Object Tracking
(with Yolo and DeepSORT)

Amirata Ghaffarian
Shervin Ghaffari

CONTENTS

1、Introduction

Object Tracking

- ***Object tracking*** *is the task of taking an initial set of object detections, creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video, maintaining the ID assignment.*

Challenges

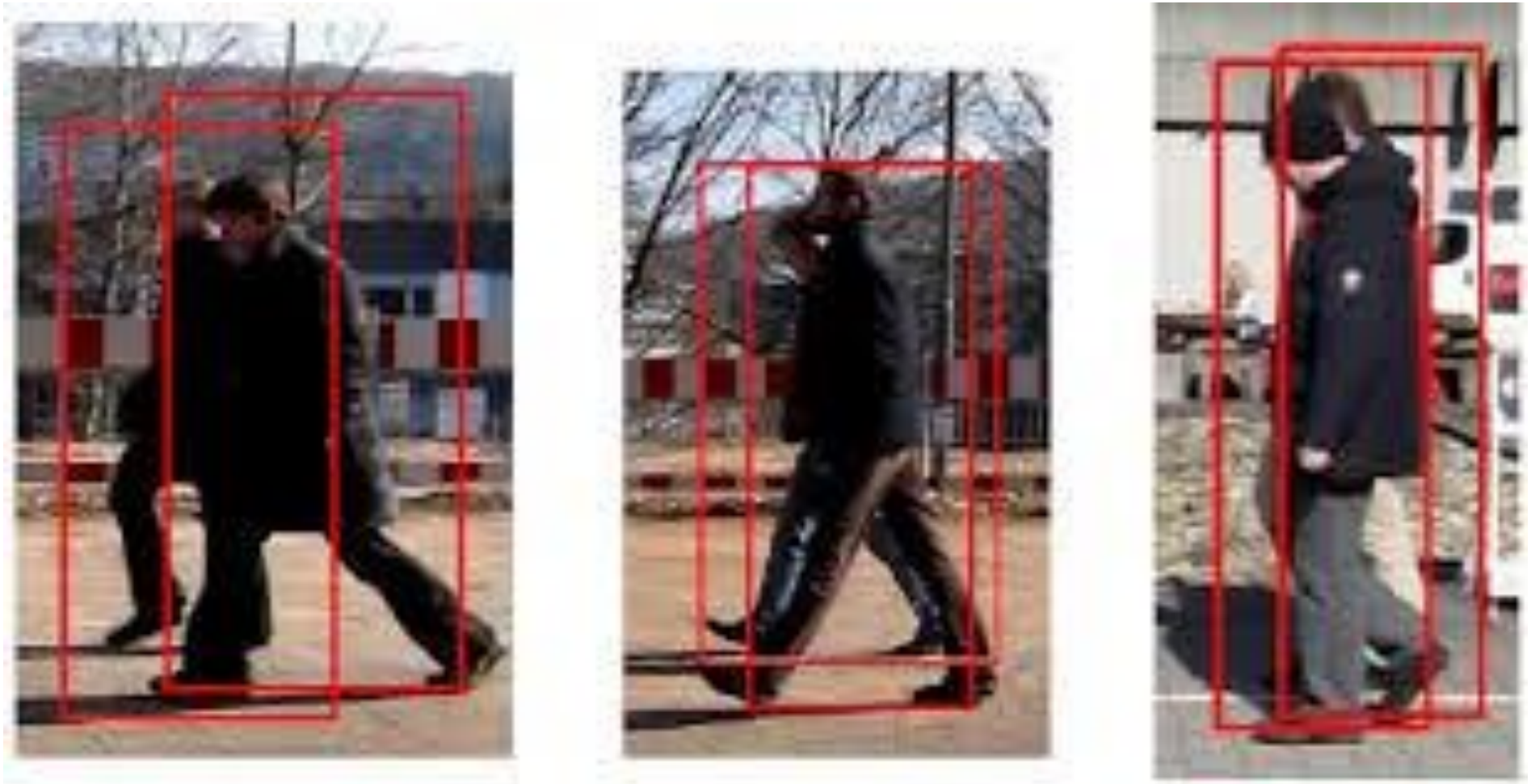
- Here are some challenges in object tracking:

1. Occlusion: It occurs when an object we are tracking is hidden (occluded) by another object. Like two persons walking past each other, or a car that drives under a bridge. The problem in this case is what you do when an object disappears and reappears again.

2. Scale change

3. Background clutter: Background near object has similar color or texture as the object. Hence, it become harder to track results for a small object with cluttered background.

4. Appearance change: Different viewpoint of an object may look very different visually and without the context. Hence, it become very difficult to identify the object using only visual detection.



Occlusion problem on pedestrian tracking

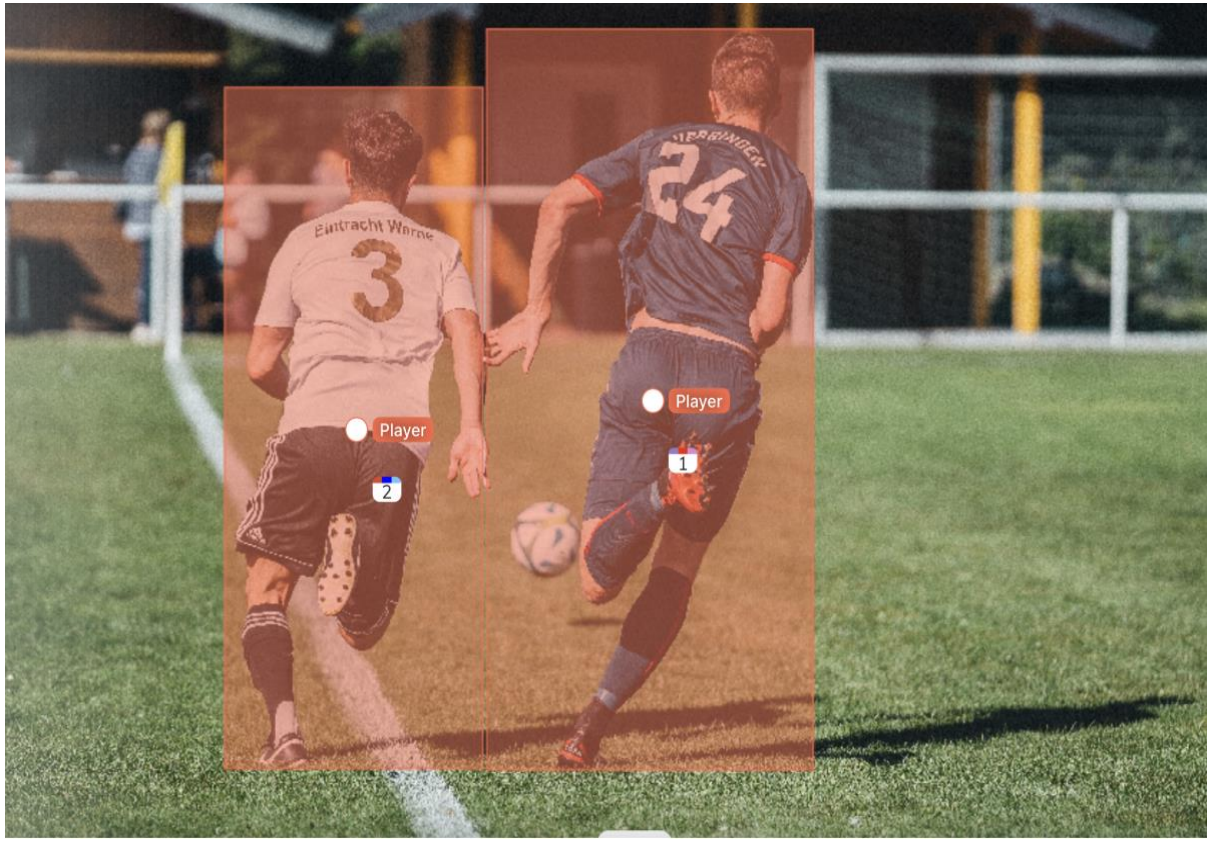
Traditional methods

- Optical Flow
  - Optical flow calculates motion by analyzing changes in image brightness across frames. It's effective but sensitive to noise and illumination changes, requiring high computational power.
- Meanshift
  - Mean Shift is an iterative algorithm that tracks objects based on color features. It's simple and fast but can struggle in complex visual environments.
- Kalman Filters
  - Kalman Filters predict an object's position and velocity by averaging past data with new measurements, effectively handling small occlusions and movement complexities. It assumes noise is Gaussian, limiting its use in non-linear situations.

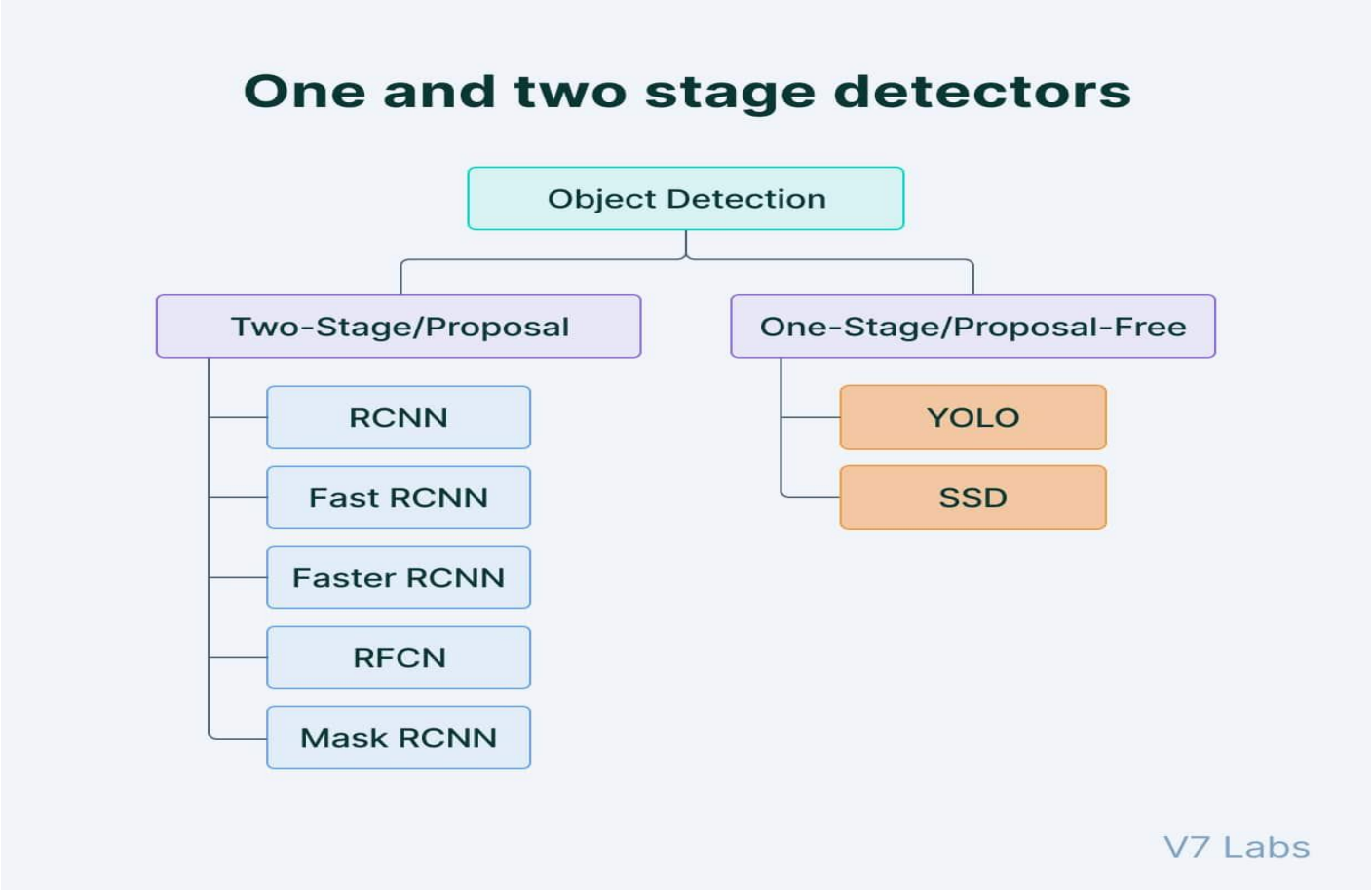
2、YOLO

What is object detection?

- Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. Object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors.



Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network.



Single-shot Object Detection:

- Single-shot object detection processes an entire image in one pass, making it computationally efficient but generally less accurate, particularly for detecting small objects; exemplified by YOLO, a fully convolutional neural network (CNN) model.

Two-shot Object Detection:

- Two-shot object detection involves two passes of the input image, with the first pass generating proposals and the second pass refining predictions; offering increased accuracy at the cost of higher computational complexity, making it suitable for applications prioritizing accuracy over real-time performance.

Object detection models performance evaluation metrics

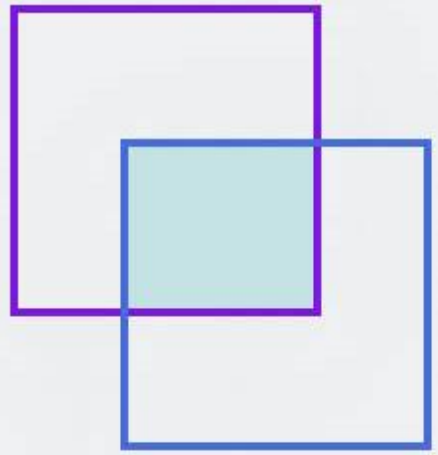
Intersection over Union (IoU):

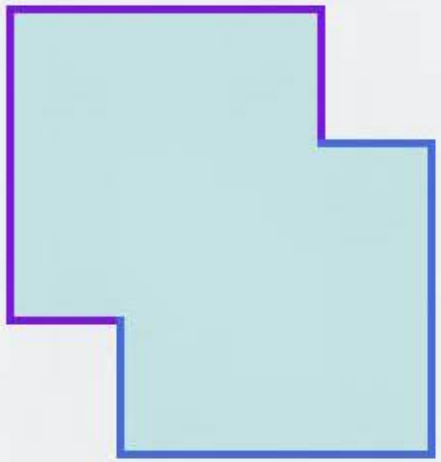
- IoU is a metric that measures the overlap between predicted and ground truth bounding boxes in object detection, calculated as the ratio of the intersection to the union of the two bounding boxes, providing an indication of localization accuracy.

Average Precision (AP):

- AP is calculated as the area under the precision-recall curve, quantifying the trade-off between precision and recall for a set of predictions in object detection; mean Average Precision (mAP) is the average AP value taken over all classes, indicating overall model performance.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$





YOLO: Algorithm for Object Detection

- YOLO (You Only Look Once) is a popular object detection model known for its speed and accuracy. It was first introduced by Joseph Redmon et al.
- YOLO employs a single neural network to predict bounding boxes and class probabilities simultaneously, distinguishing itself from traditional methods by achieving real-time performance and state-of-the-art results.

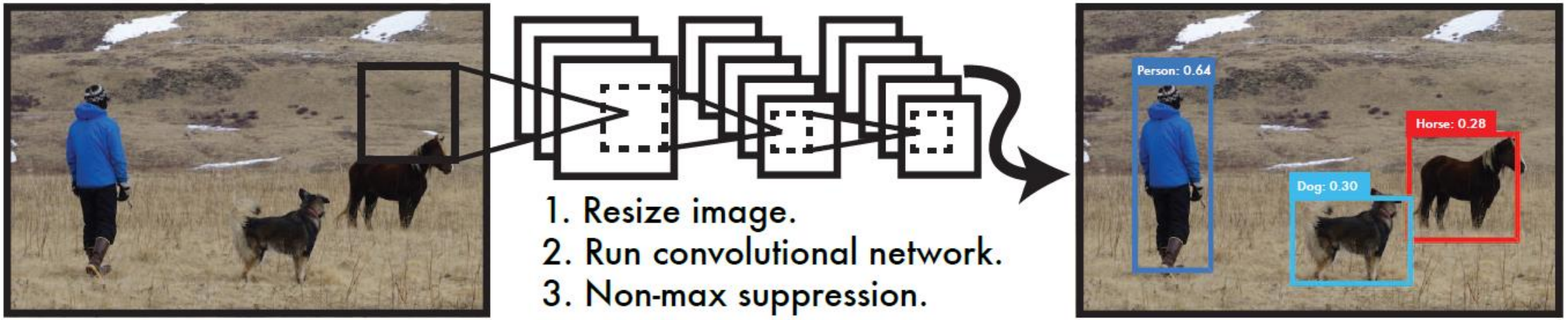


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

we define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

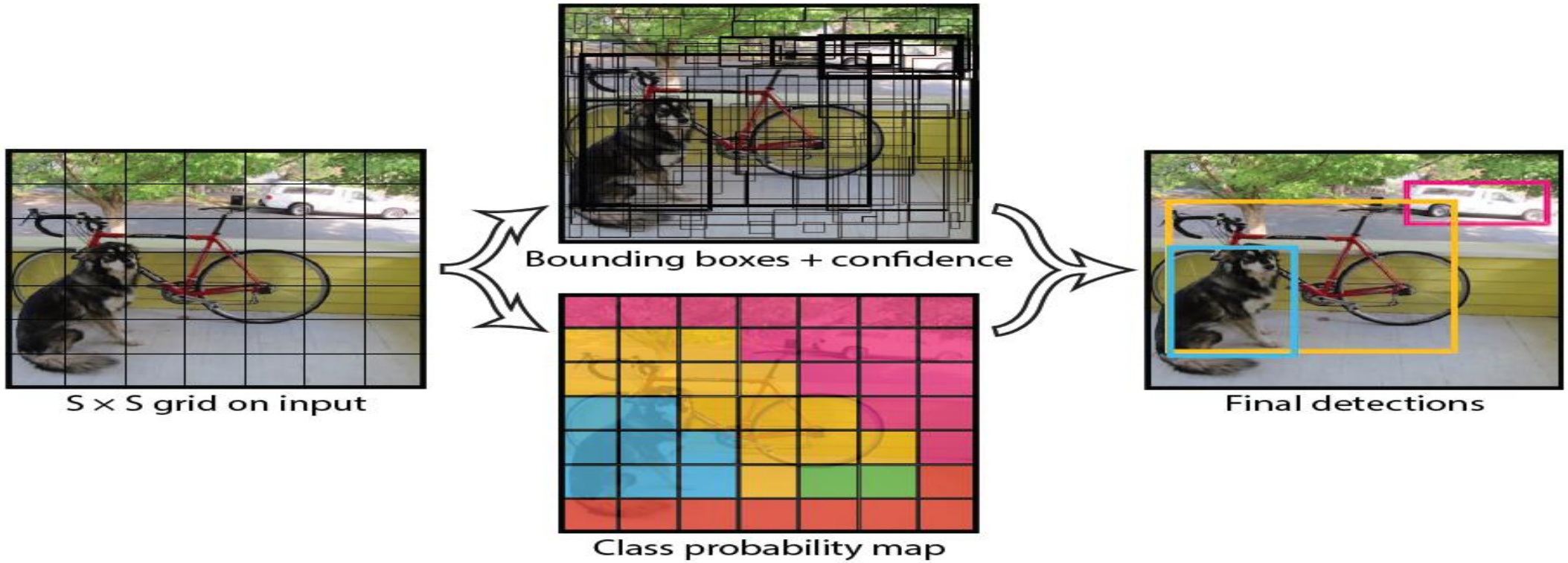


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

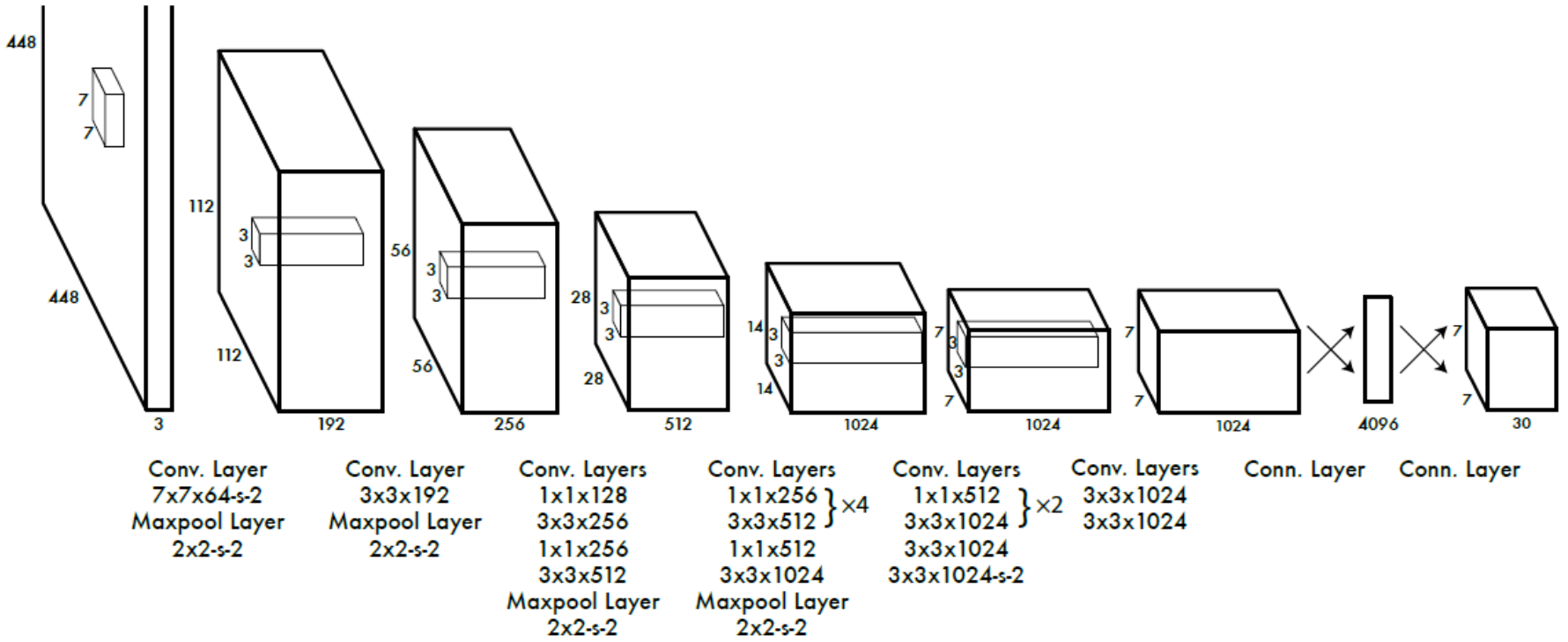


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

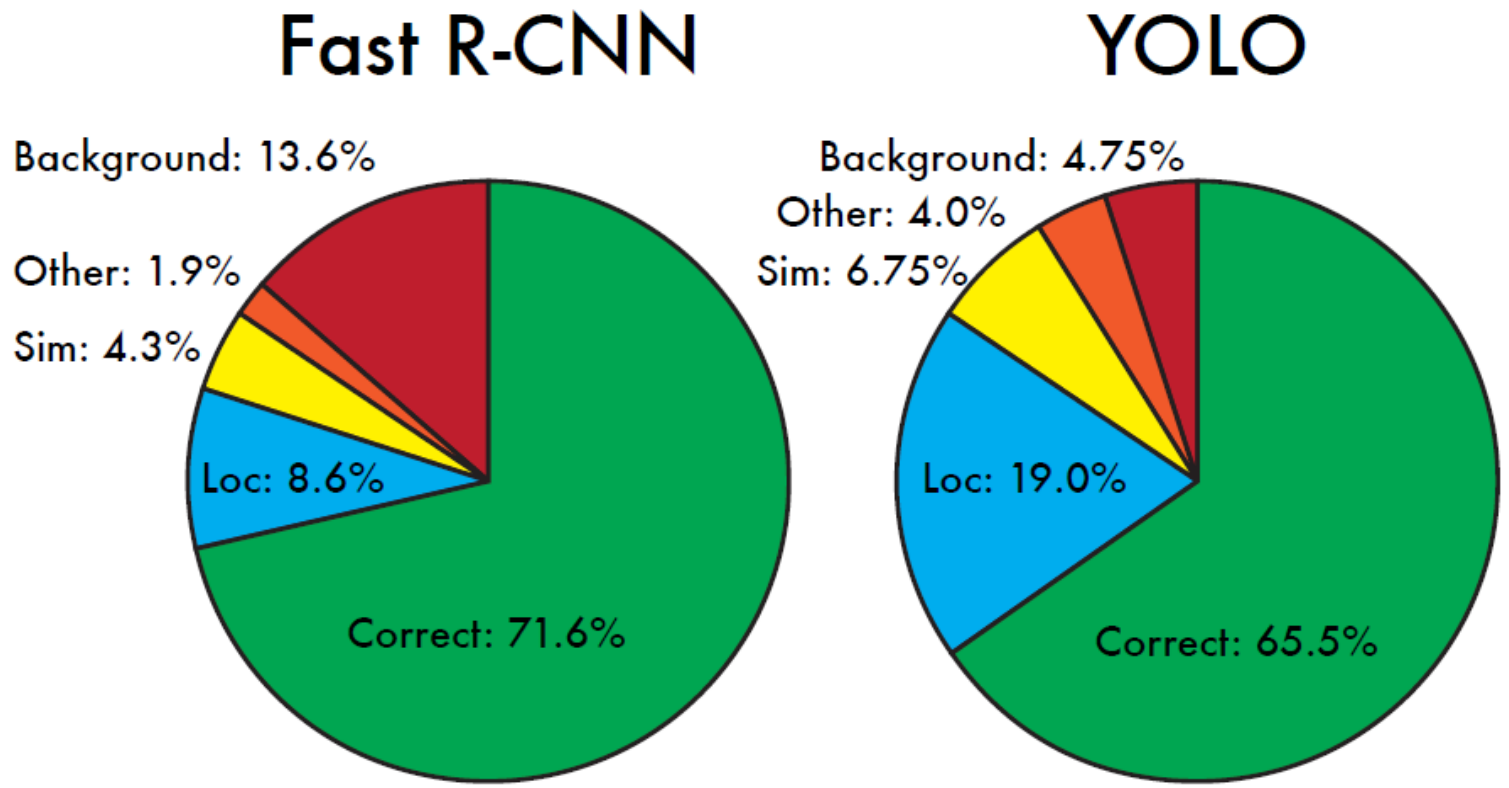
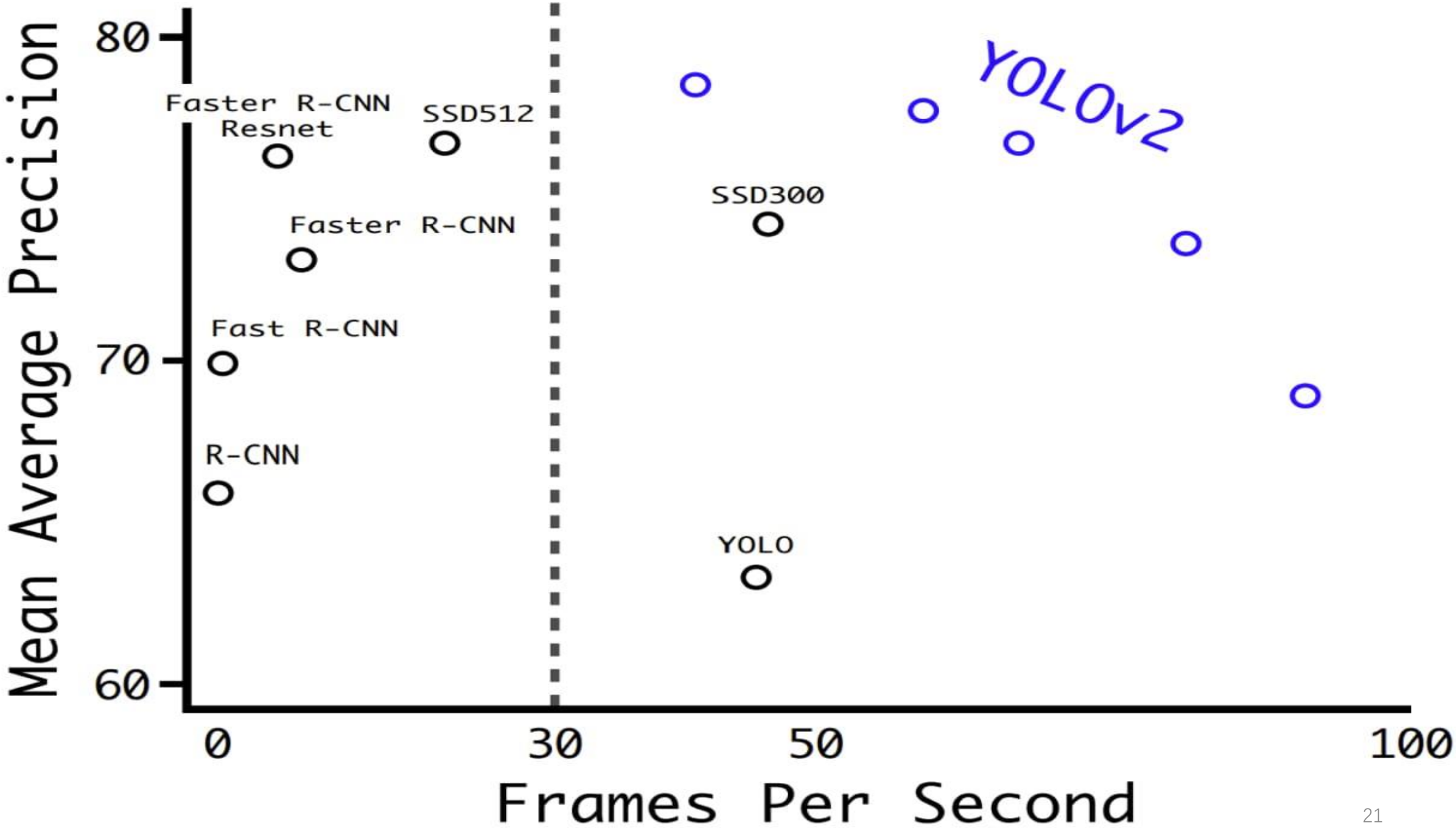


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).


Mean Average Precision
80
70
60
Faster R-CNN
Resnet
SSD512
Faster R-CNN
Fast R-CNN
R-CNN
SSD300
YOLO
YOLOv2
0
30
50
100
Frames Per Second

YOLO timeline

YOLOv10

- **Refined Model Architecture**: Utilizes sequential convolution layers with grouped and pointwise convolutions for better feature extraction and parameter optimization.
- **Dual-Pathway Strategy**: Incorporates both one2one and one2many pathways to enhance detection accuracy.
- **NMS-Free Training**: Eliminates the need for non-maximum suppression (NMS) during training, reducing inference latency.
- **Enhanced Bias Initialization**: Improves model convergence and stability.
- **Custom Post-Processing**: Implements a specialized step for fine-tuning detection outputs.

3、DeepSORT

The SORT algorithm: foundation of Deep SORT

- SORT, or Simple Online and Realtime Tracking, is the foundational algorithm for DeepSORT (Deep Learning for Multiple Object Tracking). SORT efficiently associates objects in consecutive frames using a detection-based approach and corrects associations with a Kalman filter. DeepSORT enhances data association with deep learning techniques, achieving state-of-the-art performance in multiple object tracking.

What is Deep SORT?

- Deep SORT is an advanced multi-object tracking algorithm, extending SORT with a deep association metric based on appearance features. It handles occlusions, incorporates ID assignment, and shows significant improvements in tracking accuracy and robustness, making it valuable in computer vision and AI applications.

How does Deep SORT work?

Deep SORT comprises four key components:

- Detection and Feature Extraction: Utilizes a CNN like YOLO for object detection, associating each detection with a high-dimensional appearance descriptor for matching.

- Kalman Filter in State Prediction: Similar to SORT, Deep SORT employs a Kalman filter to predict the state (position, velocity, acceleration) of each object in the current frame, considering motion dynamics.

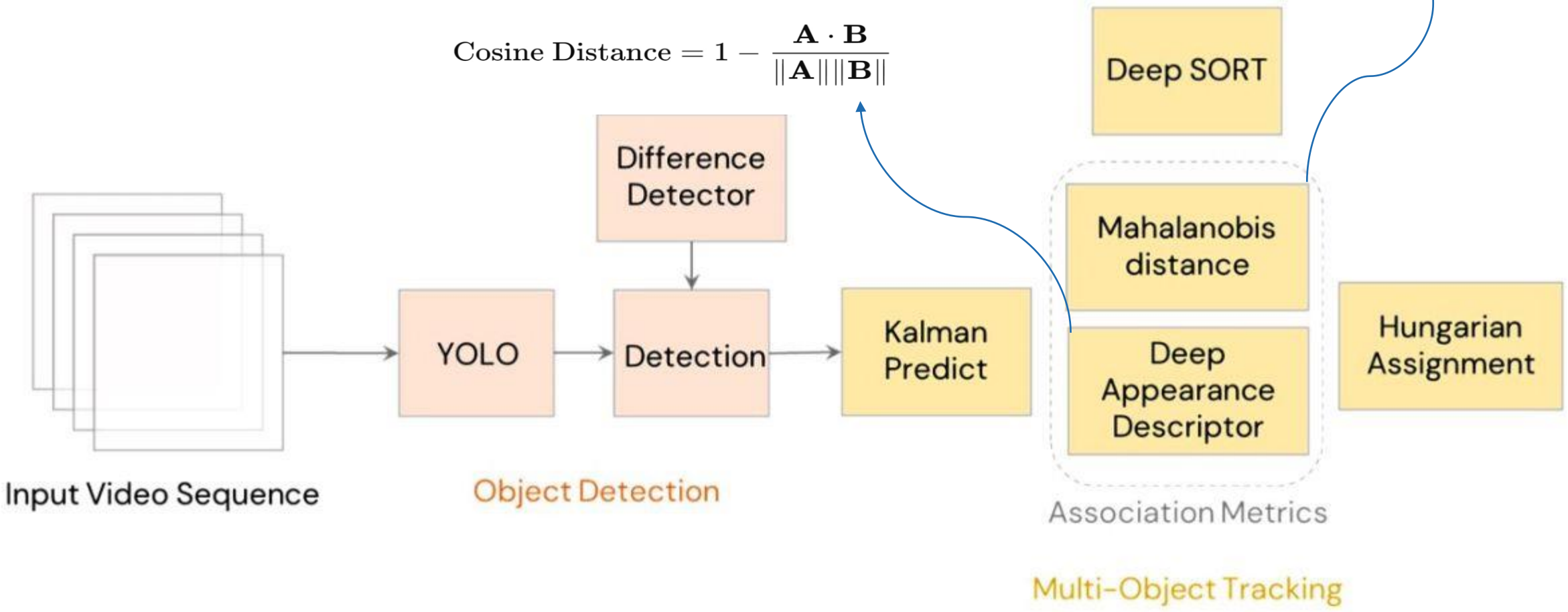
- Data Association with Deep Appearance Metrics: Differs from SORT by combining motion information and appearance features, using the Hungarian algorithm for matching based on a cost matrix considering Mahalanobis distance for motion and cosine distance for appearance.

- Track Management: Manages track lifecycles with heuristics like track confirmation (confirmed after detection in consecutive frames) and an age parameter to remove inactive tracks, reducing false positives.

$$\text{Cosine Distance} = 1 - \cos(\theta)$$

$$\text{Cosine Distance} = 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

$$D(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)}$$



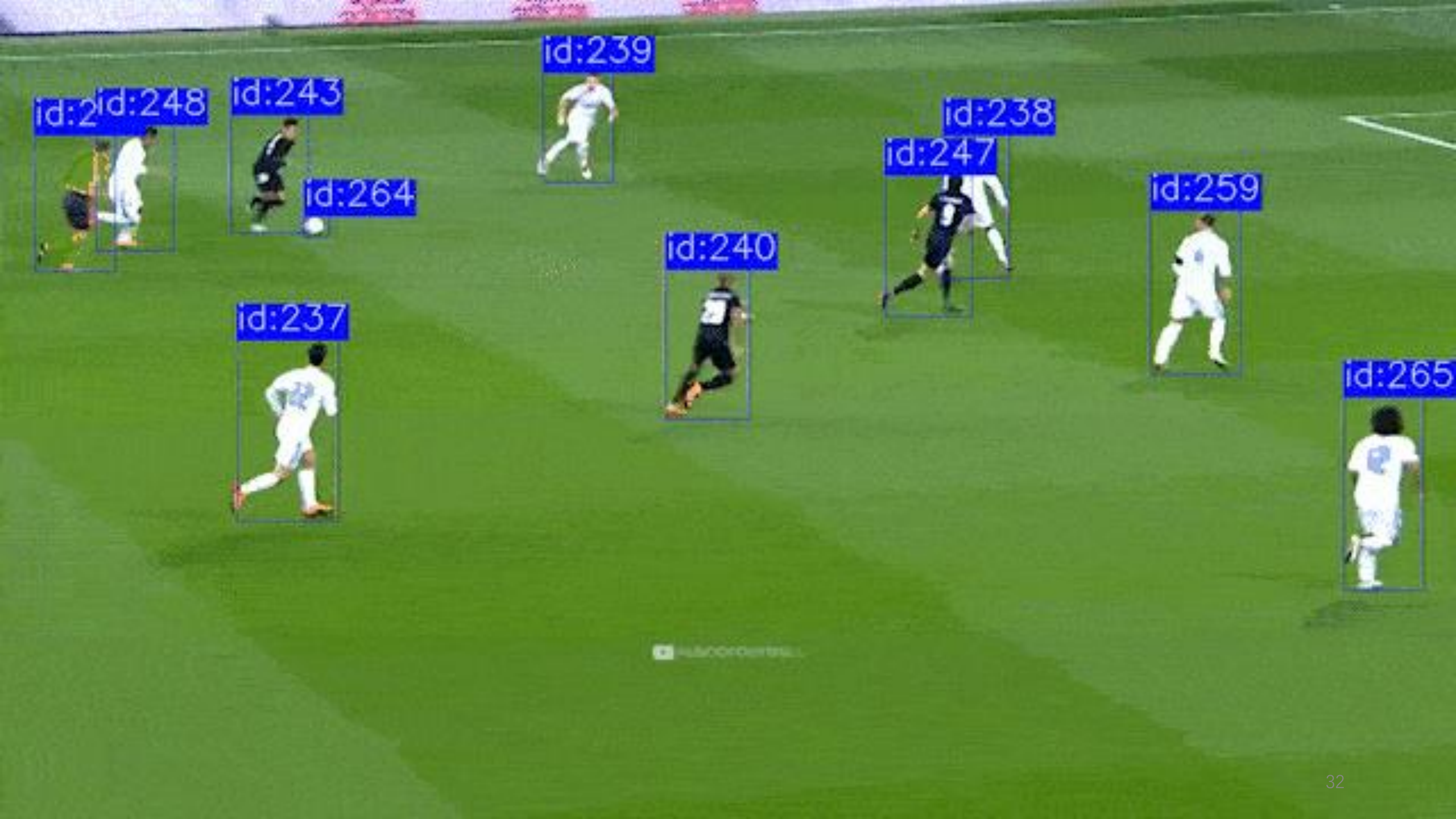
Architecture of Deep SORT

Advantages of Deep SORT include:

- Robustness to Occlusions: Utilizes appearance descriptors for re-identifying objects even after occlusions.
- Discrimination of Similar Objects: Appearance descriptors aid in distinguishing between objects that look similar.
- Real-time Performance: Maintains near real-time operation despite added complexity, thanks to efficient feature extraction and association mechanisms.
- Flexibility: Can be seamlessly combined with any state-of-the-art detector for enhanced adaptability.

Deep SORT Real-World Applications

- Surveillance: Enhancing security by tracking individuals or objects in crowded scenes.
- Sports analytics: Monitoring players to analyze game strategies and performance.
- Autonomous vehicles: Assisting in pedestrian and vehicle tracking to improve safety and navigation.
- Retail: Analyzing consumer behavior by tracking movements within a store.


id:239
id:2
id:248
id:243
id:264
id:238
id:247
id:259
id:240
id:237
id:265

Resources

- You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon, Santosh Divvala†, Ross Girshick, Ali Farhadi, University of Washington, Allen Institute for AI, Facebook AI Research
- https://www.ikomia.ai/blog/deep-sort-object-tracking-guide
- https://medium.com/analytics-vidhya/object-tracking-using-deepsort-in-tensorflow-2-ec013a2eeb4f
- https://www.v7labs.com/blog/yolo-object-detection
- Implementation : https://github.com/sujanshresstha/YOLO-NAS_DeepSORT/blob/main/object_tracking.py

Thanks.