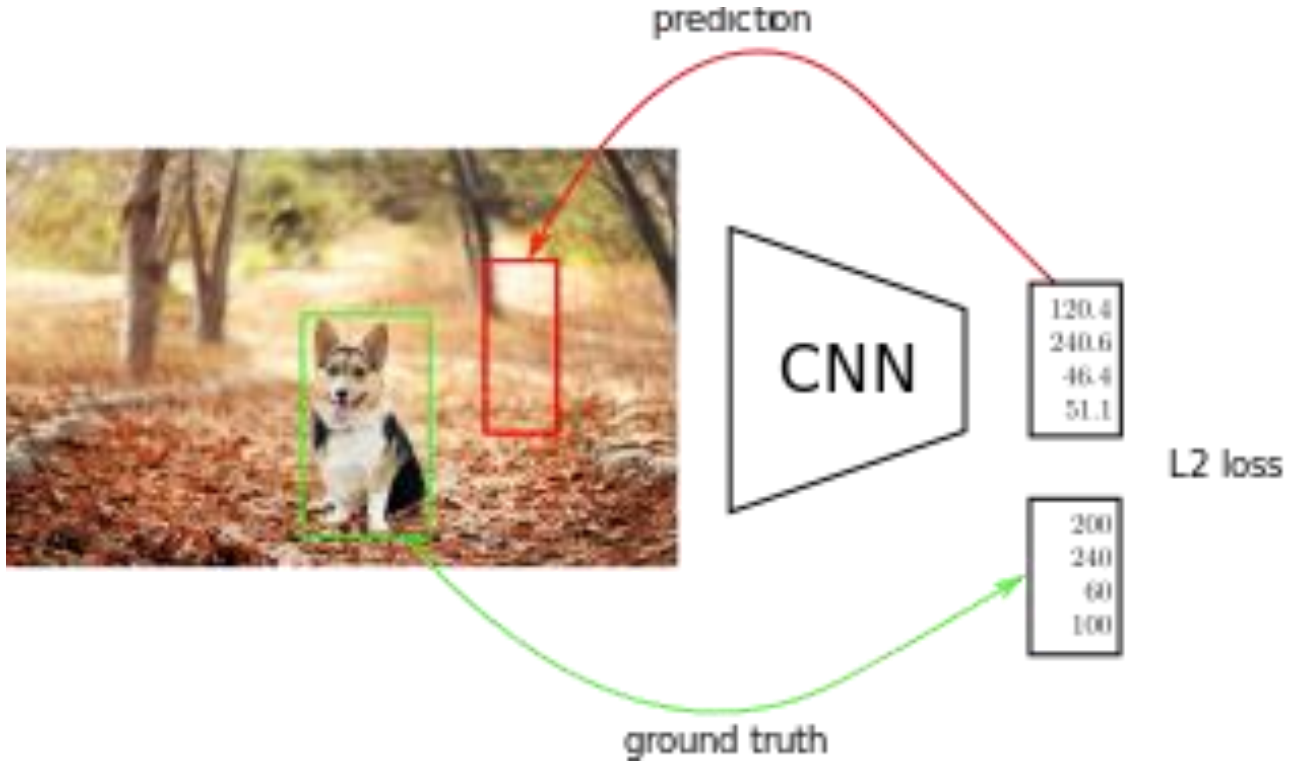


Introduction: Localization

Evaluate via Intersection over Union (IoU): Intersection over Union (IoU) is used to evaluate the performance of object detection by comparing the ground truth bounding box to the predicted bounding box.

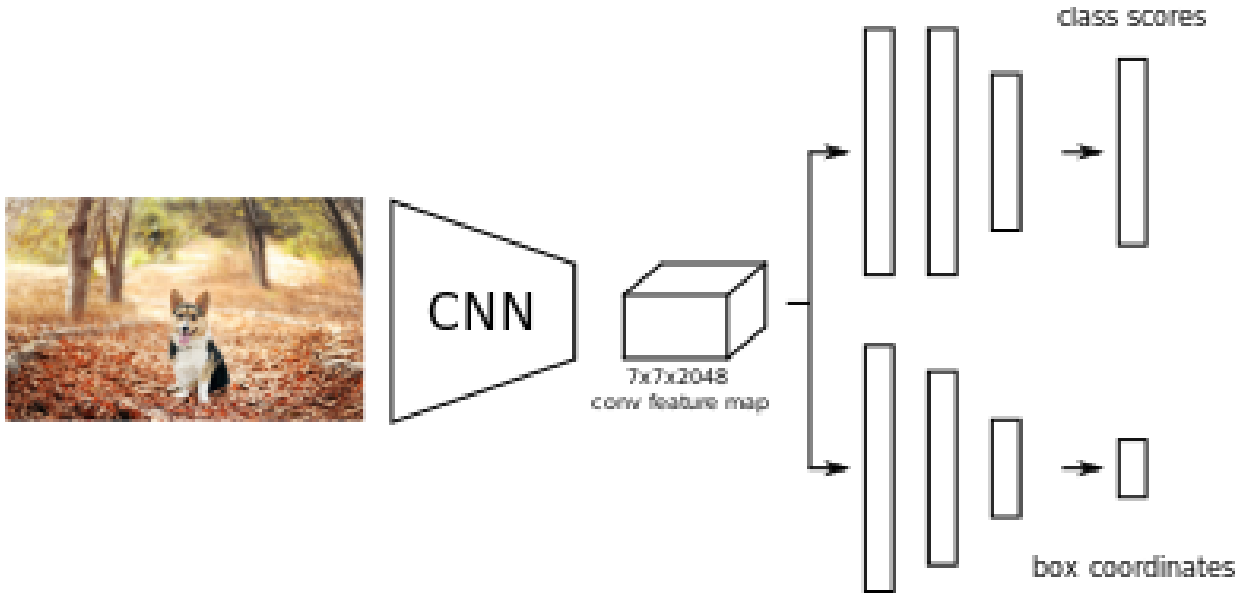


Introduction: Localization as Regression



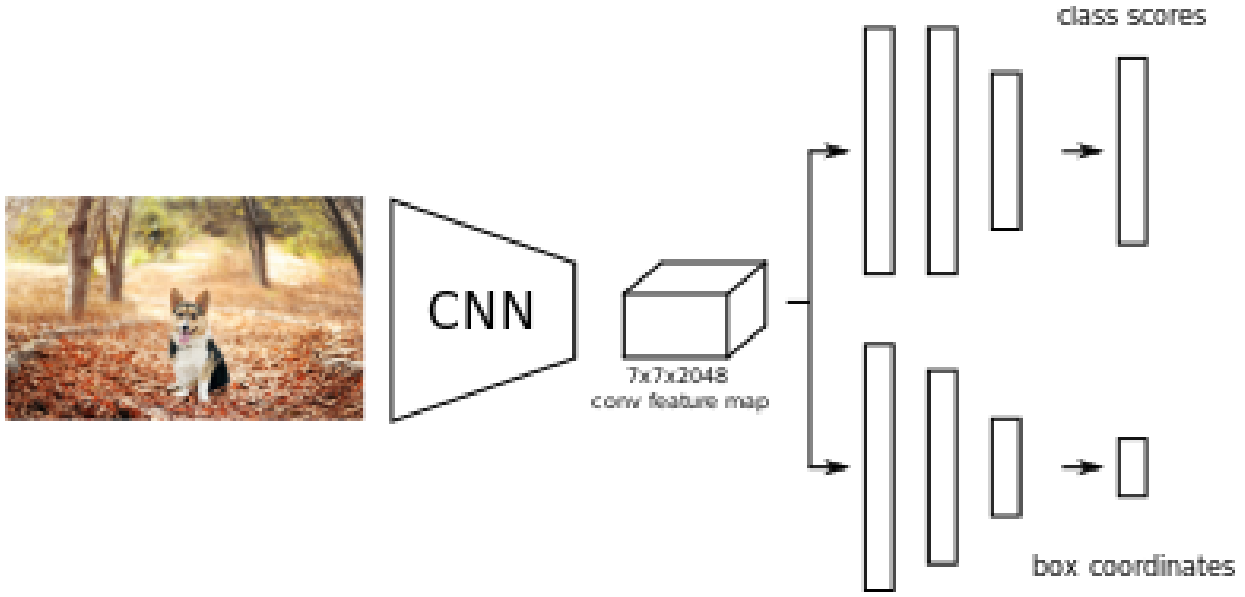
Introduction: Classification + Localization

- Use a pre-trained CNN on ImageNet (ex. ResNet)
- The "localization head" is trained separately with regression
- Possible end-to-end finetuning of both tasks
- At test time, use both heads



Introduction: Classification + Localization

- **C** classes, 4 output dimensions (1 box)
- Predict exactly N objects: predict (N×4) coordinates and (N×K) class scores



Introduction: Object Detection

Image Classification: Is this a cat or a dog?

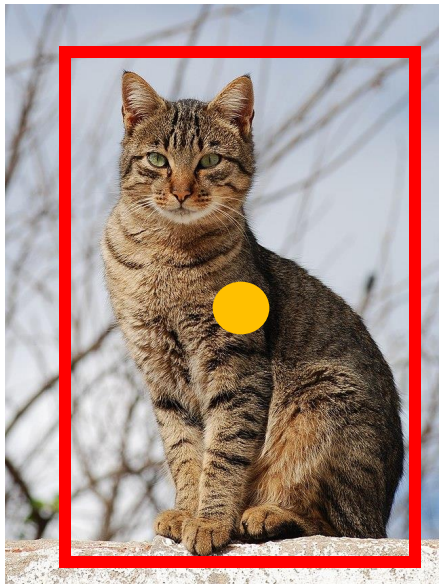
Output: Dog = 0 | Cat = 1

Object Detection: Where exactly is the cat in this image?

Output: Dog = 0 | Cat = 1 + Bounding Box (Red)

Introduction: Object Detection

Object Detection and Localization

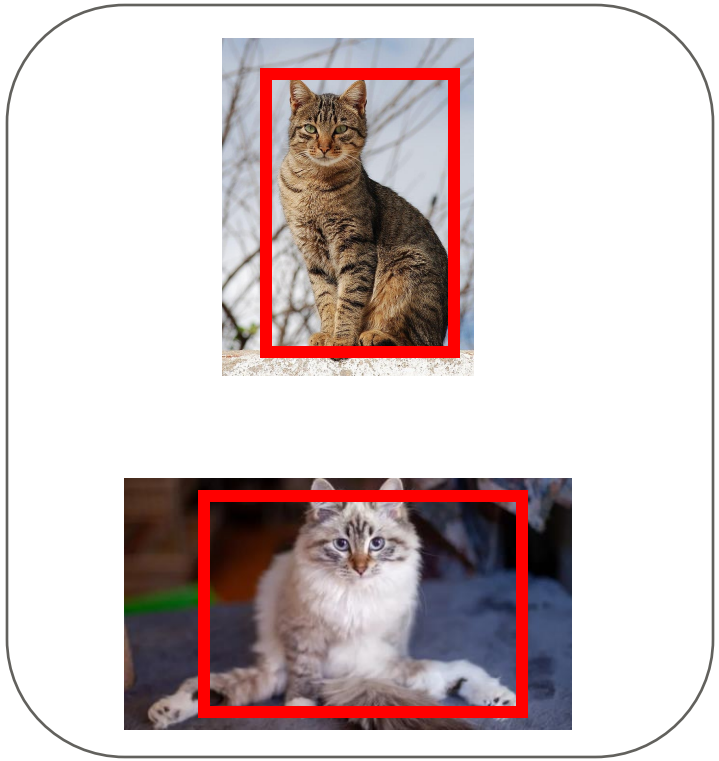


Output

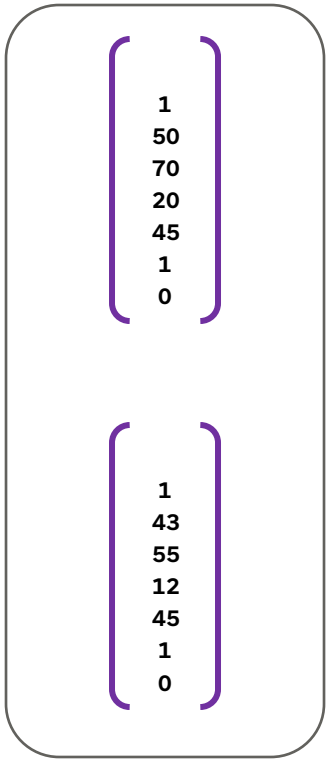
C_1 = Cat
C_2 = Dog

P_c	1
B_x	50
B_y	70
B_w	20
B_h	45
C_1	1
C_2	0

Introduction: Object Detection



X_train



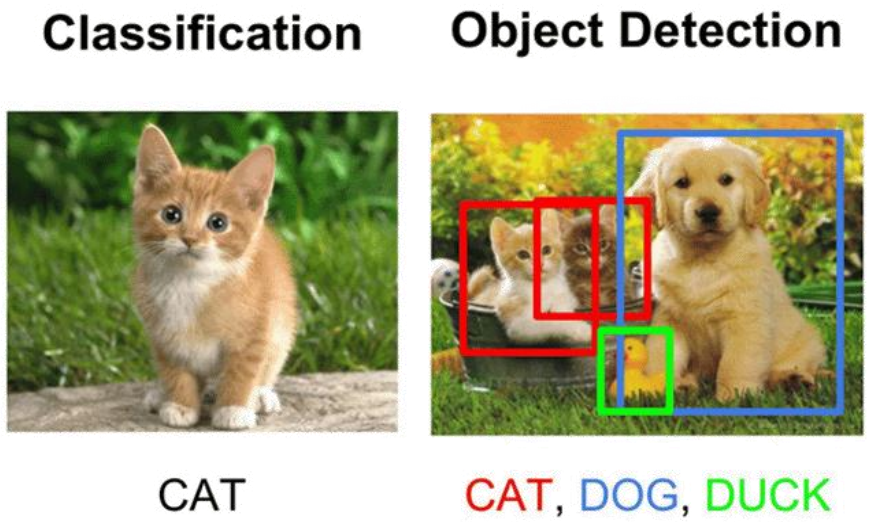
y_train

CNN

P_c
B_x
B_y
B_w
B_h
C_1
C_2

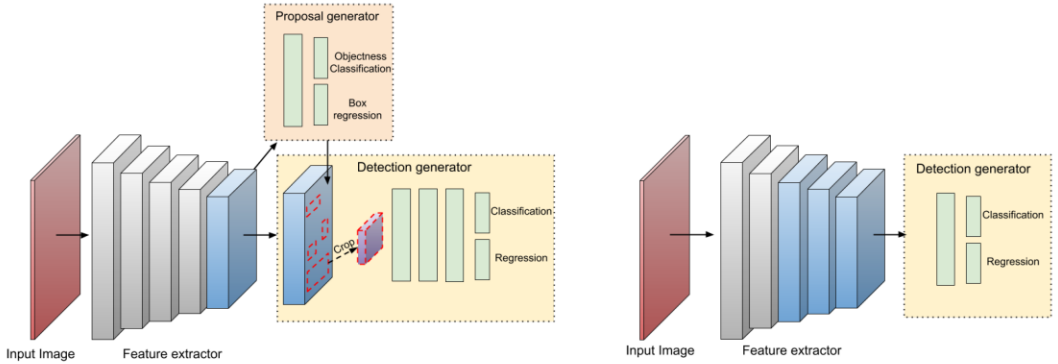
Introduction: Object Detection

- **We don't know in advance the number of objects in the image.**
  - Unlike image classification, where the goal is to assign a single label to an entire image, object detection is concerned with identifying multiple objects and their precise locations within an image. The number of objects in an image can vary, and we don't have prior information about how many there are.
- **Object detection** relies on **object proposal** and **object classification**
  - **Object detection** involves two main steps: proposing regions in the image where objects might be located (object proposal), and then classifying these proposed regions to determine what objects are present (object classification).
  - **Object proposal:** find regions of interest (**RoIs**) in the image
  - **Object classification:** classify the object in these regions
- **Two main families:**
  - Single-Stage: A grid in the image where each cell is a proposal (SSD, YOLO, RetinaNet)
  - Two-Stage: Region proposal then classification (Faster-RCNN)



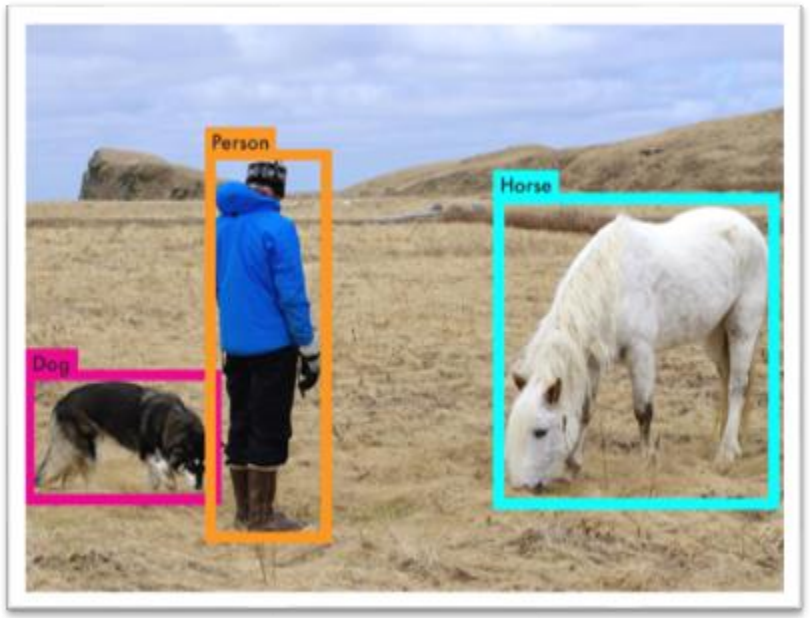
Introduction: Single-Stage vs. Two-Stage Object Detection

- **Single-Stage:**
  - In single-stage detectors, the process of proposing regions and classifying objects is done simultaneously in a single step. This means that a grid is applied to the entire image, and each cell in the grid is responsible for both proposing regions and classifying objects. Examples of single-stage detectors include SSD (Single Shot MultiBox Detector), YOLO (You Only Look Once), and RetinaNet.
- **Two-Stage:**
  - In two-stage detectors, the process is divided into two steps. The first stage involves generating region proposals, which are areas in the image likely to contain objects. These proposed regions are then passed to the second stage, which performs the actual object classification. An example of a two-stage detector is Faster R-CNN (Region-based Convolutional Neural Network).



(a) Two-stage Faster R-CNN (b) One-stage RetinaNet

YOLO: You Only Look Once



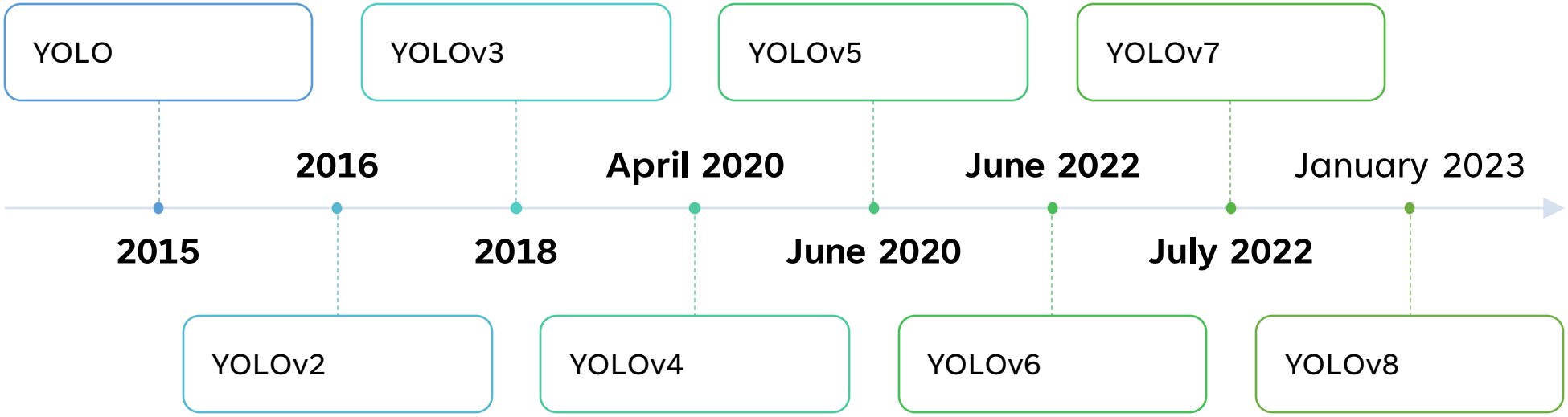
Paper Title: You Only Look Once: Unified, Real-Time Object Detection

YOLO: You Only Look Once

You Only Look Once (YOLO) is a **state-of-the-art**, **real-time** object detection algorithm introduced in **2015** by **Joseph Redmon**, **Santosh Divvala**, **Ross Girshick**, and **Ali Farhadi** in their famous research paper "**You Only Look Once: Unified, Real-Time Object Detection**".
The authors frame the object detection problem as a **regression** problem instead of a **classification** task by spatially separating bounding boxes and associating probabilities to each of the detected images using a single convolutional neural network (CNN).

YOLO for Object Detection: State-of-the-art

YOLO for Object Detection: State-of-the-art



Ultralytics YOLOv8:
[The State-of-the-Art YOLO Model]

YOLO for Object Detection: What Makes YOLO Popular for Object Detection?

Real-time Detection:

- YOLO is designed for real-time object detection. It can process images and provide object detections in a single pass, making it significantly faster than many other object detection algorithms.

Single-shot Detection:

- Unlike traditional object detection methods that involve multiple stages (like region proposal networks followed by classification and refinement), YOLO is a single-shot detector. This means it predicts bounding boxes and class probabilities directly from the entire image, which leads to faster inference times.

Efficiency and Speed:

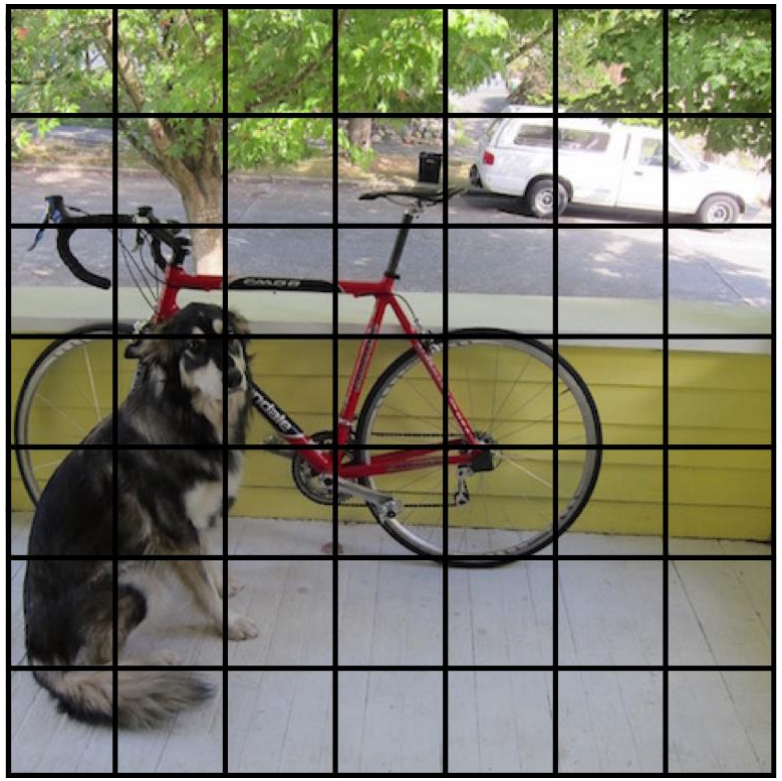
- YOLO's architecture is efficient and can be optimized for deployment on various hardware platforms, including CPUs, GPUs, and even specialized hardware like TPUs (Tensor Processing Units).
- YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

Better generalization:

- This is especially true for the new versions of YOLO. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.
- For instance, the Automatic Detection of Melanoma with Yolo Deep Convolutional Neural Networks paper shows that the first version YOLOv1 has the lowest mean average precision for the automatic detection of melanoma disease, compared to YOLOv2 and YOLOv3.

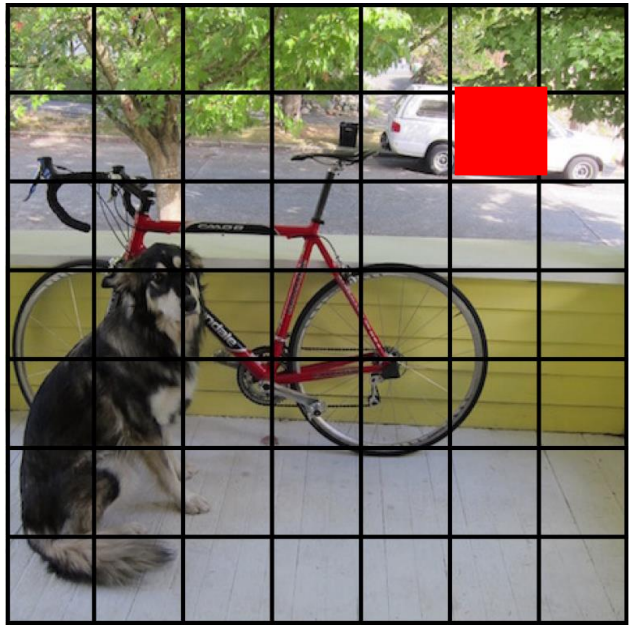
YOLO for Object Detection: Understanding YOLO

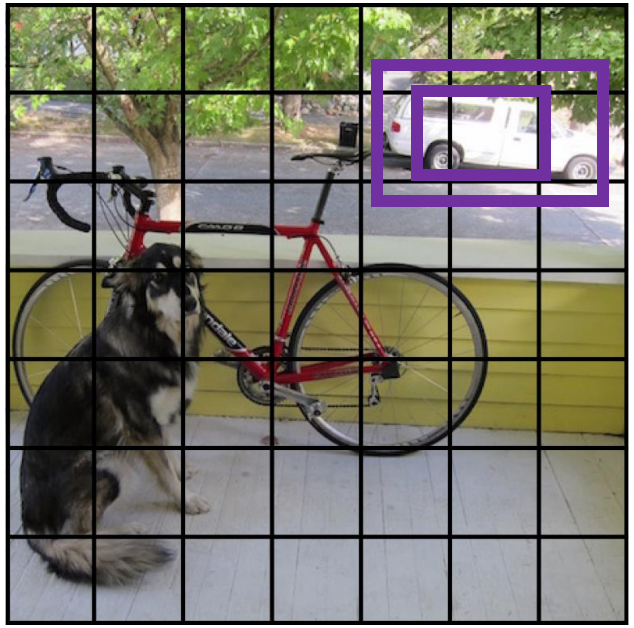
We split the image into an S*S grid (S = 7)



YOLO for Object Detection: Understanding YOLO

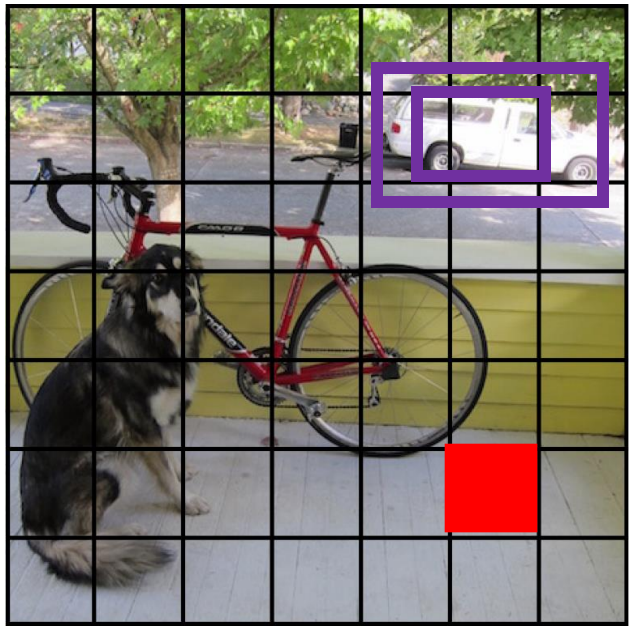
Each cell predicts boxes and confidences: P(object)

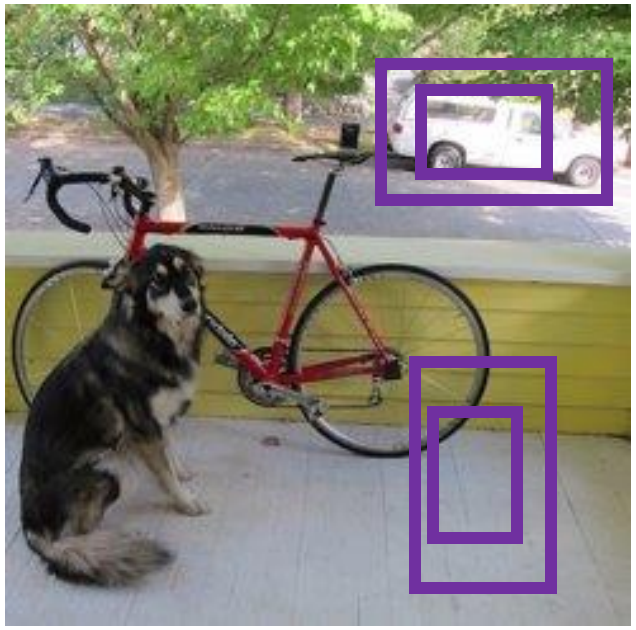




YOLO for Object Detection: Understanding YOLO

Each cell predicts boxes and confidences: P(object)





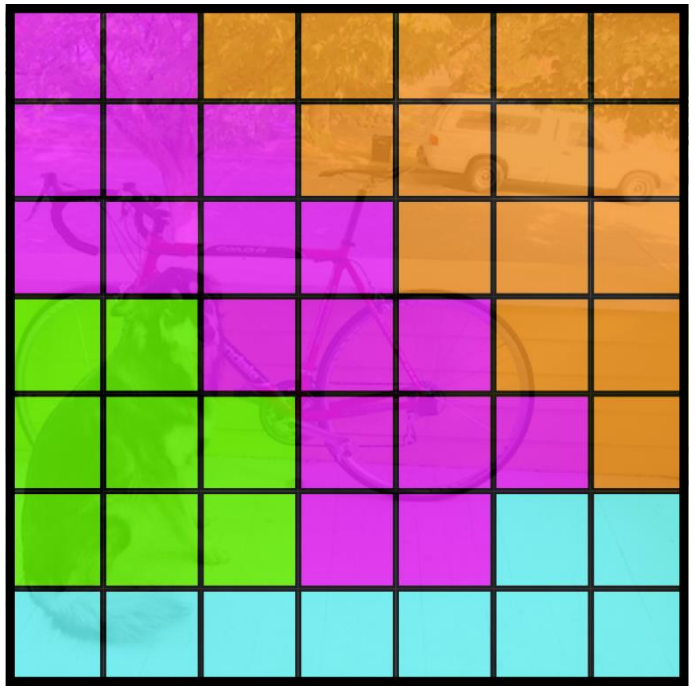
YOLO for Object Detection: Understanding YOLO

Each cell predicts boxes and confidences: P(object)

YOLO for Object Detection: Understanding YOLO

Each cell predicts a class probability

Bicycle



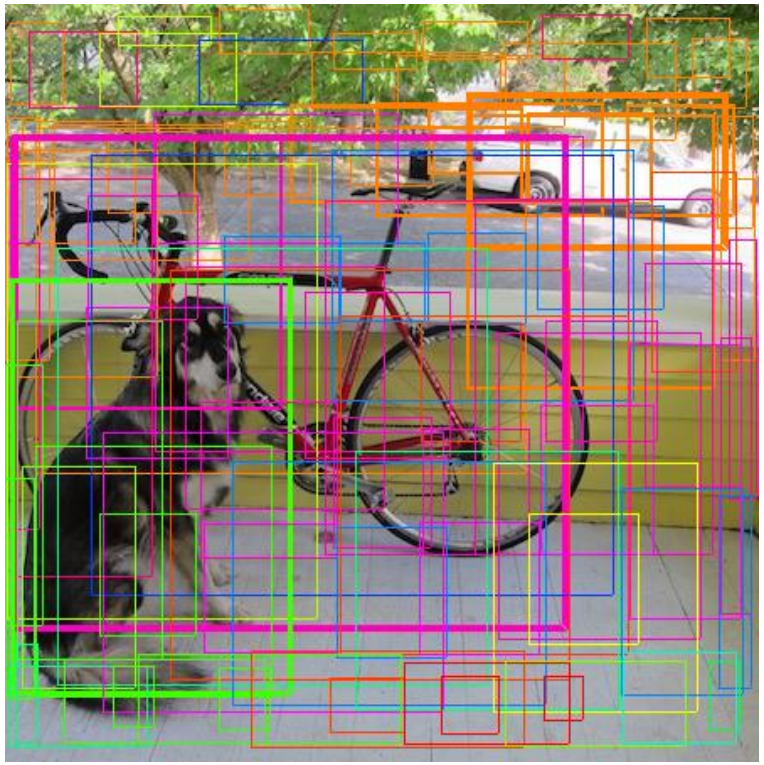
Car

Dog

Dining table

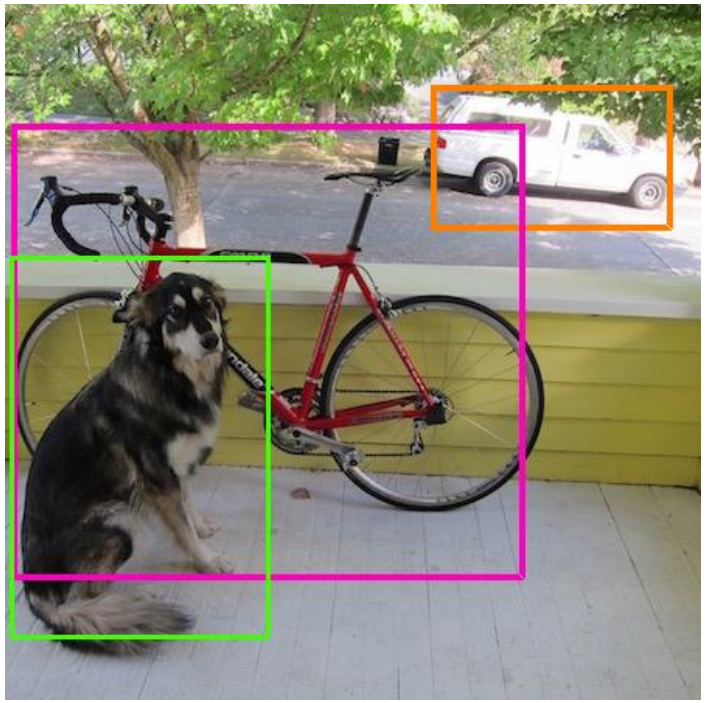
YOLO for Object Detection: Understanding YOLO

Then we combine the box and class predictions

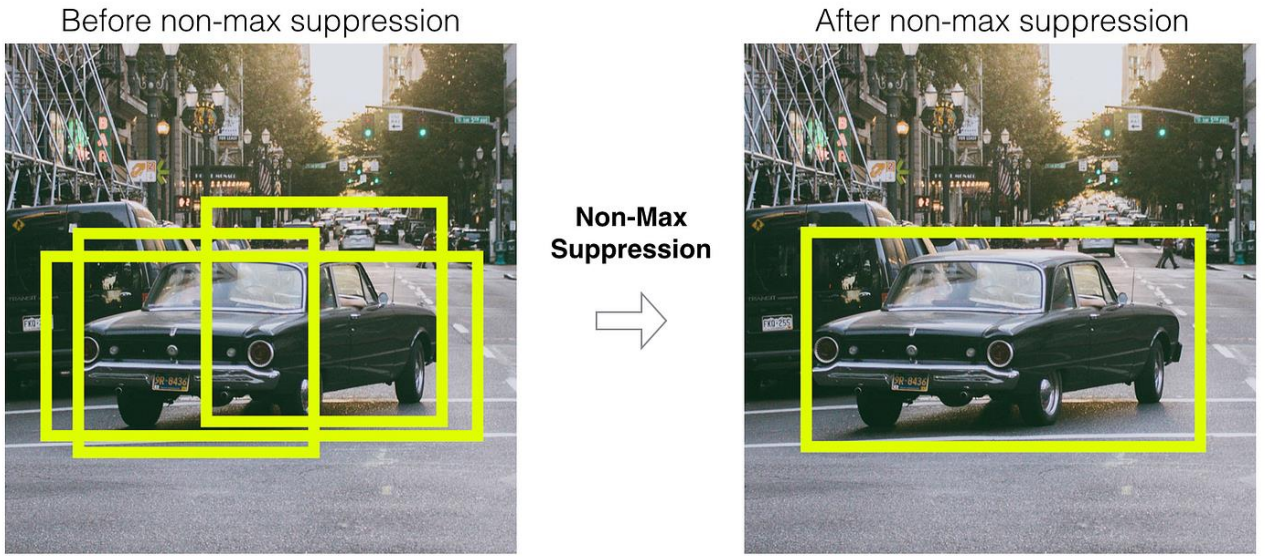


YOLO for Object Detection: Understanding YOLO

Finally, we do NMS and threshold detections



YOLO for Object Detection: Understanding YOLO



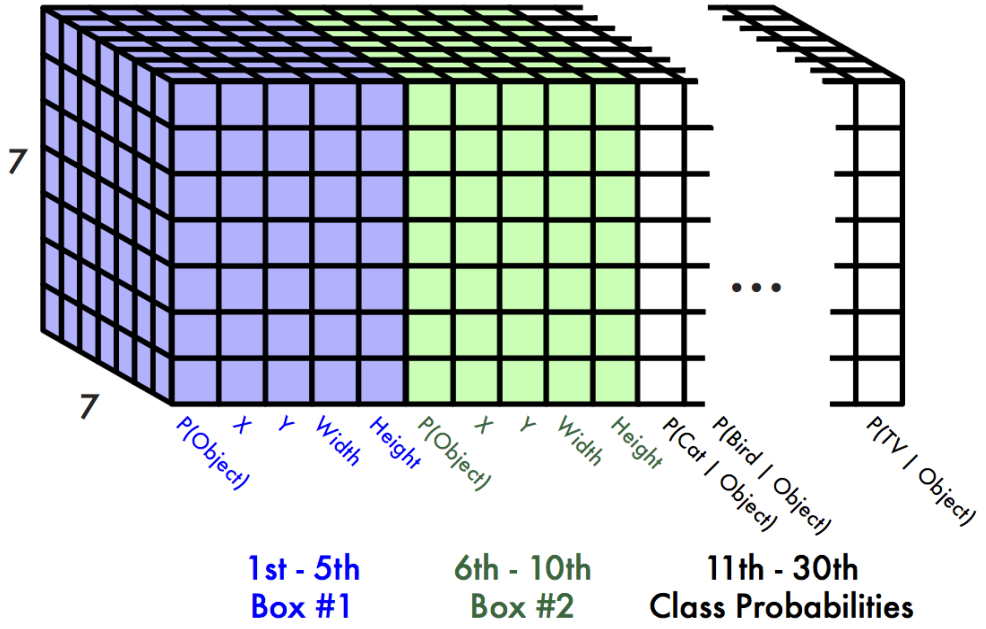
YOLO for Object Detection: Understanding YOLO

This parameterization fixes the output size

Each cell predicts:
- For each bounding box:
- 4 coordinates (x, y, w, h)
- 1 confidence value
- Some number of class probabilities

For Pascal VOC:
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**



YOLO for Object Detection: Understanding YOLO

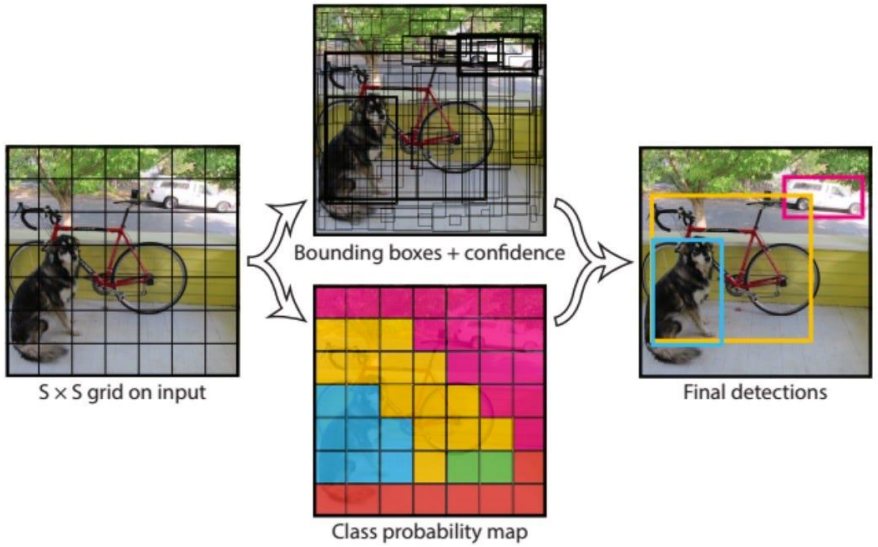
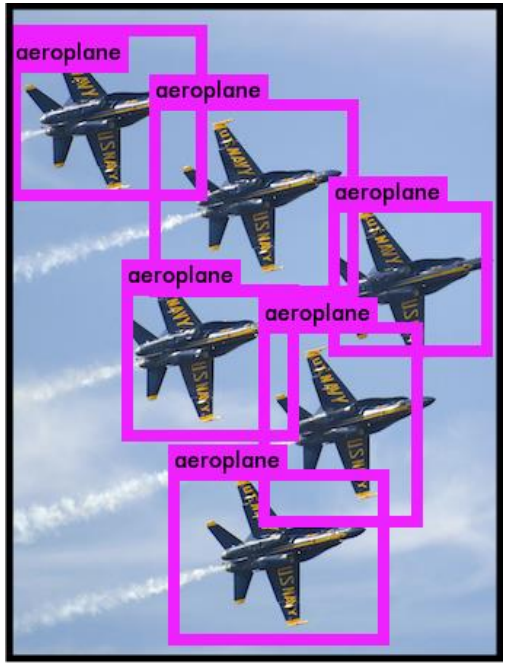
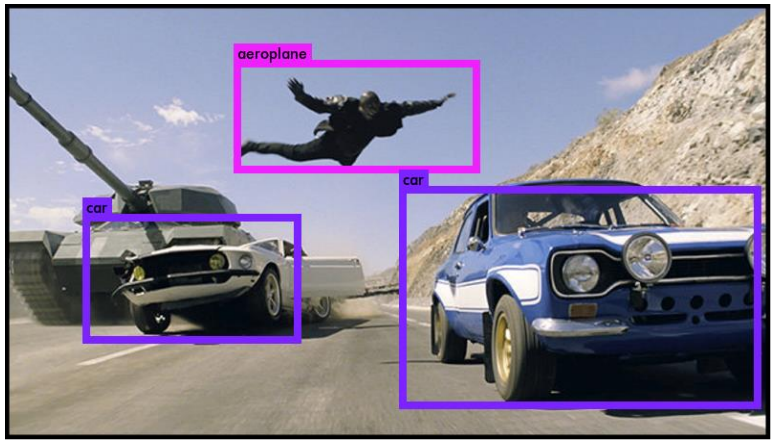
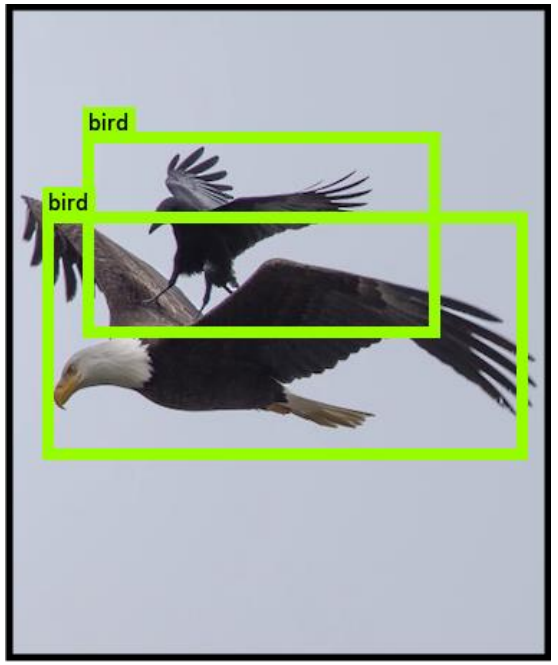


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

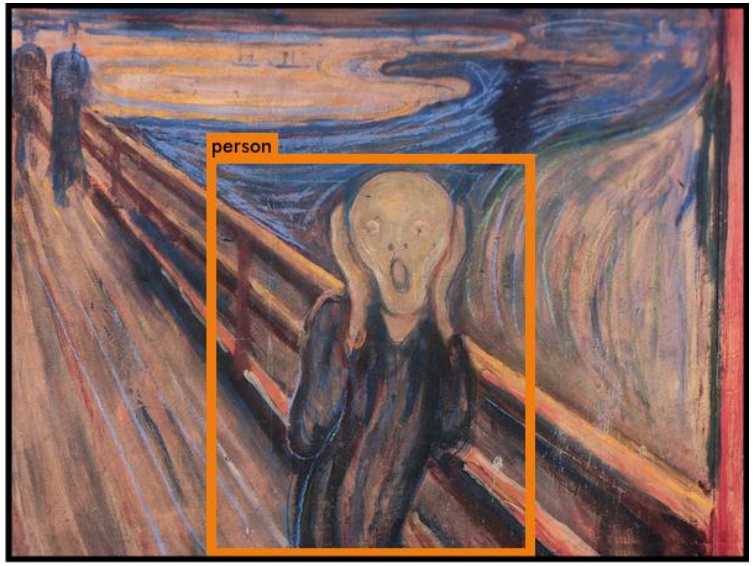
YOLO for Object Detection: YOLO works across a variety of natural images







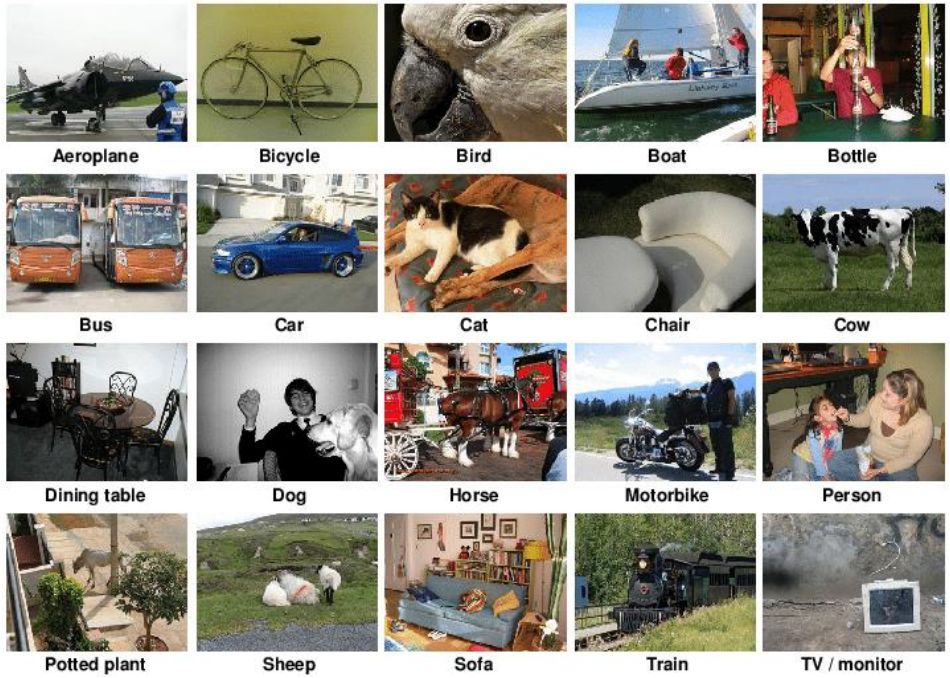
YOLO for Object Detection: It also generalizes well to new domains (like art)



YOLO for Object Detection: Understanding YOLO <Training Dataset>

Link to the paper: The PASCAL Visual Object Classes Challenge: A Retrospective

- **Pascal VOC detection dataset:**
  - The PASCAL VOC (Visual Object Classes) dataset is another widely used benchmark in computer vision, specifically for object detection, segmentation, and classification tasks. It was created to encourage the development and evaluation of algorithms for object detection and related tasks.
- 20 classes
- A total of 11500 images
- **Note:** Most of the recent papers use COCO instead of Pascal VOC



YOLO for Object Detection: Understanding YOLO <Training Dataset>

Link to the dataset: [COCO Dataset]

- **COCO Dataset:** The COCO dataset, which stands for Common Objects in Context, is a widely used benchmark in computer vision. It is designed for object recognition, segmentation, and captioning tasks. The dataset is curated by the Microsoft Research team and contains a large collection of images with objects in complex scenes.
- Over 80 object categories
- around 330,000 images

YOLO for Object Detection: Architecture and Implementation

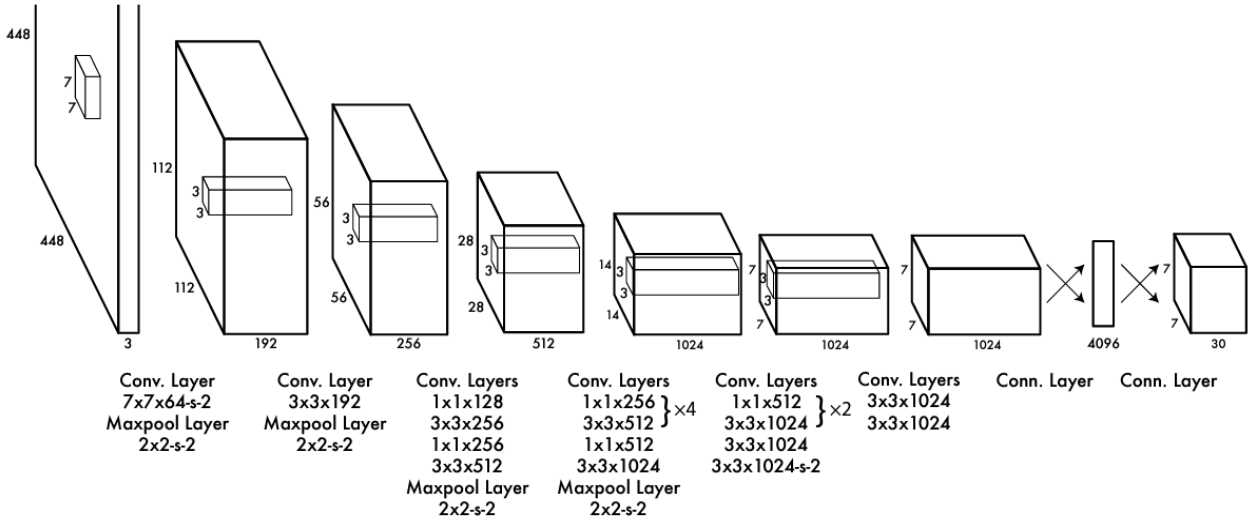
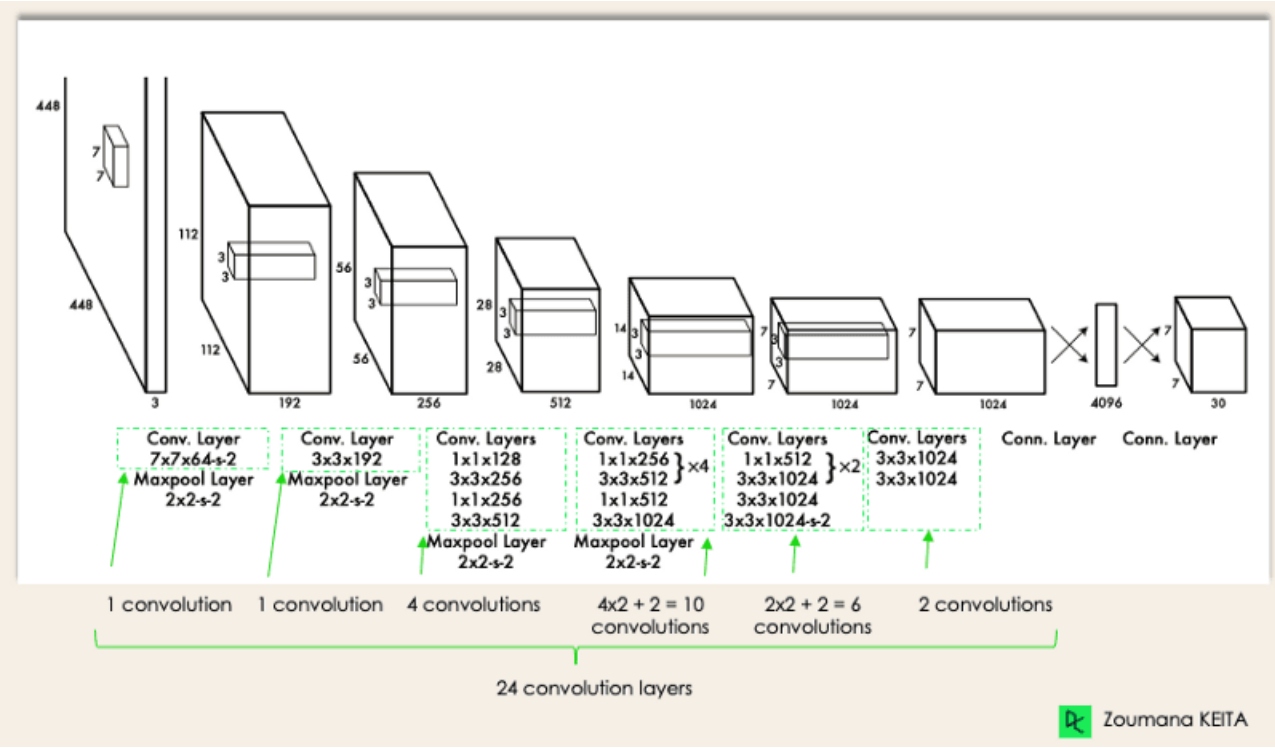


Fig. YOLO Architecture

Paper Link: [You only look once: Unified, real-time object detection]

YOLO for Object Detection: Architecture and Implementation



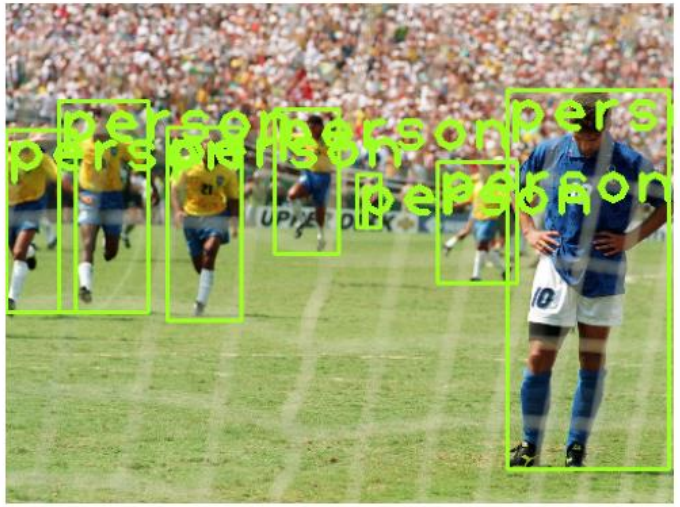
Paper Link: [You only look once: Unified, real-time object detection]

Computer Vision Libraries: OpenCV

- **OpenCV** is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.
- **OpenCV** was originally developed by **Intel** in June 2000 and has since been maintained by a community of developers. It is written in C++ and has bindings for various programming languages including Python, Java, and MATLAB, making it accessible to a wide range of developers.

OpenCV and YOLOv3 for Object Detection

Demo: Session 4- YOLO for Object Detection



Tutorial: YOLO Object Detection

YOLO for Object Detection: TO-DO Project

Project: [Fruit and Vegetable Detection using YOLO]