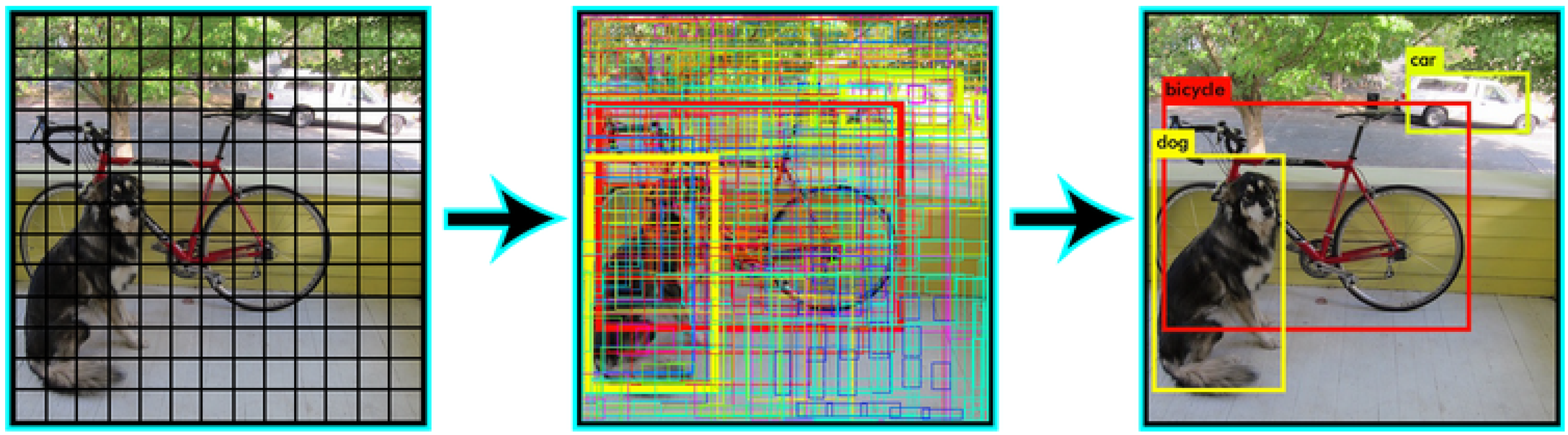
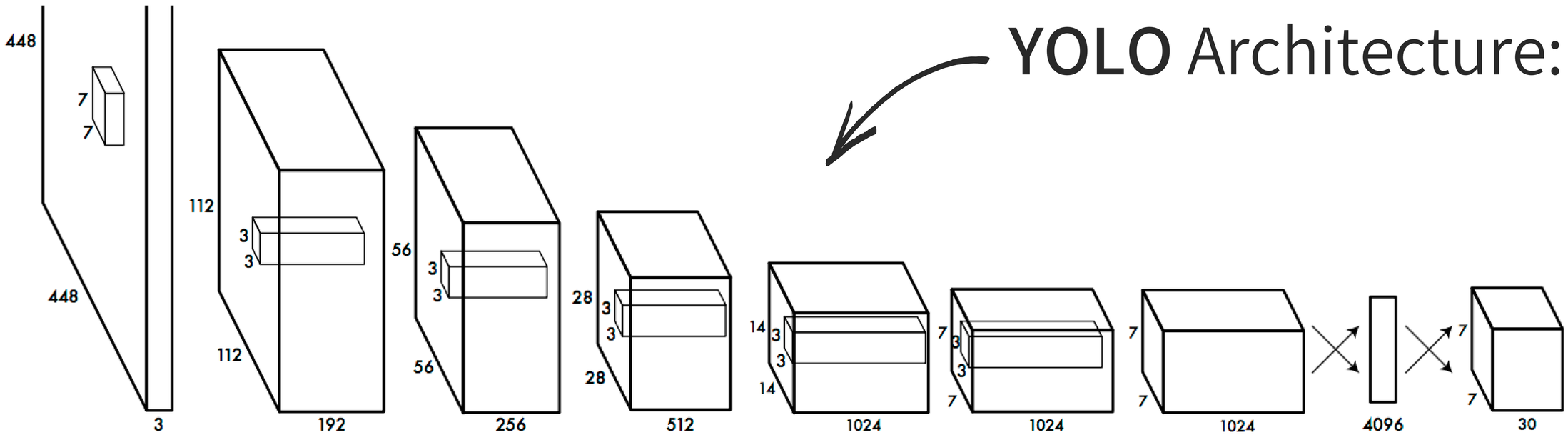
YOLO (You only look once)

YOLO is the **state-of-the-art, real time system** built on deep learning for solving **object detection** problems.



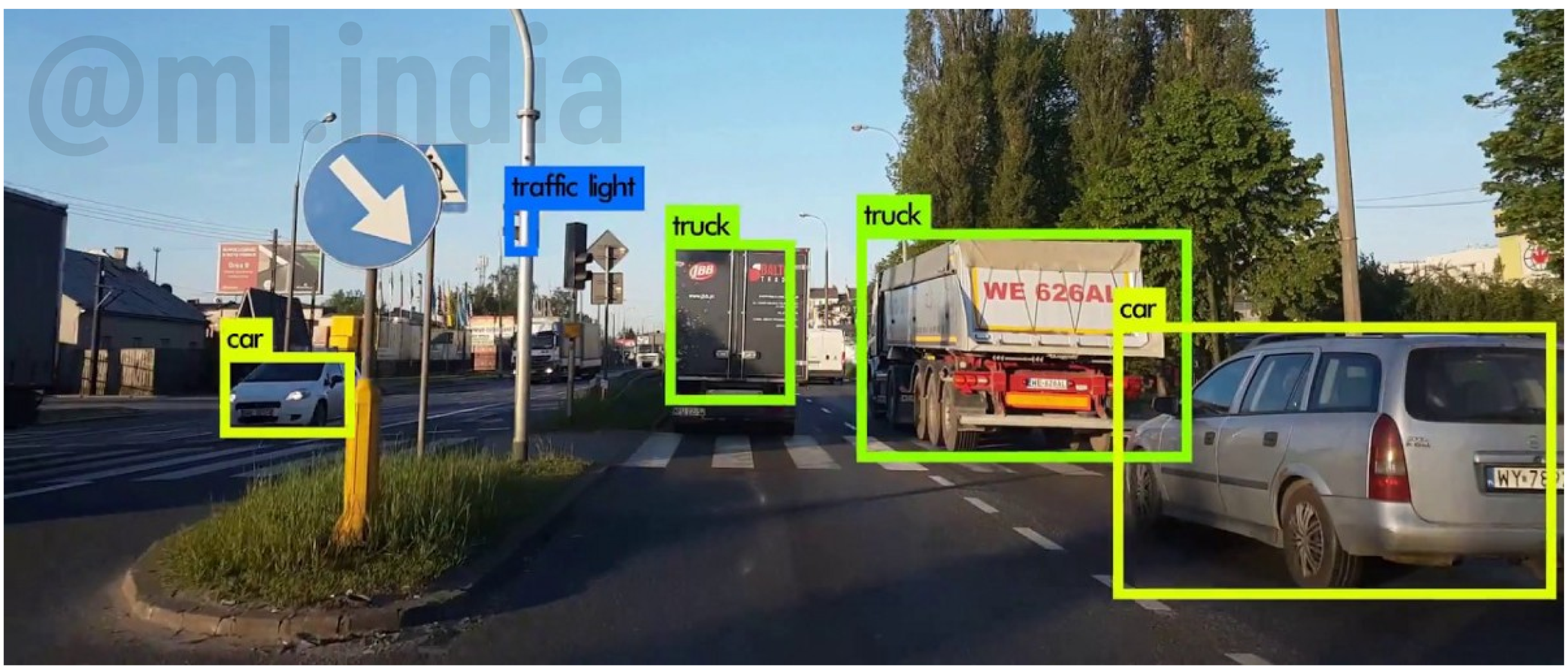
YOLO Architecture:



❤ Hit to support!

Swipe. →

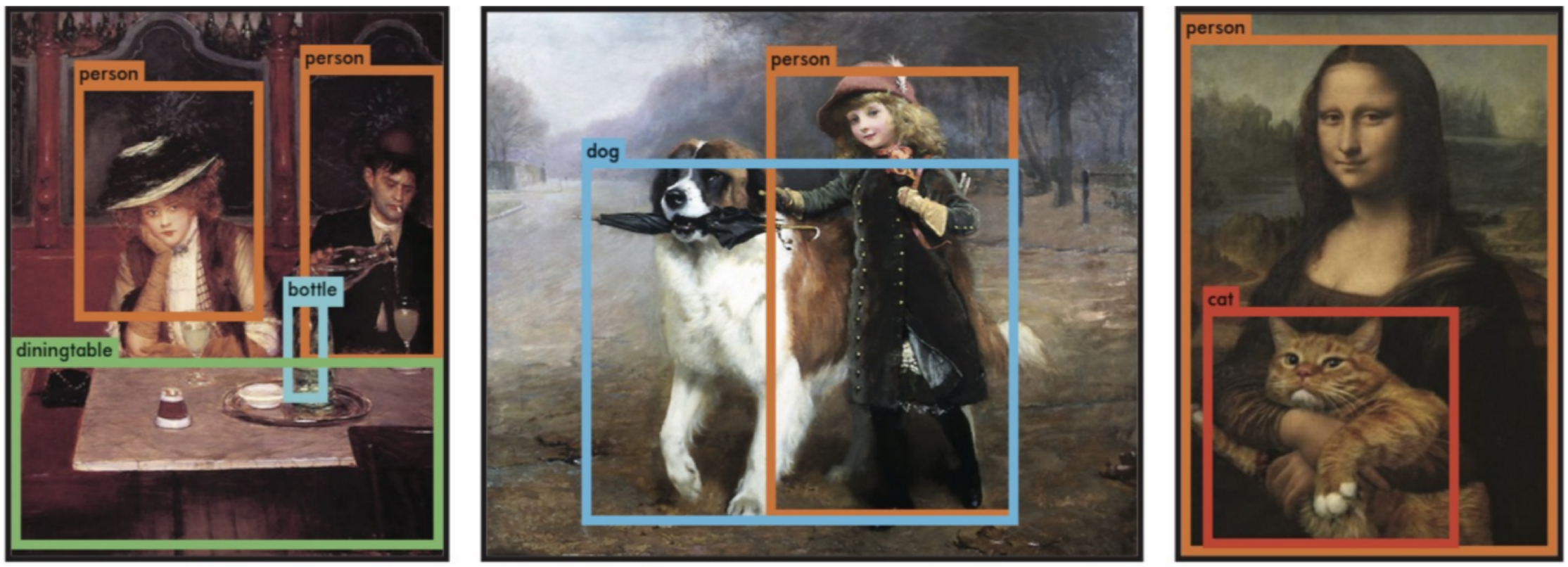
As show in the first image on the previous slide, the algorithm first divides the image into **defined bounding boxes**, then runs a recognition algorithm in parallel for all of these to **identify which object class** they belong to, lastly it goes on to **merging these boxes** intelligently to form optimal bounding boxes around the objects.



Hit to support!

Save for later!

YOLO looks at the whole image at test time so its predictions are informed by **global context** in the image.



Previous methods, like R-CNN and its variations, used a pipeline to perform this task in multiple steps. This can be **slow to run** and also **hard to optimize**, because each individual component must be trained separately. YOLO, does it all with a **single neural network**.

Hit to support! **Save for later!**

The training:

The authors describe the training in the following way:

- First, pretrain the first 20 convolutional layers using the ImageNet 1000-class competition dataset, using a input size of 224x224.
- Then, increase the input resolution to 448x448.
- Train the full network for about 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005.
- Learning rate schedule: for the first epochs, the learning rate was slowly raised from 0.001 to 0.01. Train for about 75 epochs and then start decreasing it.
- Use data augmentation with random scaling and translations, and randomly adjusting exposure and saturation.

The loss:

YOLO predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, we only want one of them to be responsible for the object. For this purpose, we select the one with the highest **IoU (intersection over union) with the ground truth**.

Each prediction gets better at predicting certain sizes and aspect ratios. YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:

- the **classification loss**.
- the **localization loss** (errors between the predicted boundary box and the ground truth).
- the **confidence loss** (the objectness of the box).

The loss function:

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

Localization loss.

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

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

Confidence loss.

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

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

Classification loss.

Overwhelmed? Don't be! Check this amazing blog for the **mathematical details**: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

The original **YOLO paper**: You Only Look Once: Unified, Real-Time Object Detection by Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi (2015).

Link: https://pjreddie.com/media/files/papers/yolo.pdf

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

Joseph Redmon
University of Washington
pjreddie@cs.washington.edu

Santosh Divvala
Allen Institute for Artificial Intelligence
santoshd@allenai.org

Ross Girshick
Facebook AI Research
rbg@fb.com

Ali Farhadi
University of Washington
ali@cs.washington.edu

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is far less likely to predict false detections where nothing exists. Finally, YOLO learns very general representations of objects. It outperforms all other detection methods, including DPM and R-CNN, by a wide margin when generalizing from natural images to artwork on both the Picasso Dataset and the People-Art Dataset.

1. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate, algorithms for object detection would allow computers to drive cars in any weather without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [10].

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding box, eliminate duplicate detections, and rescore the box based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new image at test

There have been many improvements proposed since then, that were combined in the newer **YOLOv2** and **YOLOv3**. Will talk about them in some another time.

Link to the code is available on YOLO's official site:
https://pjreddie.com/darknet/yolo/

Sources and references:

- https://hackernoon.com/understanding-yolo-f5a74bbc7967
- https://pjreddie.com/darknet/yolo/
- https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
- https://arxiv.org/pdf/1506.02640.pdf