香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)
信息枢纽
INFORMATION HUB
人工智能学域
ARTIFICIAL INTELLIGENCE

Object Detection & Tracking

Course AIAA 4220

Week4- Lecture 7&8

Changhao Chen

Assistant Professor

HKUST (GZ)

Recap: Image and Video Storage

What happens after camera capture?

The digital image is not stored in its raw form

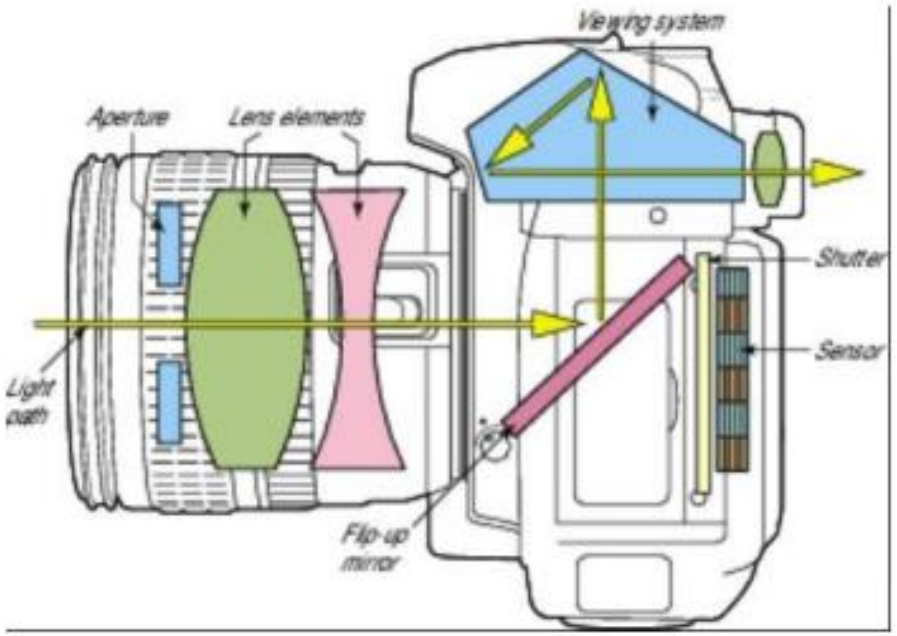


Image is processed in multiple stages before it is stored

- White balancing
- Demosaicing
  - Interpolation for colors
- Gamma correction
- And other processing steps

Recap: Visual Feature

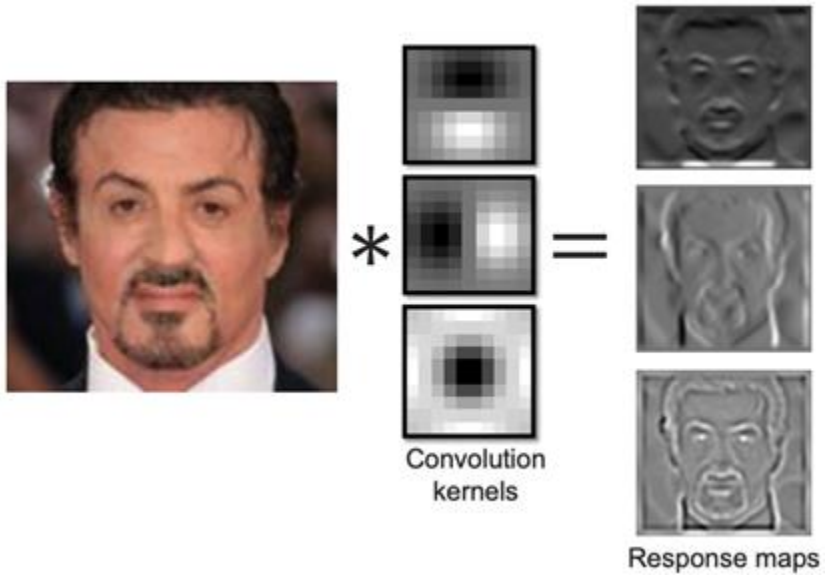
Local descriptors

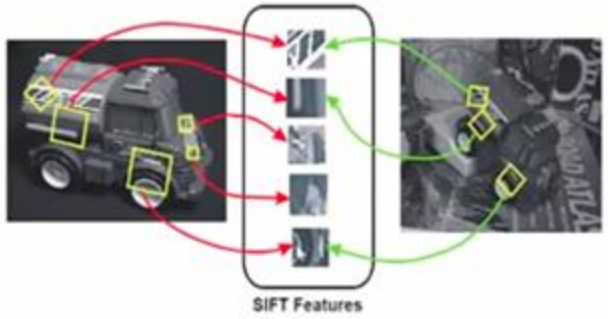
1) histogram of oriented gradients (HOG),

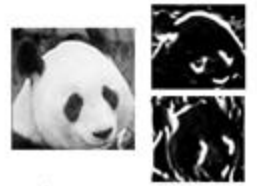
2) Scale-Invariant Feature Transform (SIFT), etc.

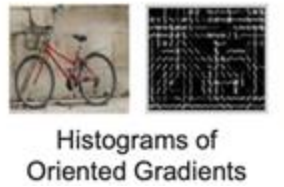
3) Haar-like features

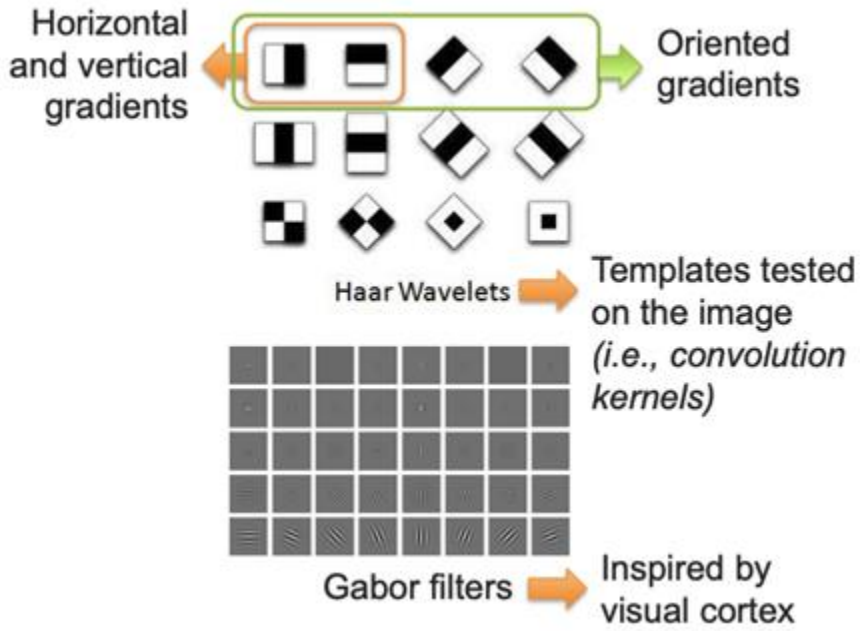
- Hand-crafted convolution filters
- Pixel “templates” sliding across image
- For face detection







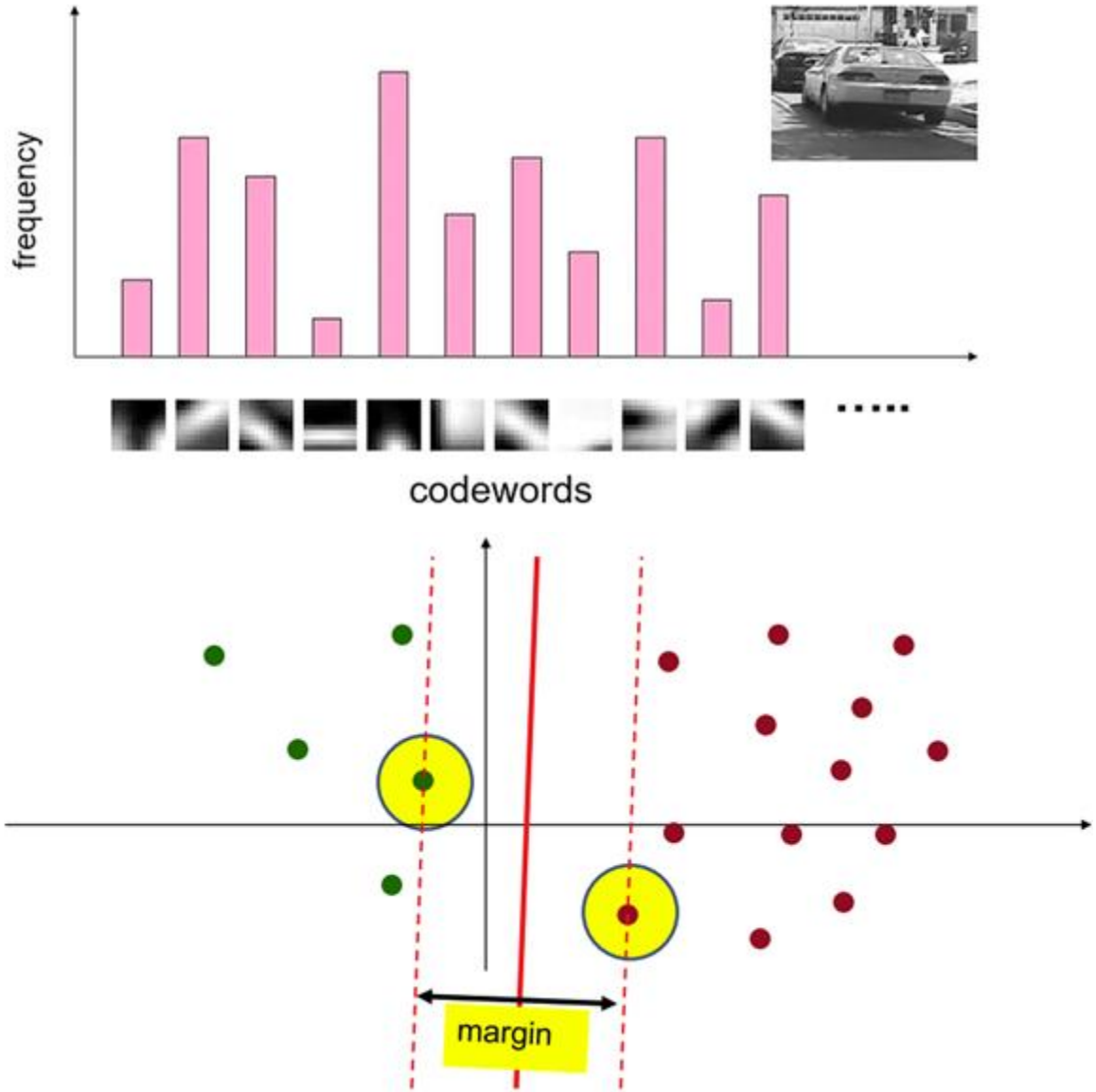




Recap: Visual Feature

Global representation

- Bag-of-visual-words
- Local descriptors
- Clustering: K-means
- Got fixed length vector representation for any size image/video
- Then apply ML methods like support vector machines (SVM)



Recap: Visual Recognition

Four Basic Neural Networks:

1. Multilayer Perceptron (MLP)

The simplest type of artificial neural network. Composed of multiple fully connected (dense) layers where every neuron in one layer is connected to every neuron in the next layer.

2. Convolutional Neural Network (CNN or ConvNet)

The dominant architecture for visual tasks. Designed to process data with a grid-like topology (e.g., pixels).

3. Recurrent Neural Network (RNN)

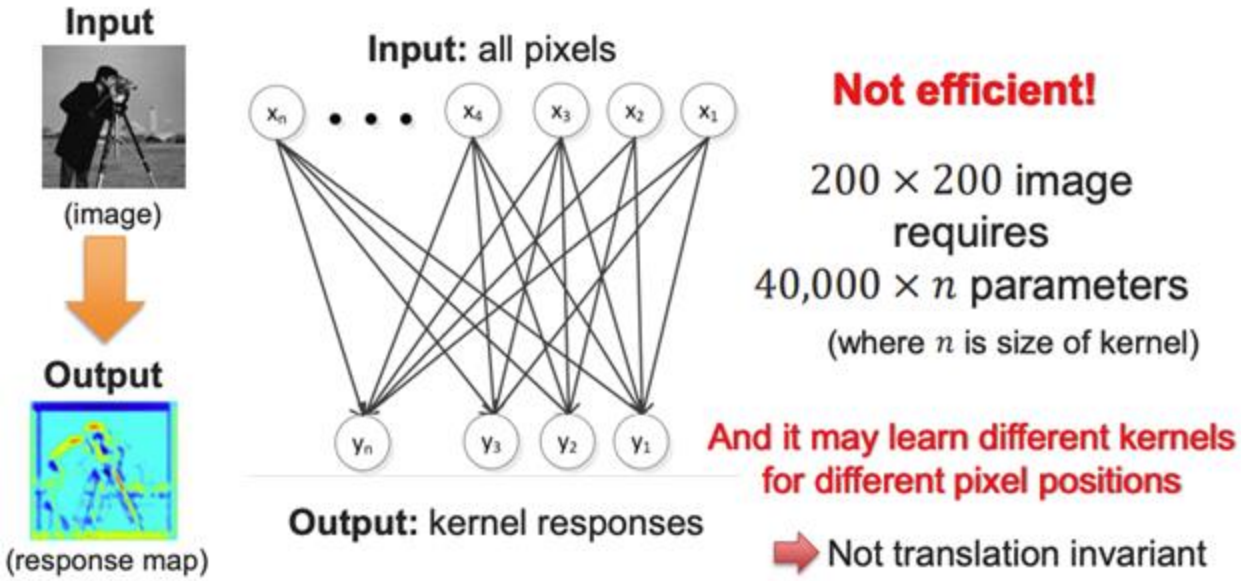
Networks with internal loops designed for sequential data. They maintain a "hidden state" that acts as a memory of previous inputs in the sequence.

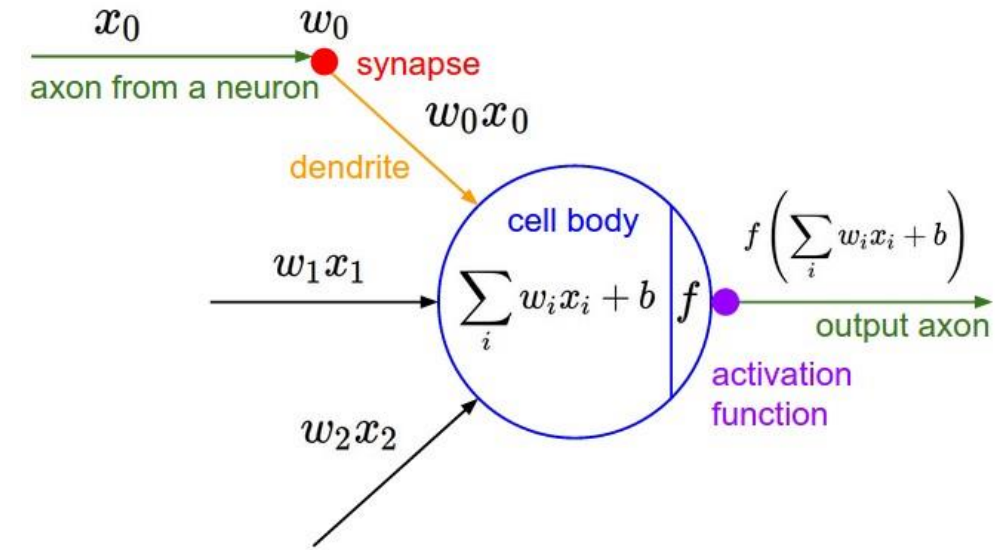
4. Self-Attention Mechanism

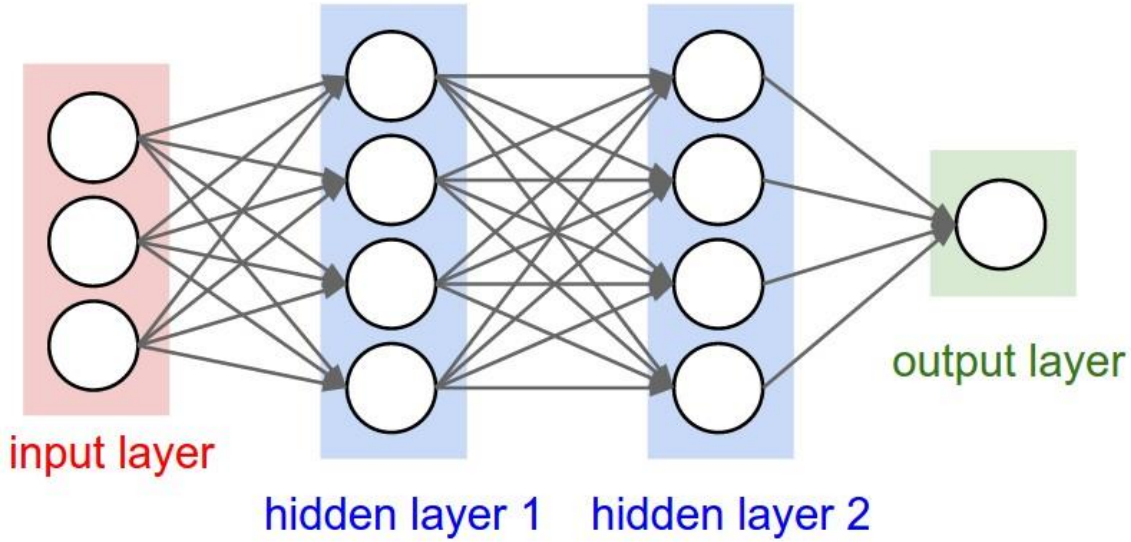
A mechanism that allows a model to weigh the importance of different parts of the input sequence when computing a representation for a specific part. The foundation of the Transformer architecture.

Recap: Multi-Layer Perceptron (MLP)

Fully-connected network





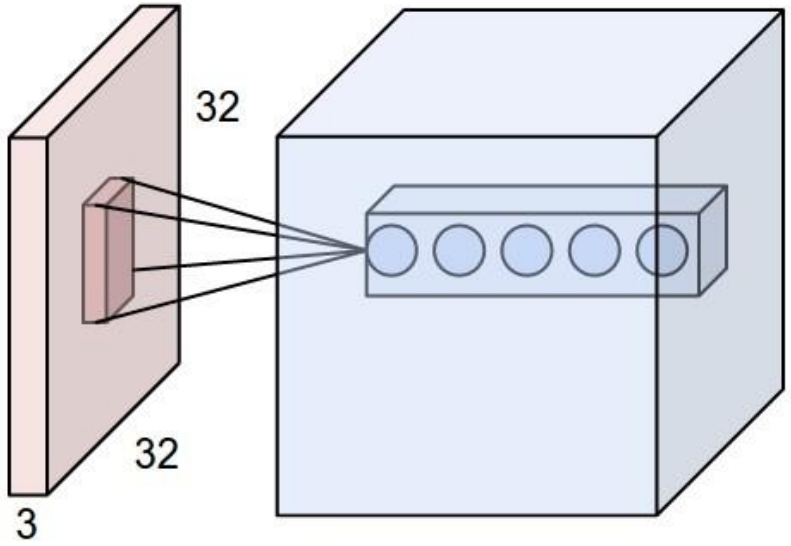


Recap: Convolutional Neural Network

A basic mathematical operation
(that given two functions returns a function)

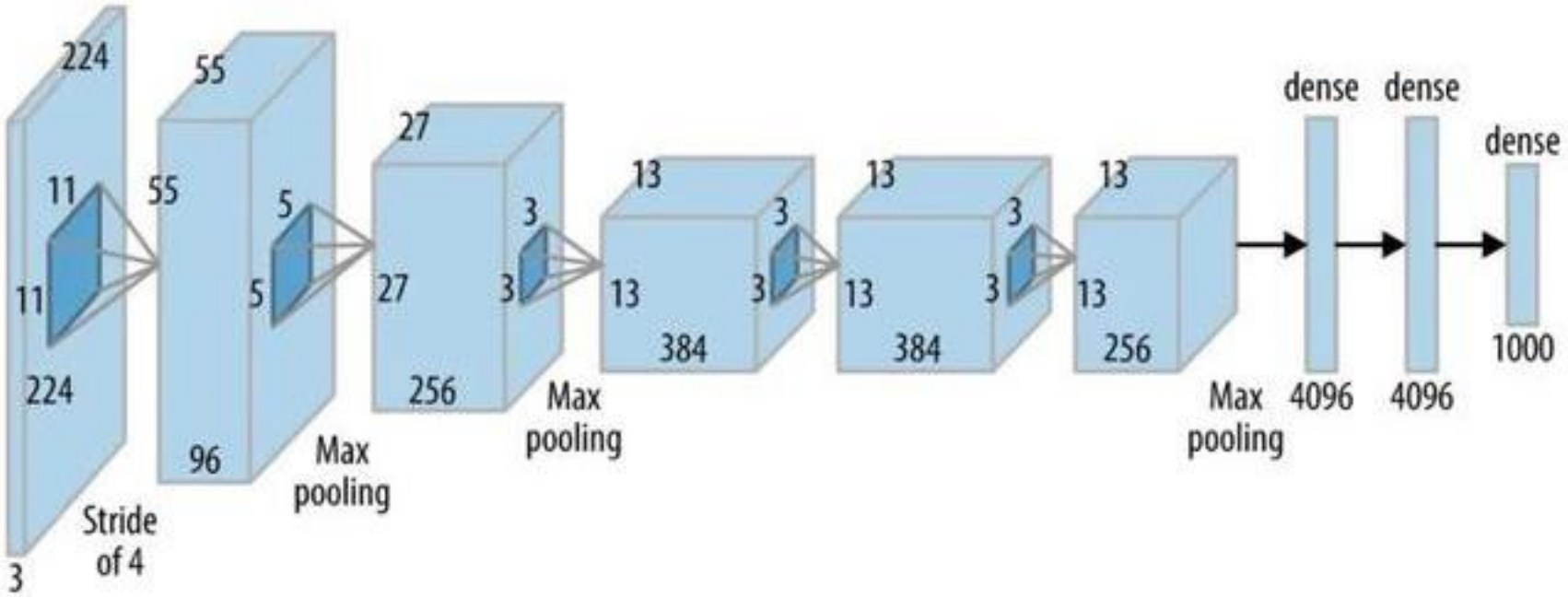
$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Summary. To summarize, the Conv Layer:



- Accepts a volume of size $\mathbf{W_1 \times H_1 \times D_1}$
- Requires four hyperparameters:
  - Number of filters $\mathbf{K}$,
  - their spatial extent $\mathbf{F}$,
  - the stride $\mathbf{S}$,
  - the amount of zero padding $\mathbf{P}$.
- Produces a volume of size $\mathbf{W_2 \times H_2 \times D_2}$ where:
  - $\mathbf{W_2 = (W_1 - F + 2P)/S + 1}$
  - $\mathbf{H_2 = (H_1 - F + 2P)/S + 1}$ (i.e. width and height are computed equally by symmetry)
  - $\mathbf{D_2 = K}$
- With parameter sharing, it introduces $\mathbf{F \cdot F \cdot D_1}$ weights per filter, for a total of $\mathbf{(F \cdot F \cdot D_1) \cdot K}$ weights and $\mathbf{K}$ biases.
- In the output volume, the $\mathbf{d}$-th depth slice (of size $\mathbf{W_2 \times H_2}$) is the result of performing a valid convolution of the $\mathbf{d}$-th filter over the input volume with a stride of $\mathbf{S}$, and then offset by $\mathbf{d}$-th bias.

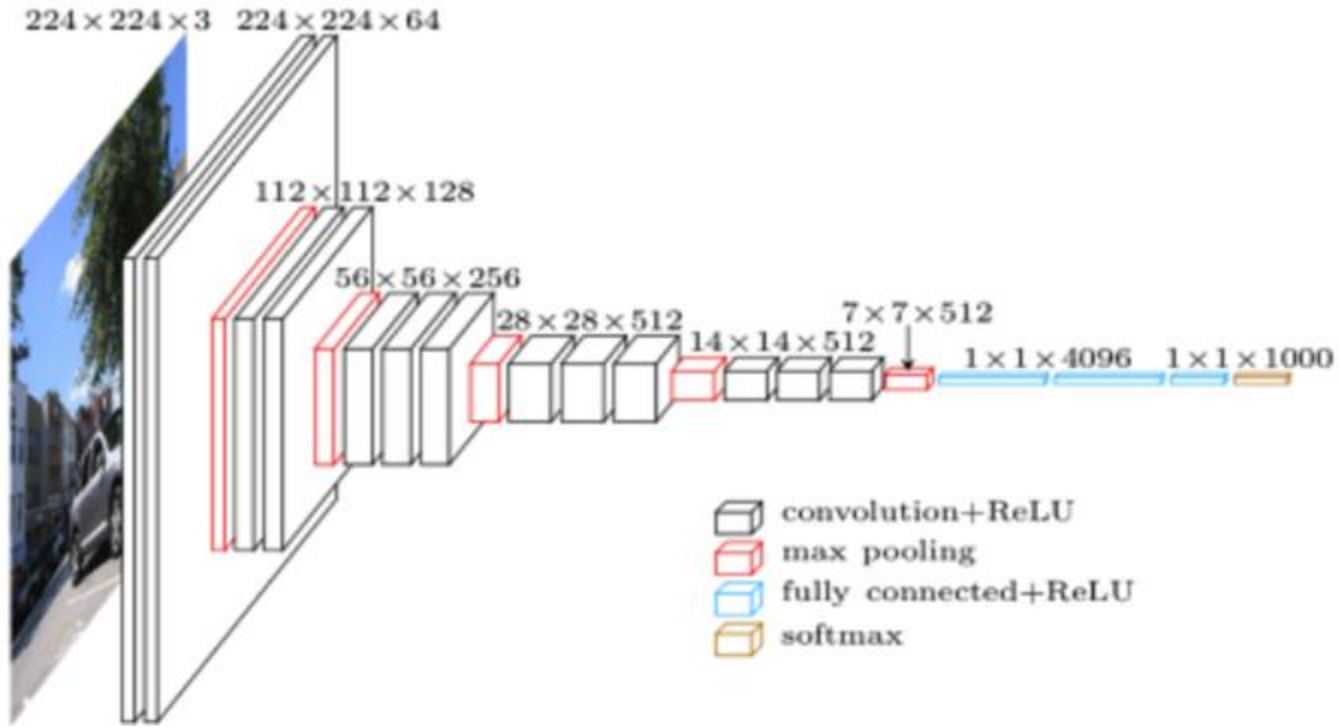
Recap: Typical CNN Architectures



AlexNet

- Deep Architecture: It was one of the first deep CNNs (8 learned layers: 5 convolutional, 3 fully-connected) to be successfully trained on a large, complex dataset (ImageNet with 1.2M images).
- ReLU Nonlinearity: Used the Rectified Linear Unit (ReLU) activation function instead of Tanh/Sigmoid. This drastically accelerated training (~6x faster) by mitigating the vanishing gradient problem.
- GPU Acceleration: Pioneered the use of GPUs (NVIDIA GTX 580) for training large neural networks, making deep learning feasible.
- Overlapping Max Pooling: Used pooling with a stride smaller than the kernel size, improving feature richness and helping to reduce overfitting.

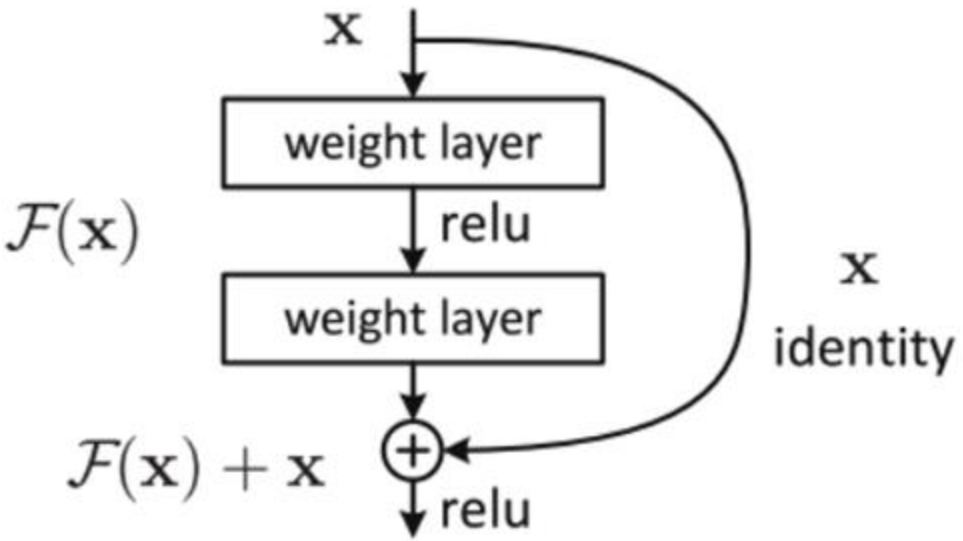
Recap: Typical CNN Architectures

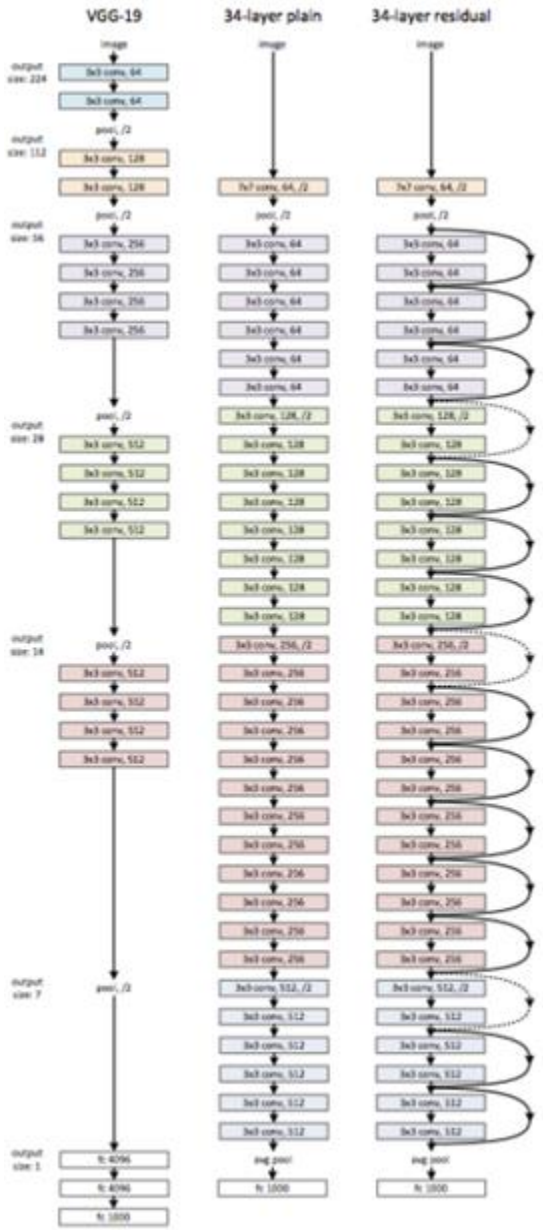


- VGGNet model
  - Used for 1000-way image classification task
  - 138 million parameters / 16-19 layers
  - *Still relevant today due to its simple connection and fast inference

Recap: Typical CNN Architectures

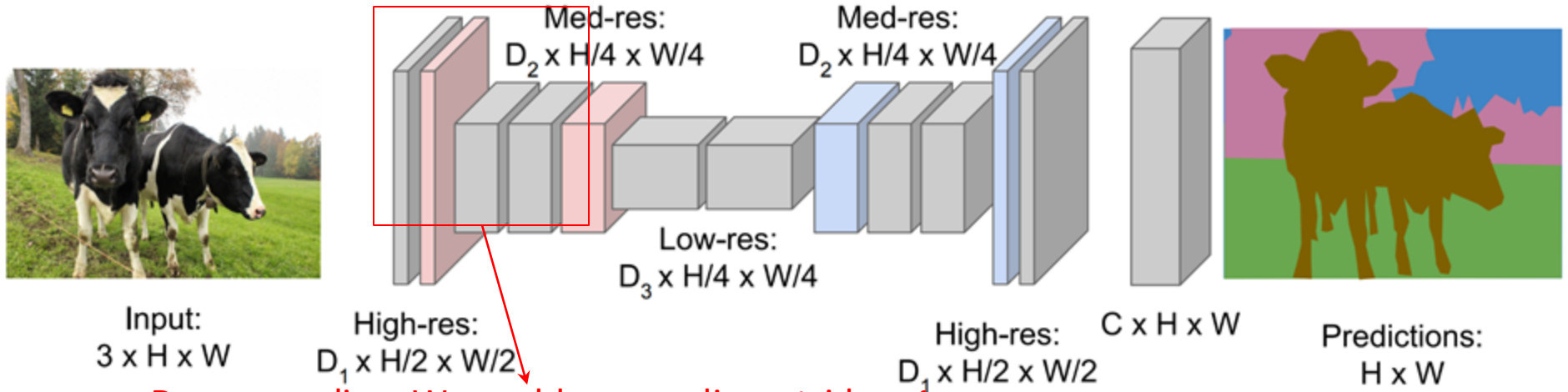
- Adding residual connections
- ResNet (He et al., 2015)
  - Up to 152 layers
  - But only 60 million parameters (138M for VGG, 3% better)





Recap: Semantic Segmentation

- Idea: convolutional network
  - We see that convolution can get same-sized outputs
  - Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!



Downsampling: We could use pooling, stride > 1 convolution

Recap: Multimodal Learning: Vision-Language Model

- CLIP (Contrastive Language–Image Pre-training) by OpenAI in 2021
  - Train on 400 million image+text crawled from the internet
  - Proved that large-scale (noisy) vision-language pre–training can lead to great performance (with proper prompt engineering)
    - Any zero-shot downstream classification tasks
    - AI generated content (text-to-image generation)
      - DALL-E, etc.
- Why is this useful for Embodied AI?
  - We need to generalize our perception model to just recognizing discrete object classes
    - CIFAR-10/100, ImageNet-1K, ImageNet-21K
    - Finite set of vocabularies

* https://openai.com/blog/clip/

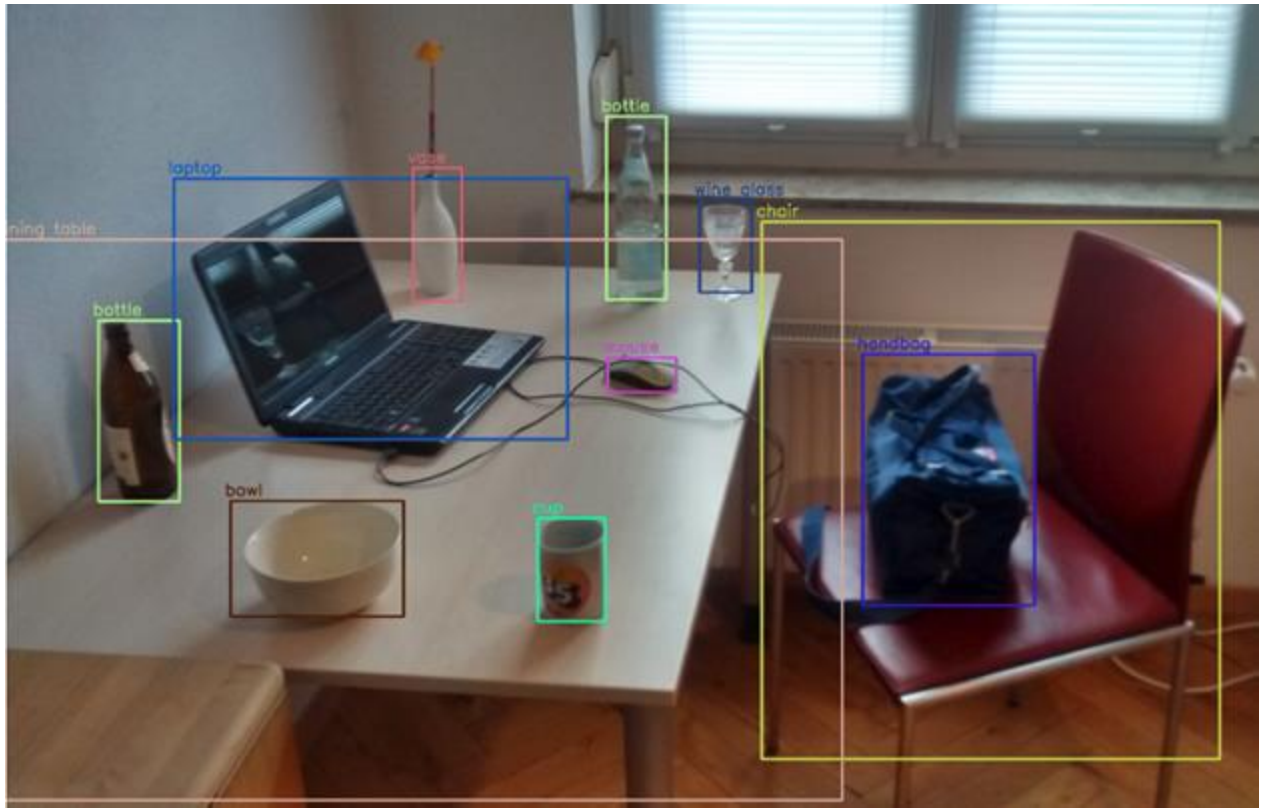
Lecture Roadmap

•High-level overview: Image Classification -> Object Localization -> Object Detection -> Object Tracking

1. Object Detection Fundamentals
2. The Evolution of Deep Learning-Based Detectors (e.g. R-CNN Family, YOLO)
3. Advanced Architectures for Object Detection (Feature Pyramids, Transformers)
4. Object Tracking Fundamentals
5. Multi-Object Tracking (MOT) Strategies

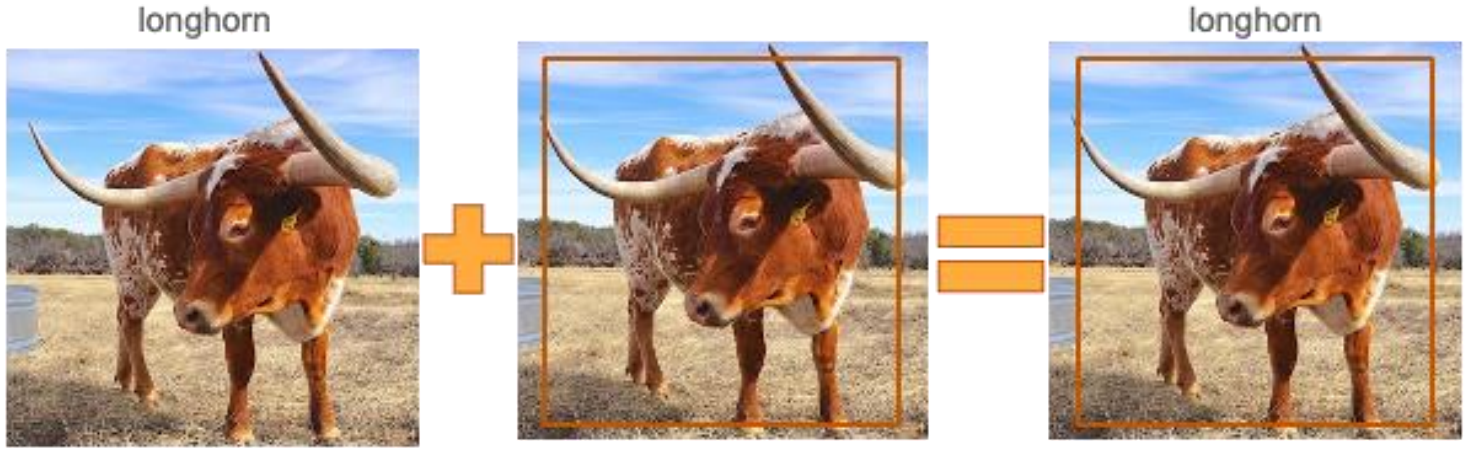
Object Detection

- Why object detection is important?
  - Core component in various computer vision tasks like facial recognition, image search, and augmented reality
  - Object detection is widely used in security systems for monitoring and tracking people or objects of interest
  - Object detection is also crucial for robots!



Object Detection

Task: Classification + Localization
Formal Definition: "What is it and where is it?"

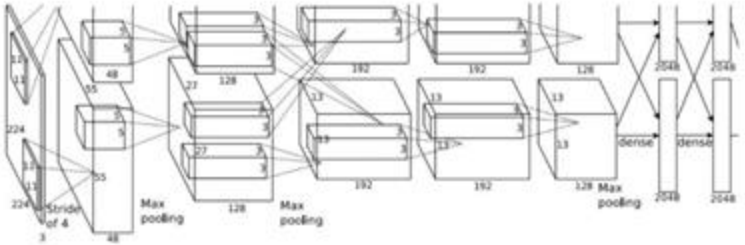


	classification	localization	bbox detection
output	class	(xyxy) or (xywh)	(xyxy, cls) or (xywh, cls)

- **Localization as Regression:** Predicting the bounding box coordinates (e.g., x, y) is a regression task.
- **Pixel Coordinate System:** The top-left corner of an image is defined as the origin (0, 0).
- **Bounding Box Representations:**
  - **Corners:** (x_1, y_1) of the top-left and (x_2, y_2) of the bottom-right.
  - **Center & Dimensions:** (x, y) of the center point, plus width (w) and height (h).

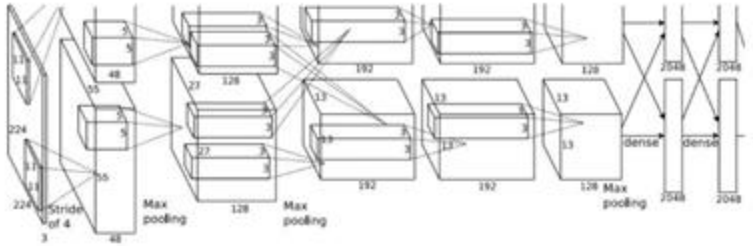
Object Detection

- Sliding window approach
  - Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

We can use a predefined bounding box to slide through the image. Each time we crop the image and forward it to the CNN



Dog? YES
Cat? NO
Background? NO

If the box happens to crop the object, the CNN should classify it correctly

Object Detection

Sliding window approach - Problems

1.Computational Nightmare:

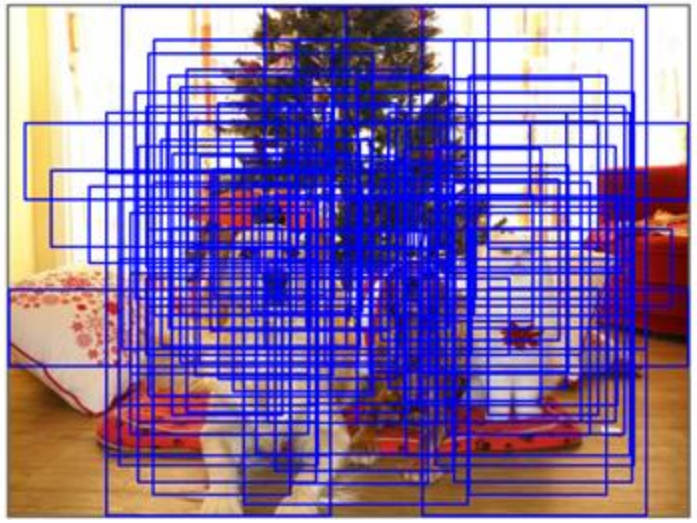
- **Inefficiency:** You have to classify a massive number of patches.
- **Redundancy:** Most windows contain only meaningless background, wasting almost all of the computation.

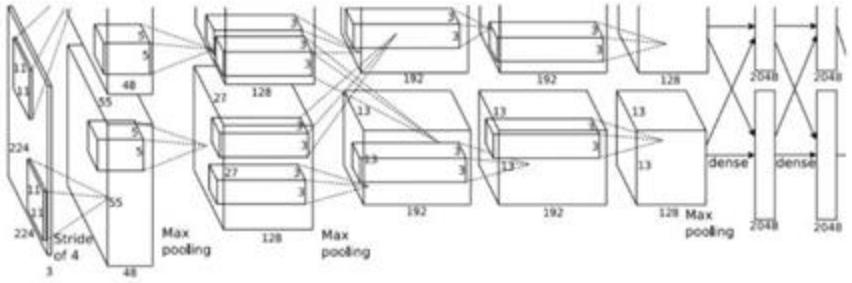
2.Fixed Window Shape:

- The rigid window shape struggles with objects that have extreme aspect ratios

3.Inaccurate Localization:

- The bounding box is constrained by the pre-defined window size and may not fit the object perfectly.





Dog? NO
Cat? YES
Background? NO

Object Detection

Solution: Region Proposals with Selective Search

Goal: Generate regions likely to contain objects.
Groups pixels into "blobby" regions using low-level features:

- Color
- Texture

Process:

- **Initial Over-segmentation:** Create small, homogeneous regions from pixels with similar color.
- **Hierarchical Merging:** Progressively merge smaller regions into larger, meaningful segments.

Why Over-segment? Ensures object parts are captured, allowing the algorithm to build objects from smaller components.

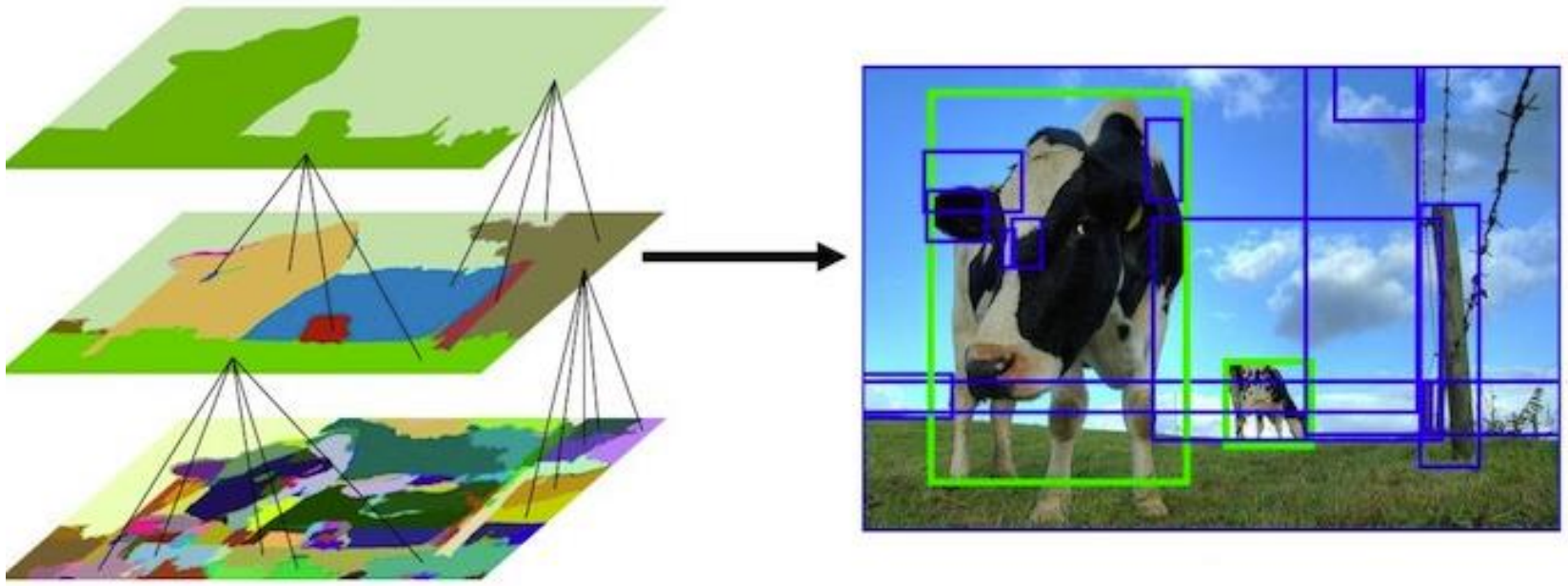
*Measuring the objectness of image windows. TPAMI 2012

Object Detection

Solution: Generating Region Proposals with Selective Search

Advantages: Computationally efficient; generates ~2000 region proposals in a few seconds on a CPU.

Output: A set of candidate regions, each defined by a bounding box drawn around the grouped pixels.



*Segmentation as Selective Search for Object Recognition. ICCV 2011

Object Detection-Two-stage method

- **R-CNN**

Combine efficient region proposals (like Selective Search) with powerful CNN classifiers.

The Strategy:

1. Propose: Use Selective Search to generate ~2,000 candidate regions.
2. Classify: Warp each region and process it with a CNN (e.g., AlexNet) to extract features.
3. Predict: Use the features to classify the object and refine the bounding box.

Shifts object detection from a slow, impractical process to a viable, high-accuracy method.

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

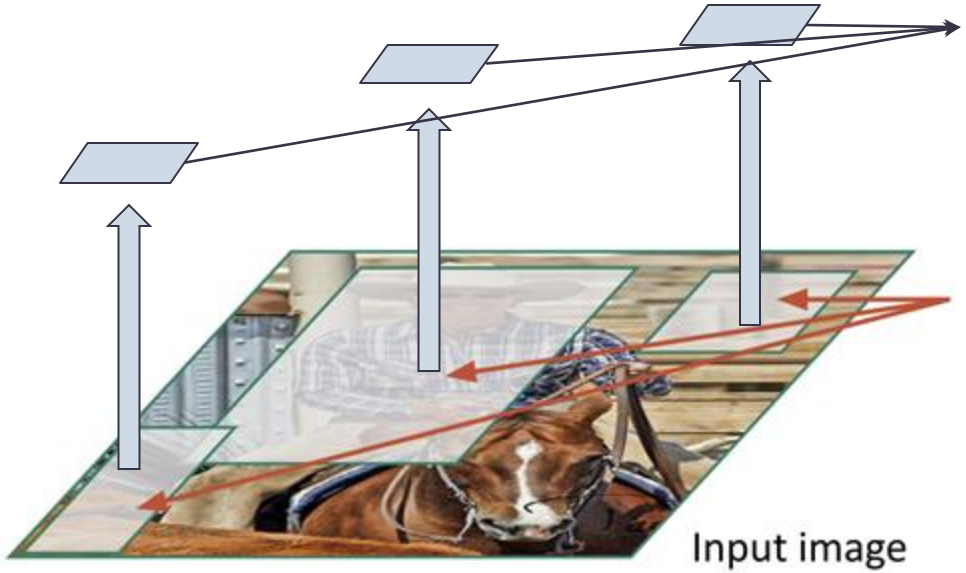
- R-CNN

1) Regions of Interest (RoI) from a proposal method (~2k)

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- R-CNN



2) Warped image regions (224x224 pixels)

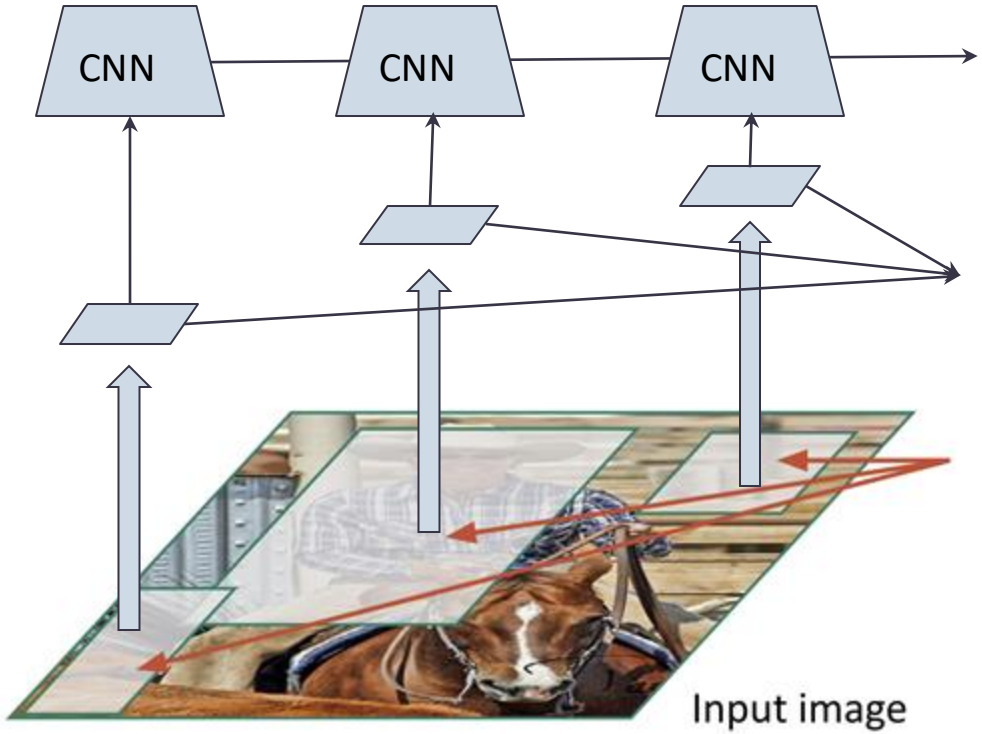
1) Regions of Interest (RoI) from a proposal method (~2k)

Why same size? We need same-sized features for the classification layer

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- R-CNN



3) Each image is fed into a shared CNN and get a vector representation

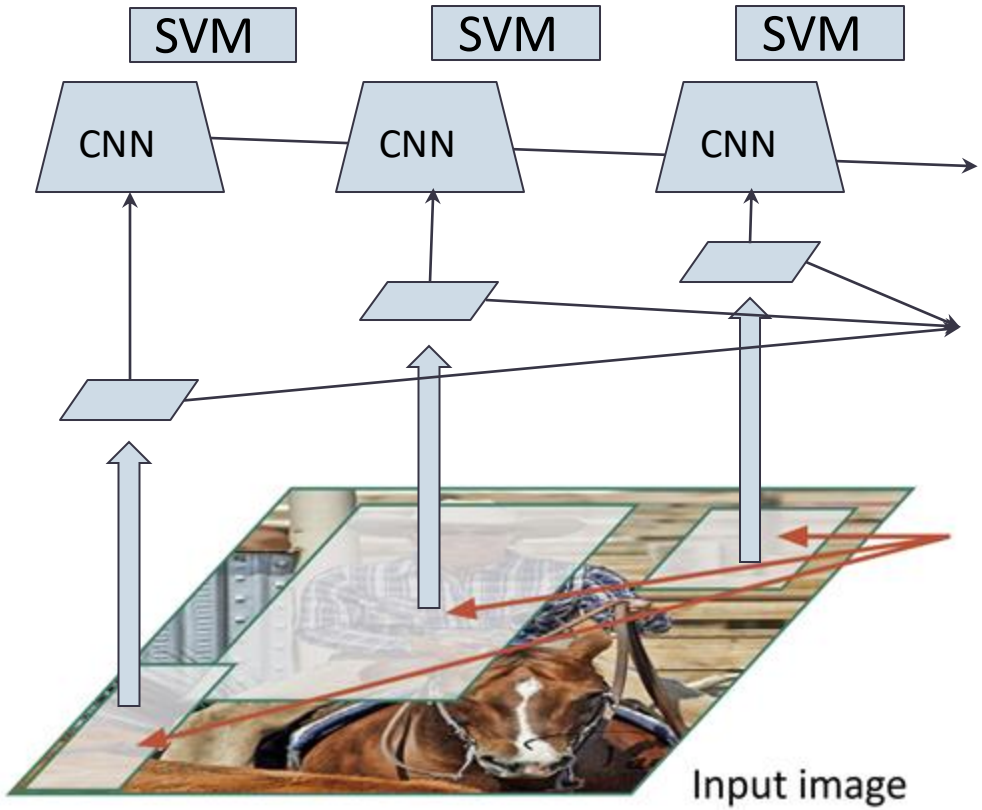
2) Warped image regions (224x224 pixels)

1) Regions of Interest (RoI) from a proposal method (~2k)

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- R-CNN



4) Use SVM to classify each region

3) Each image is fed into a shared CNN and get a vector representation

2) Warped image regions (224x224 pixels)

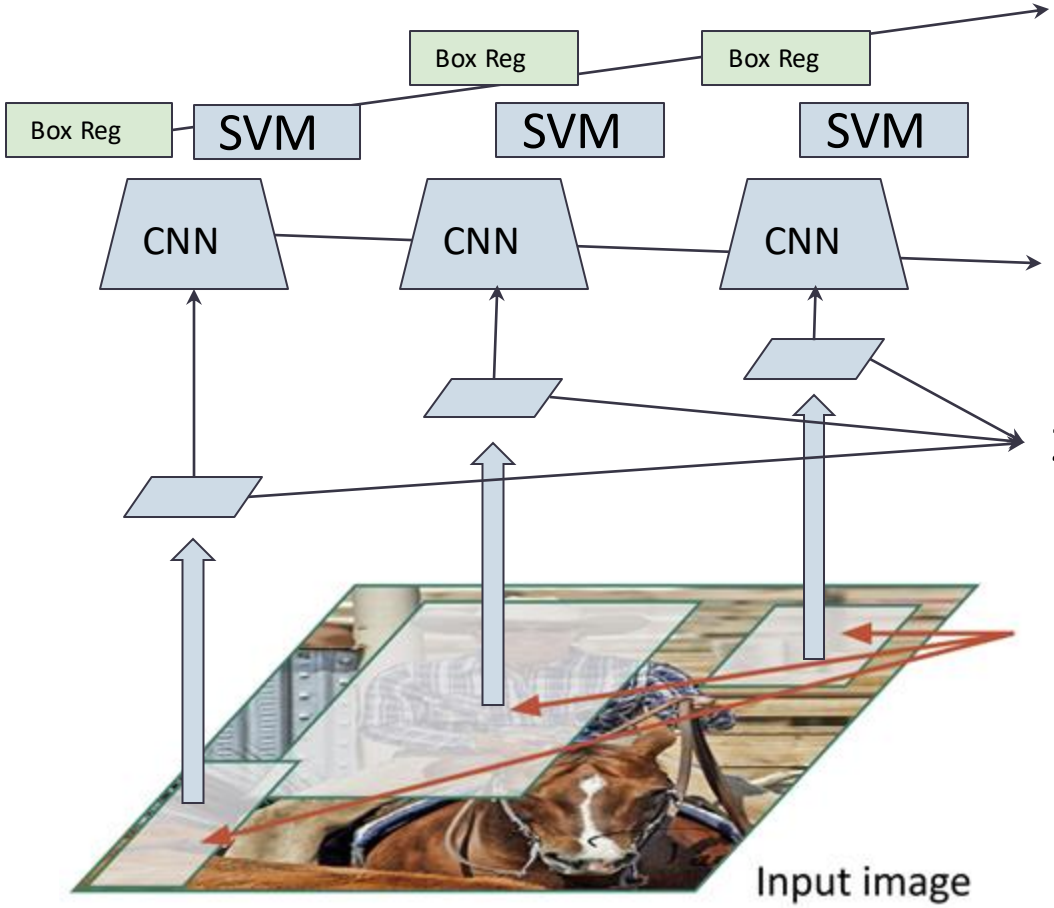
1) Regions of Interest (RoI) from a proposal method (~2k)

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- R-CNN

This is key. We still need a box regression task to get the final box (the region proposals are not accurate)



5) Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)

4) Use SVM to classify each region

3) Each image is fed into a shared CNN and get a vector representation

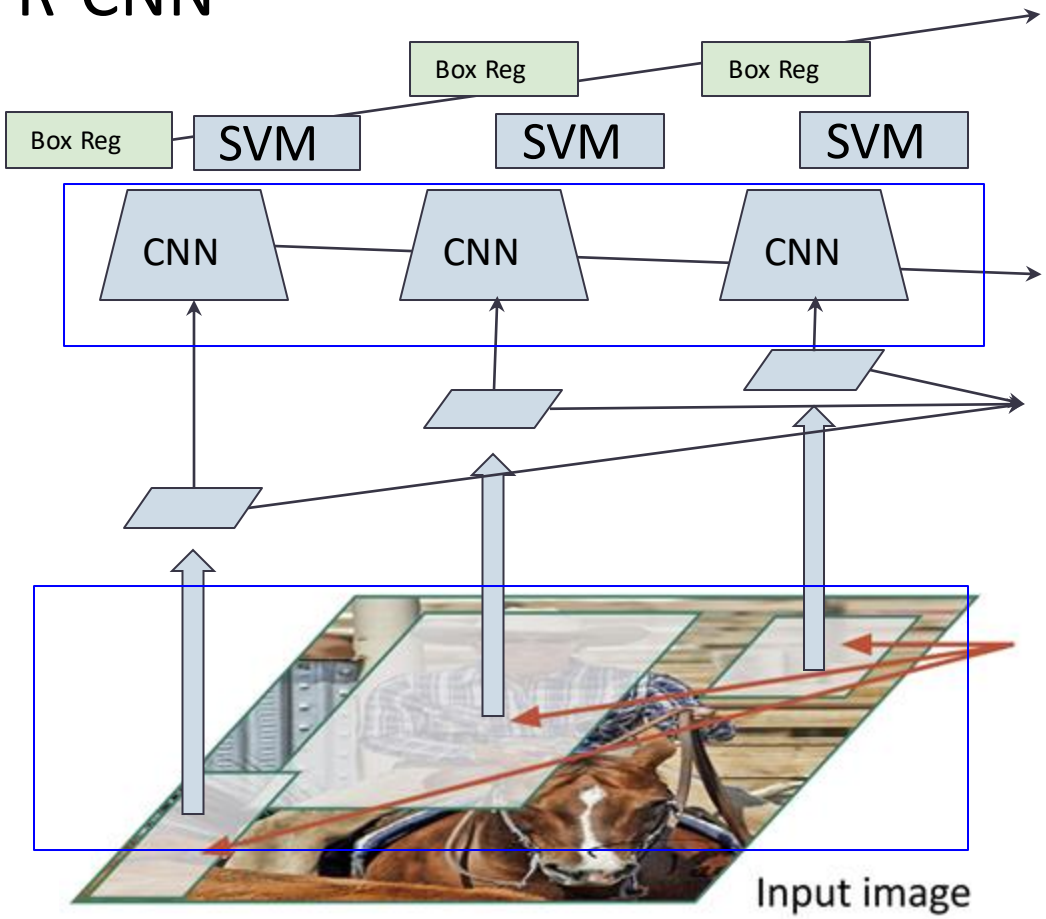
2) Warped image regions (224x224 pixels)

1) Regions of Interest (RoI) from a proposal method (~2k)

*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- R-CNN



5) Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)

4) Use SVM to classify each region

3) Each image is fed into a shared CNN and get a vector representation

2) Warped image regions (224x224 pixels)

1) Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow!
Where is the bottleneck?
Where do we waste compute?
How can we solve this?

Idea: Regions are heavily overlapped and the CNN computation is redundant.
So we compute the full-image CNN feature and crop the feature

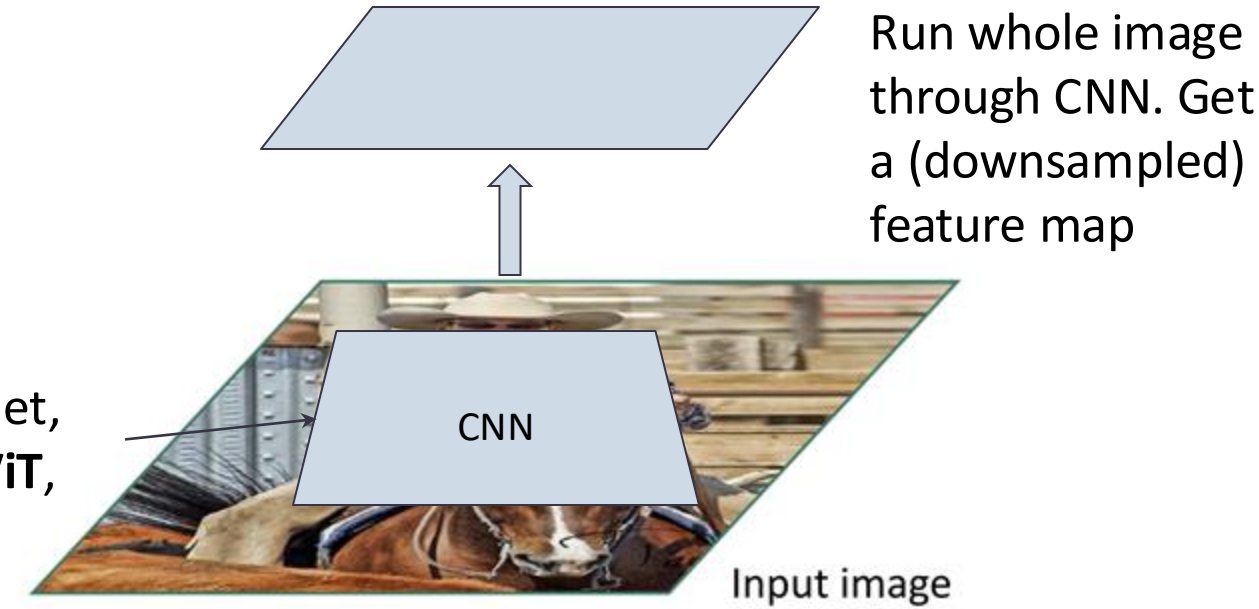
*Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014

Object Detection-Two-stage method

- Fast R-CNN

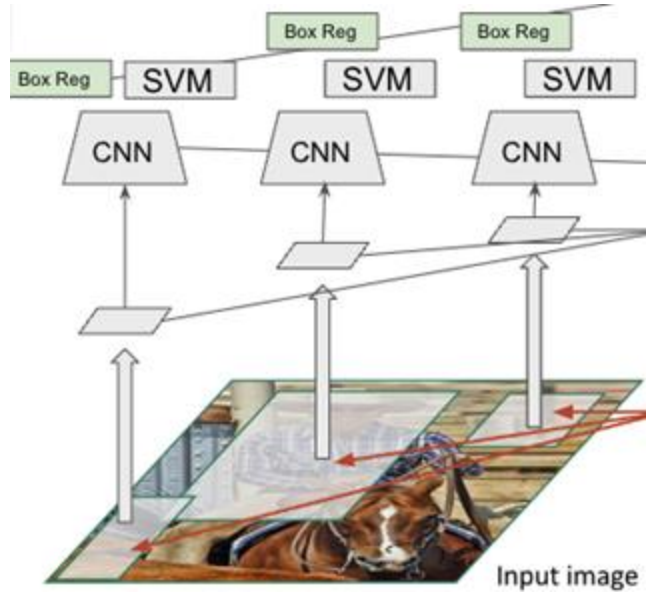
Key processes:

1. Run the *entire image* through a CNN once to get a feature map.
2. Project region proposals onto the feature map (RoI Projection).
3. Use RoI Pooling to get fixed-size features from each proposal.



"Backbone" network: AlexNet, VGG, ResNet, **ViT**, etc

Slow R-CNN

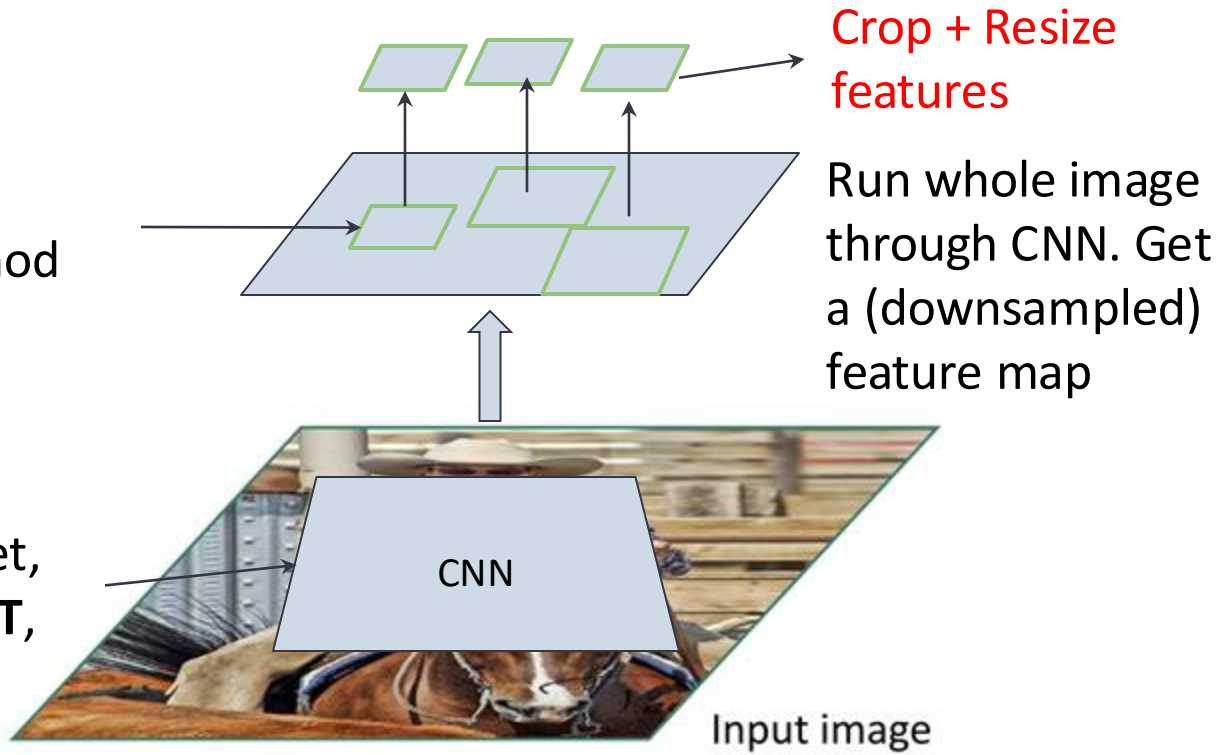


*Fast R-CNN. ICCV 2015

Object Detection-Two-stage method

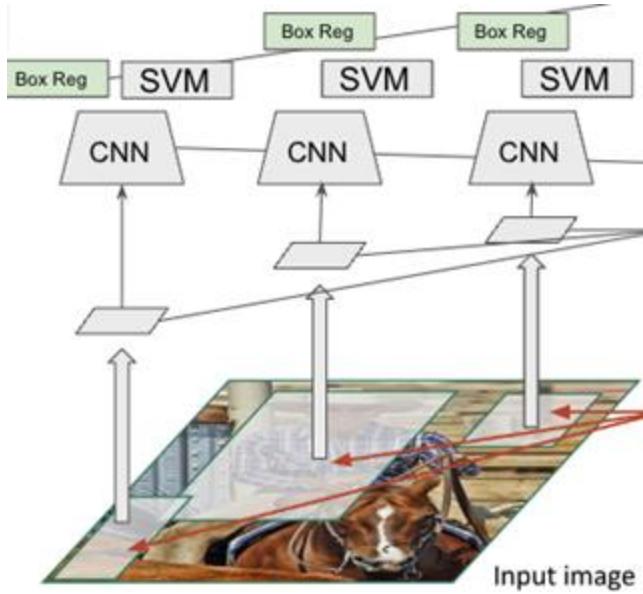
- Fast R-CNN

RoIs from a proposal method



"Backbone" network: AlexNet, VGG, ResNet, **ViT**, etc

Slow R-CNN



*Fast R-CNN. ICCV 2015

Object Detection-Two-stage method

- Fast R-CNN

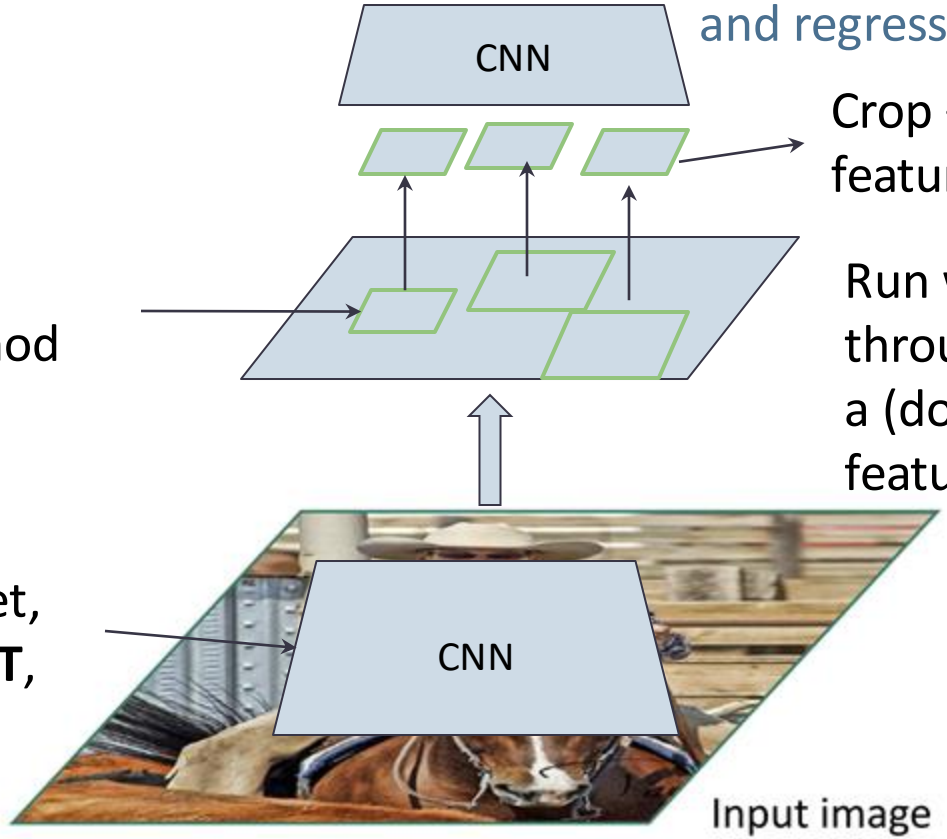
Another CNN for box classification and regression

Crop + Resize features

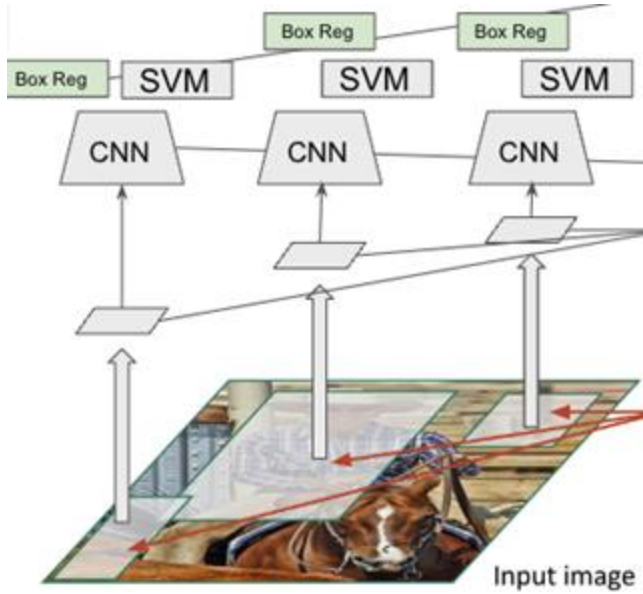
RoIs from a proposal method

Run whole image through CNN. Get a (downsampled) feature map

"Backbone" network: AlexNet, VGG, ResNet, **ViT**, etc



Slow R-CNN



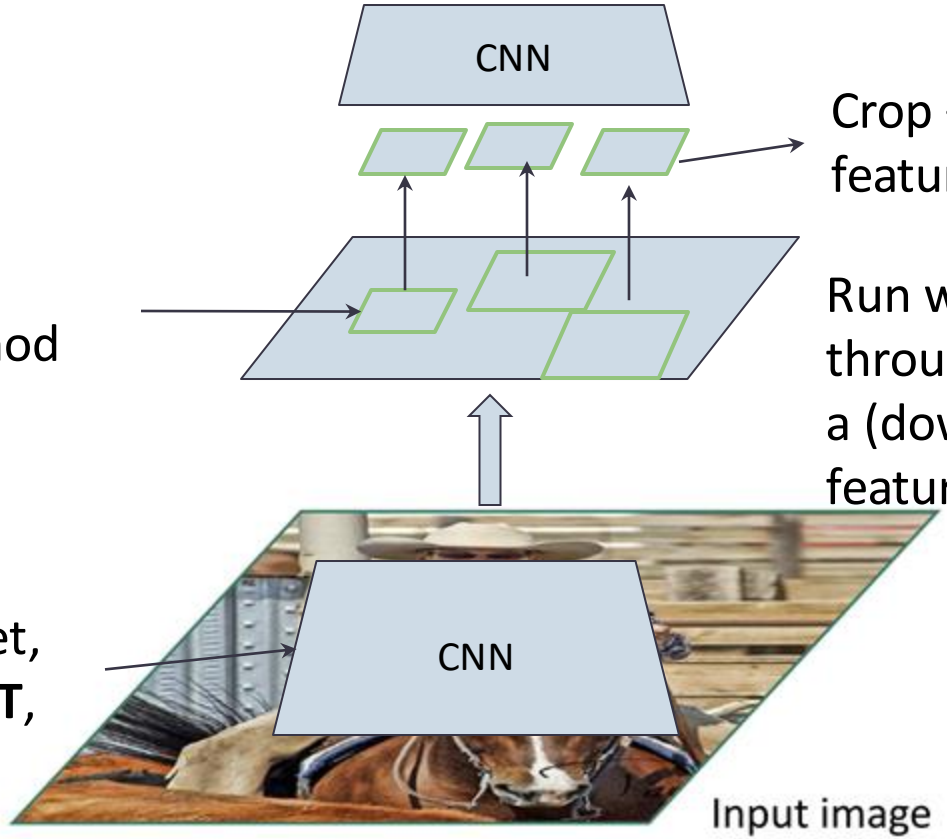
*Fast R-CNN. ICCV 2015

Object Detection-Two-stage method

- Fast R-CNN

Another CNN for box classification and regression

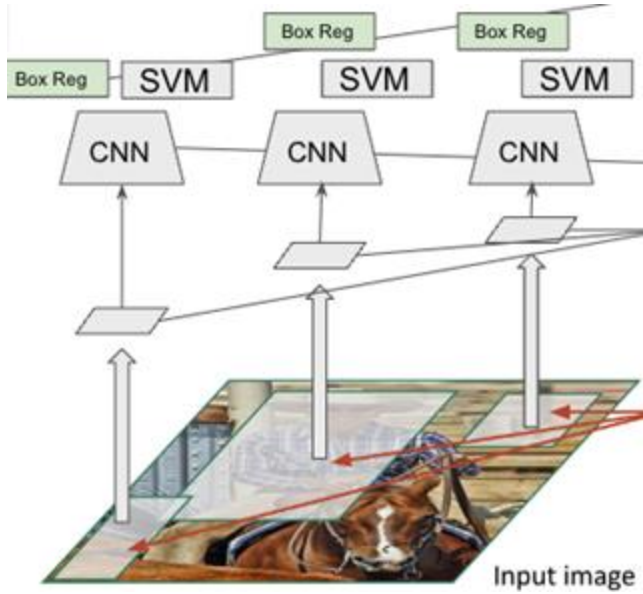
This CNN (usually 2-4 layers) is trained from scratch



"Backbone" network: AlexNet, VGG, ResNet, **ViT**, etc

This is pretrained from ImageNet

Slow R-CNN



*Fast R-CNN. ICCV 2015

Object Detection-Two-stage method

- Fast R-CNN

Another CNN for box classification and regression

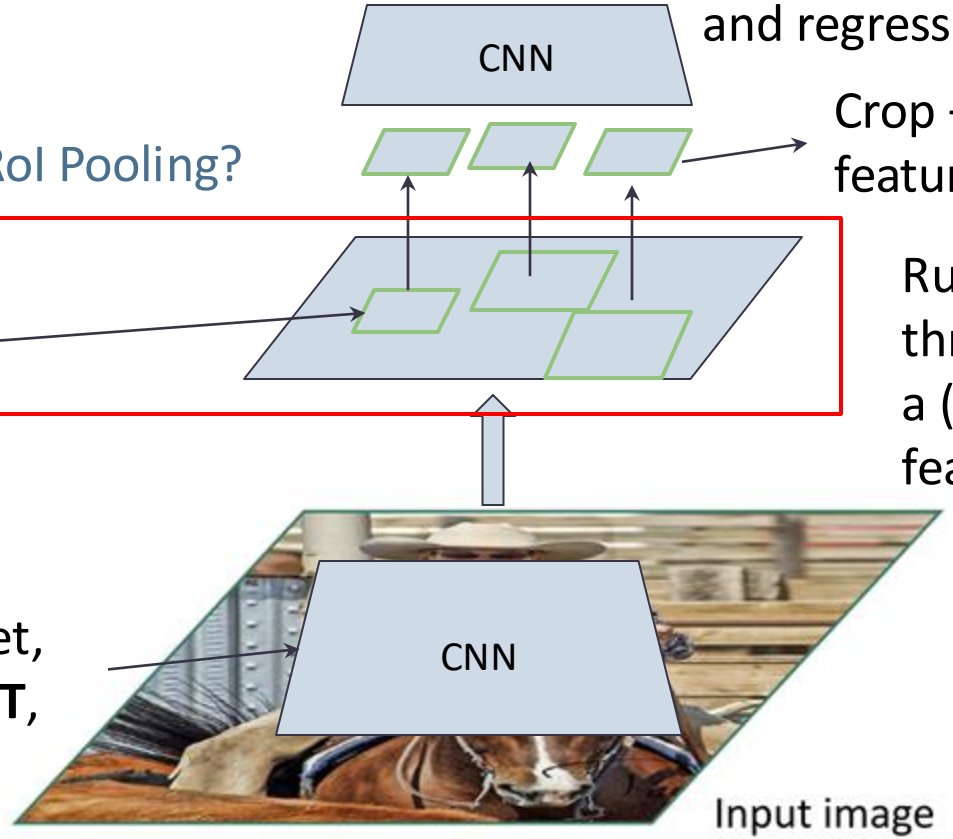
Crop + Resize features

How to do RoI Pooling?

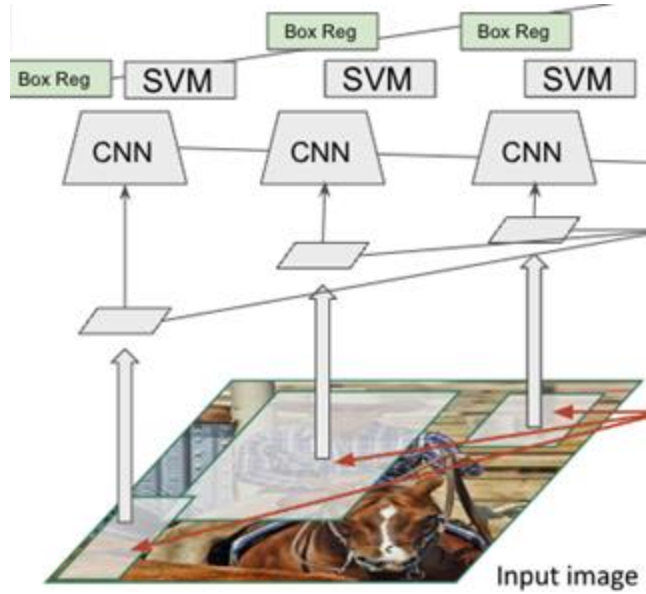
RoIs from a proposal method

Run whole image through CNN. Get a (downsampled) feature map

"Backbone" network: AlexNet, VGG, ResNet, **ViT**, etc



Slow R-CNN

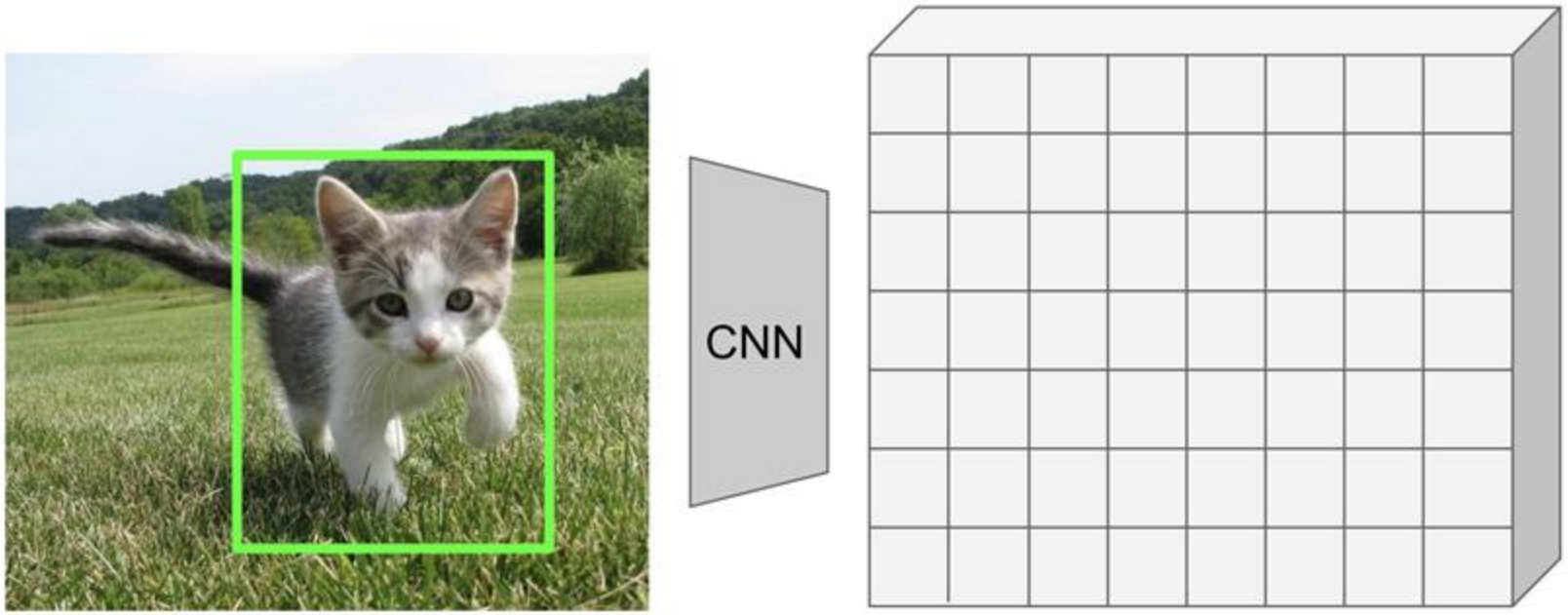


*Fast R-CNN. ICCV 2015

Object Detection-Two-stage method

- Cropping features: RoI Pool

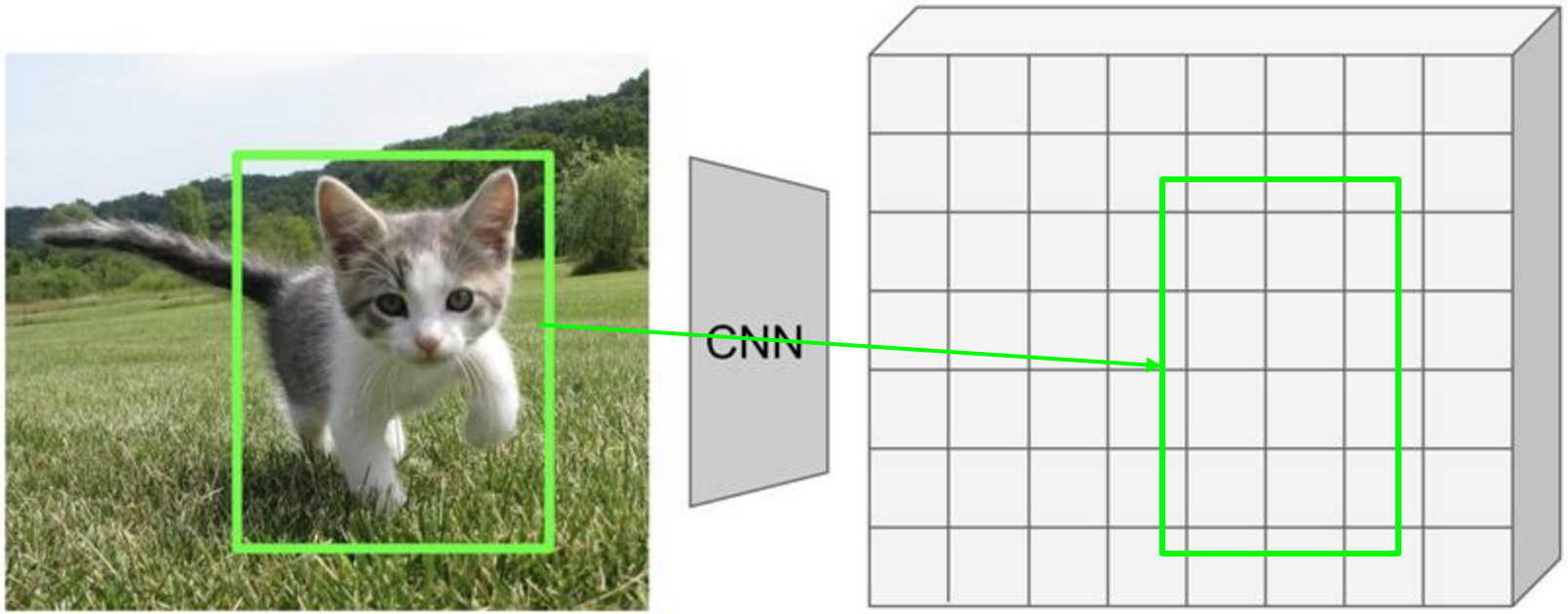
Suppose the CNN feature downsampled the image from 640x480 to 20x15 (32x down is common)



Object Detection-Two-stage method

- Cropping features: RoI Pool

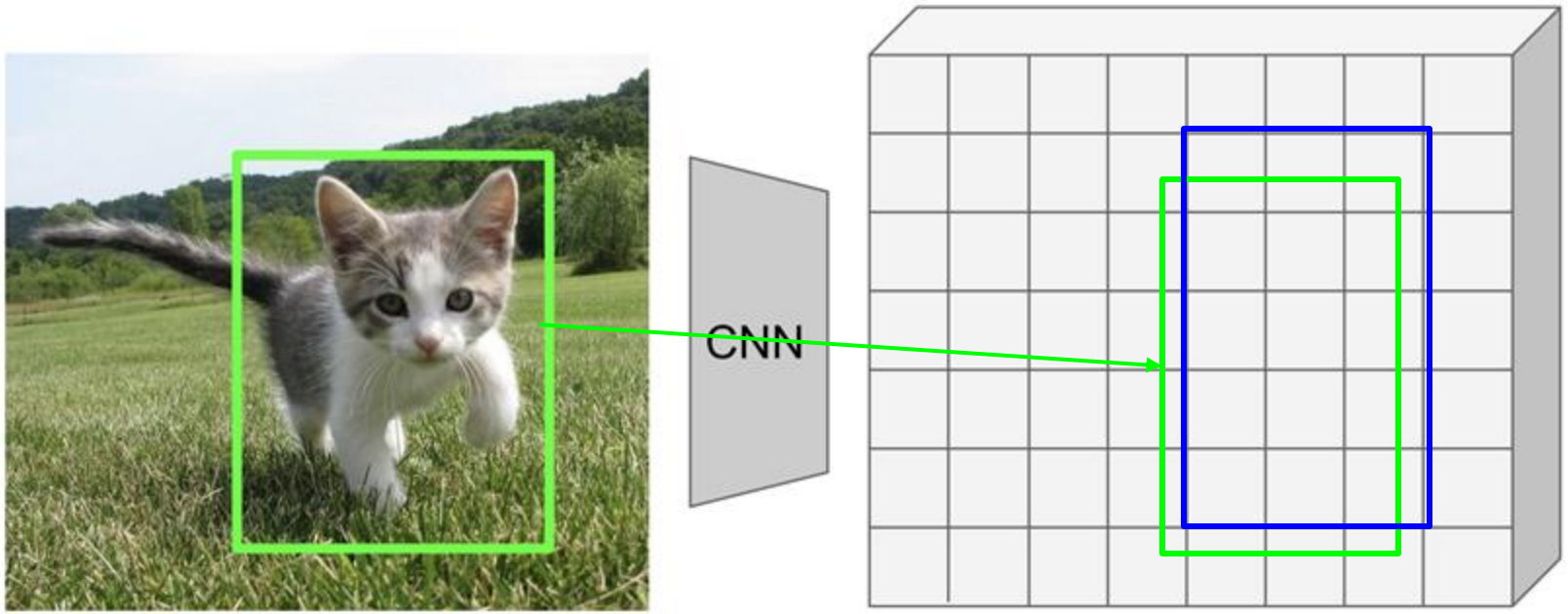
Project the box to the same relative location on the feature



Object Detection-Two-stage method

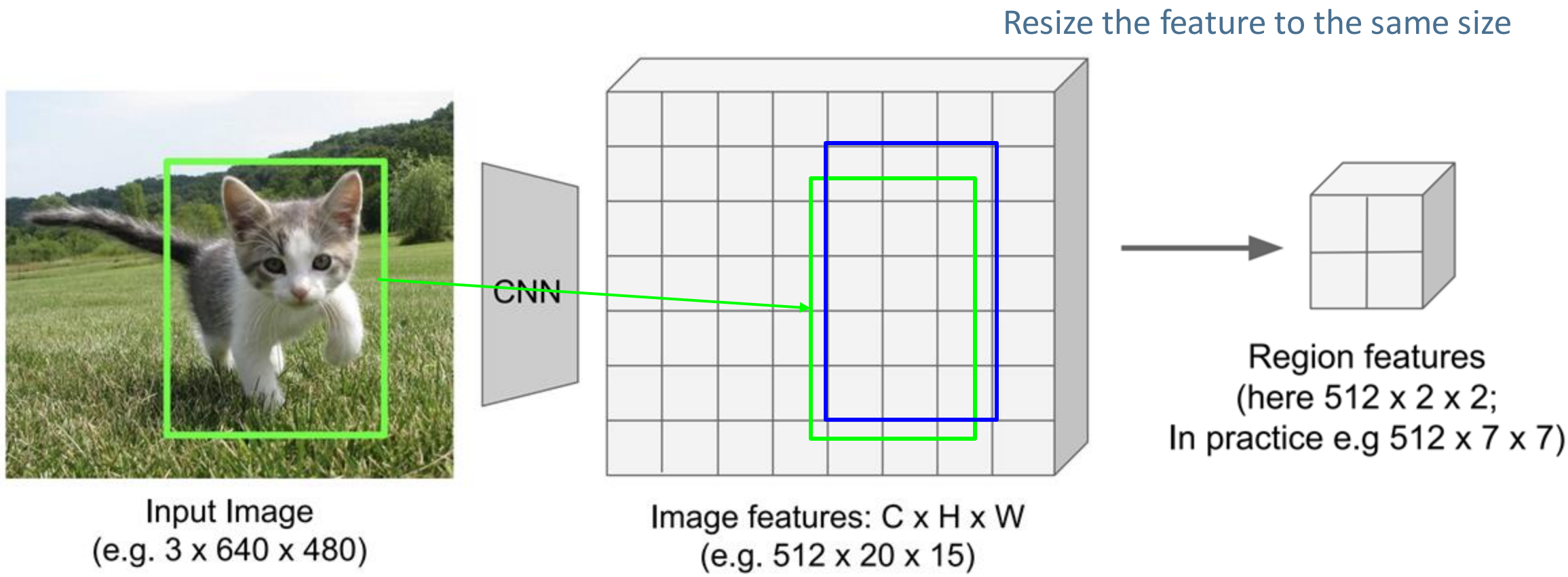
- Cropping features: RoI Pool

  "Snap" to the grid cells. You can then simply use indexing (e.g. feature[a:b, c:d]) to take from the feature numpy array



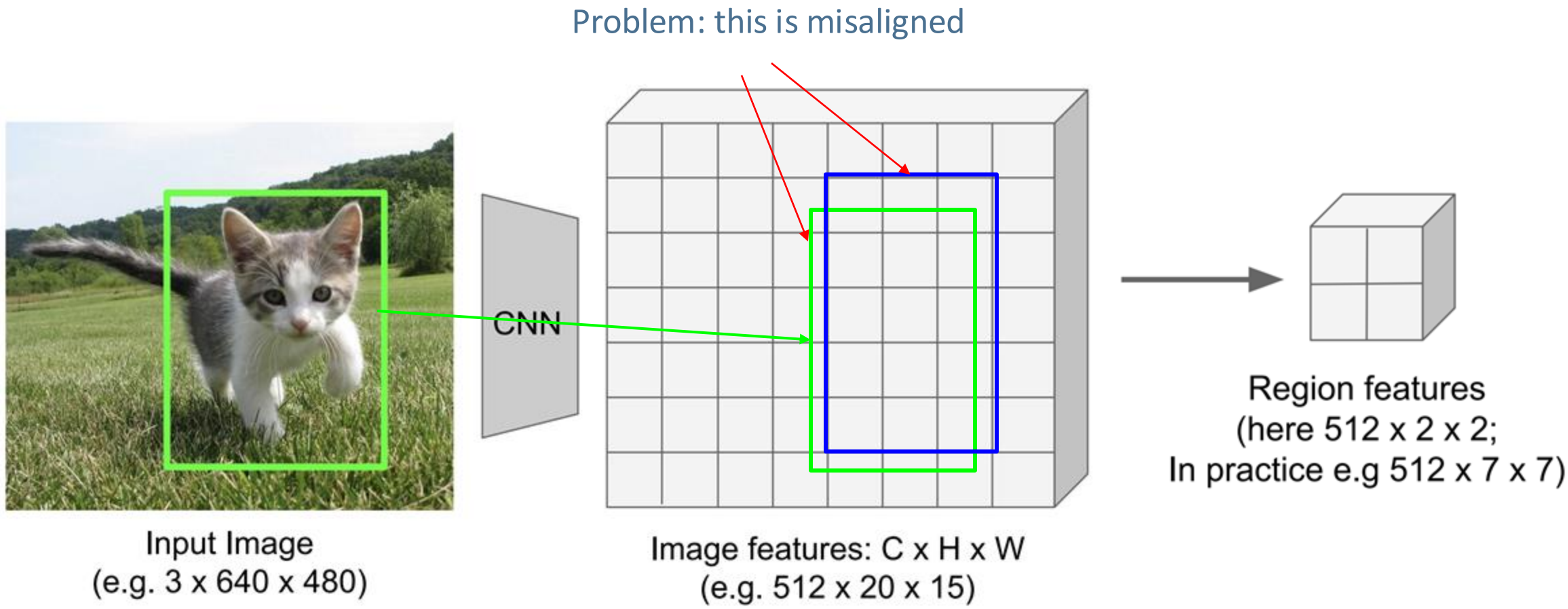
Object Detection-Two-stage method

- Cropping features: RoI Pool



Object Detection-Two-stage method

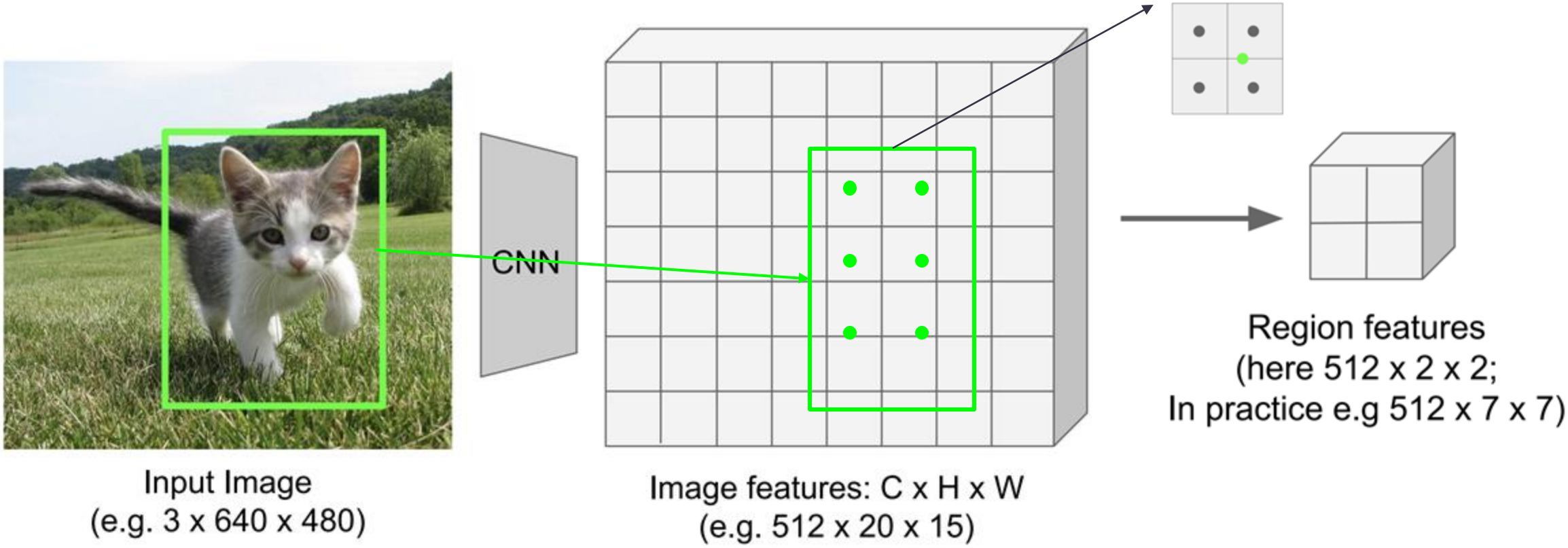
- Cropping features: RoI Pool



Object Detection-Two-stage method

- Cropping features: **RoI Align***

Perform bilinear interpolation, i.e. compute the green dot feature by weighted average of the surrounding four points based on L2 distance



* Mask R-CNN. ICCV 2017

Object Detection-Two-stage method

- R-CNN vs. Fast R-CNN

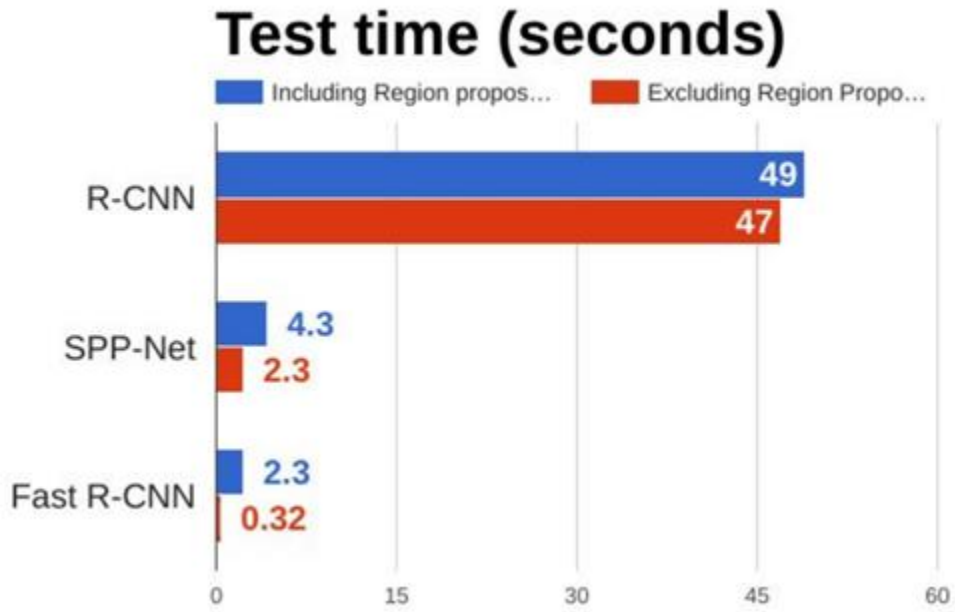
R-CNN's Limitation: Redundant Feature Extraction

Processing each of the ~2,000 region proposals through the CNN separately is highly inefficient.
This leads to long training times and slow inference.

Performance Analysis (See Chart):

- Region Proposal (e.g., Selective Search): ~2 seconds.
- CNN Feature Extraction & Classification: ~47 seconds (the main bottleneck).

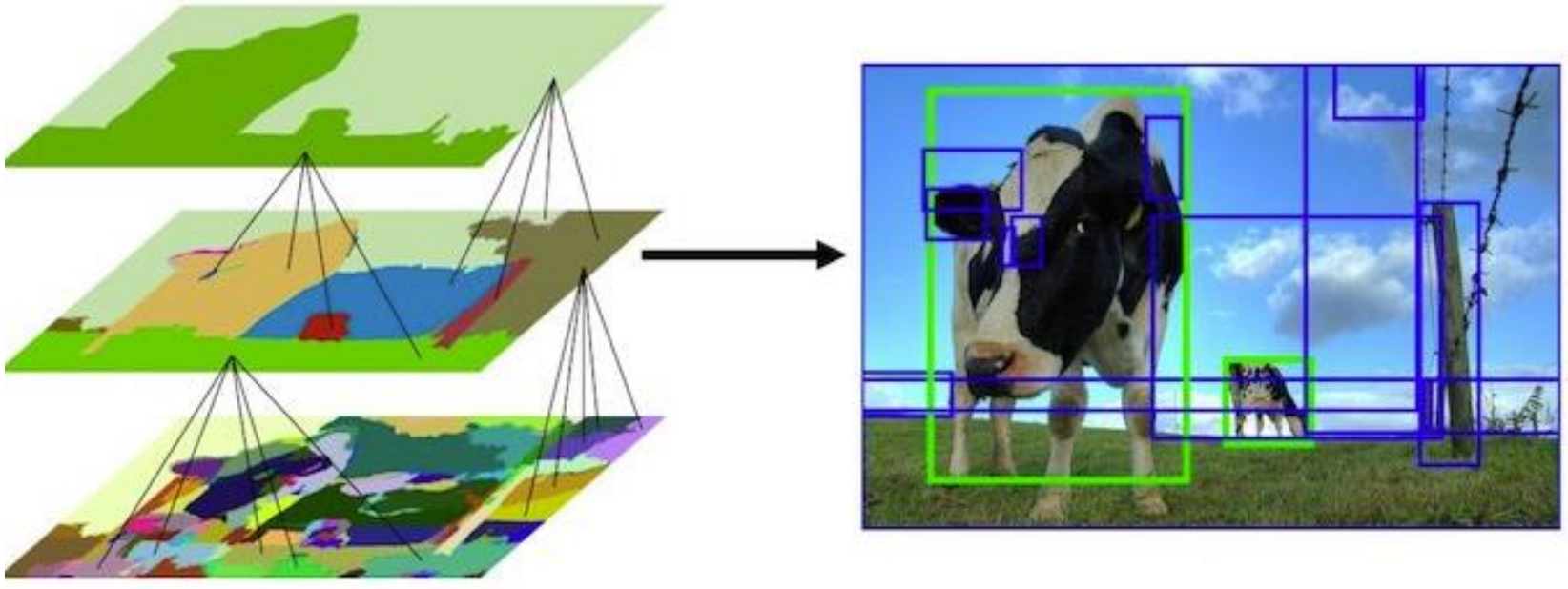
- Significant overlap exists between proposed regions. We can share computations by running the entire image through the CNN once.
- Fast R-CNN introduces a more efficient architecture that shares convolutional features across all proposals.



Object Detection -Two-stage method

- Using Selective Search to get region proposals
  - Selective search will need to **operate on pixels** to get segmentations first

Why not use a CNN to do that? -> a 50k citation idea

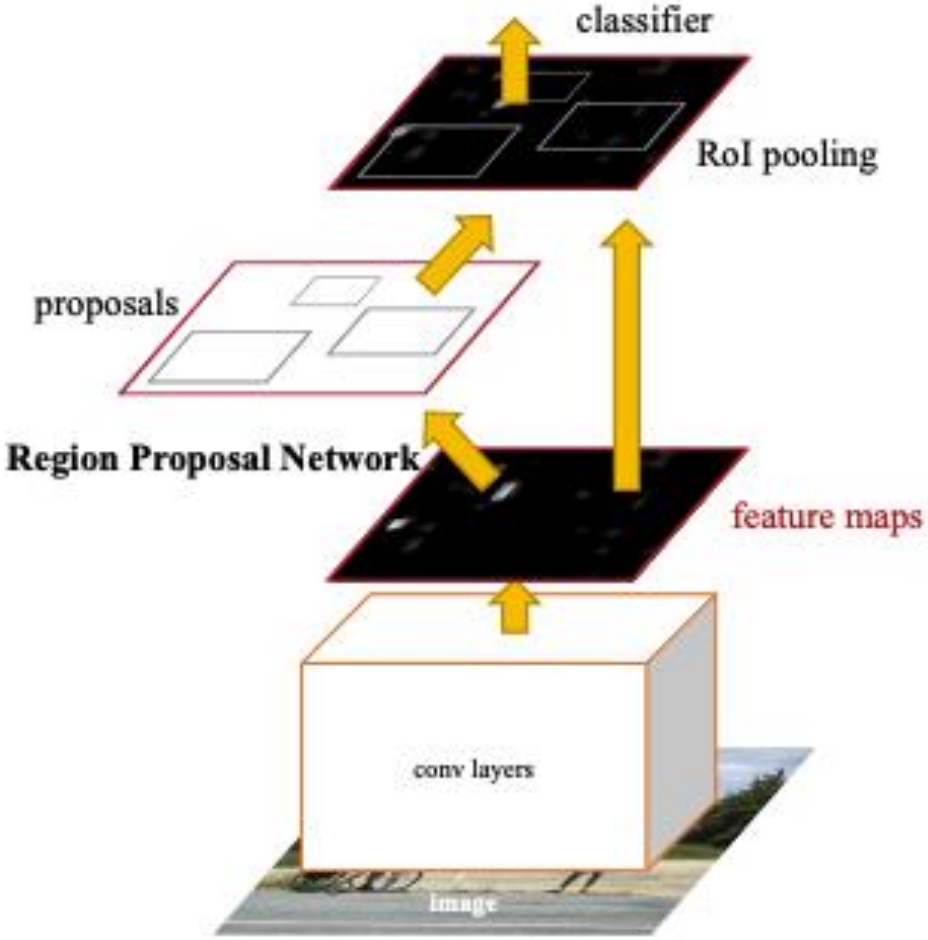


*Segmentation as Selective Search for Object Recognition. ICCV 2011

Object Detection-Two-stage method

- Faster R-CNN

  Introduce a Region Proposal Network (RPN) that learns to propose regions directly from the feature map.

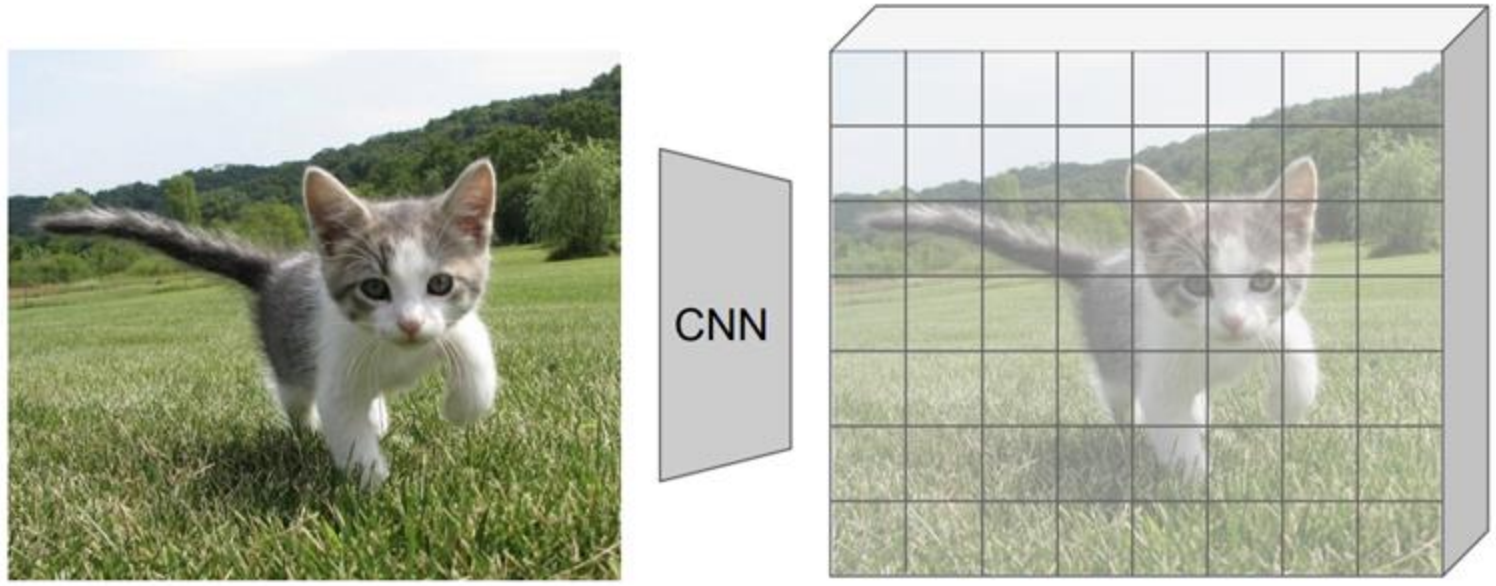


*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Region Proposal Network

Suppose the CNN feature downsampled the image from 640x480 to 20x15

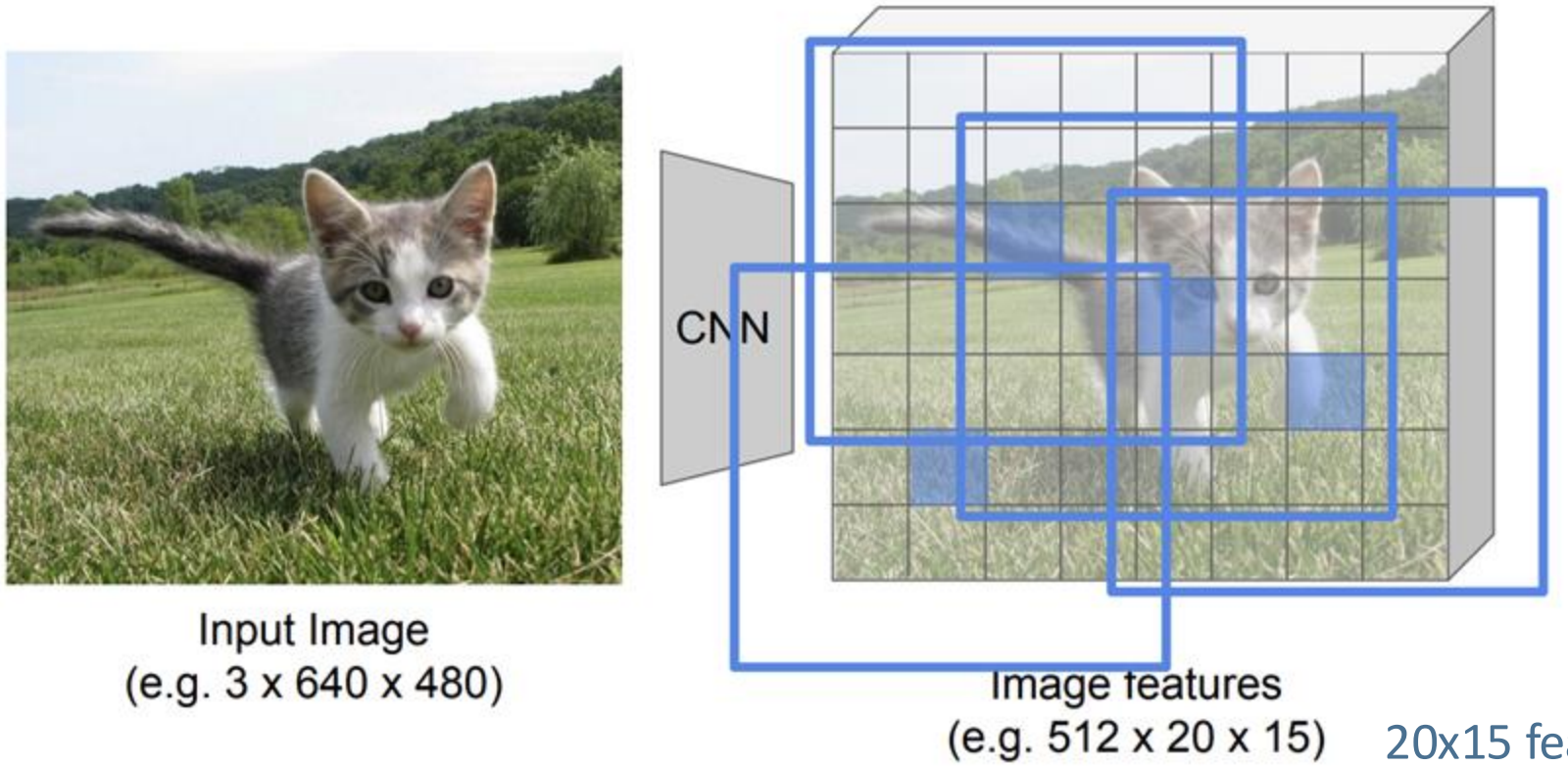


*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Region Proposal Network

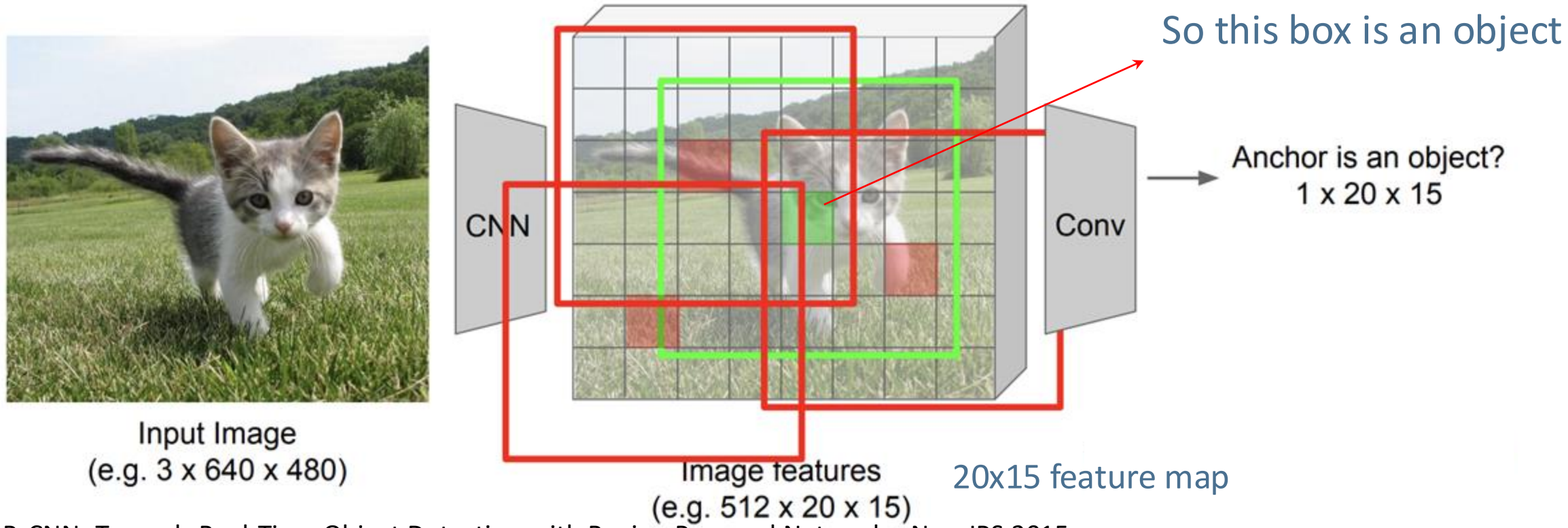
  Imagine an **anchor box** of fixed size (5x5, etc.) **at each point** in the feature map



*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

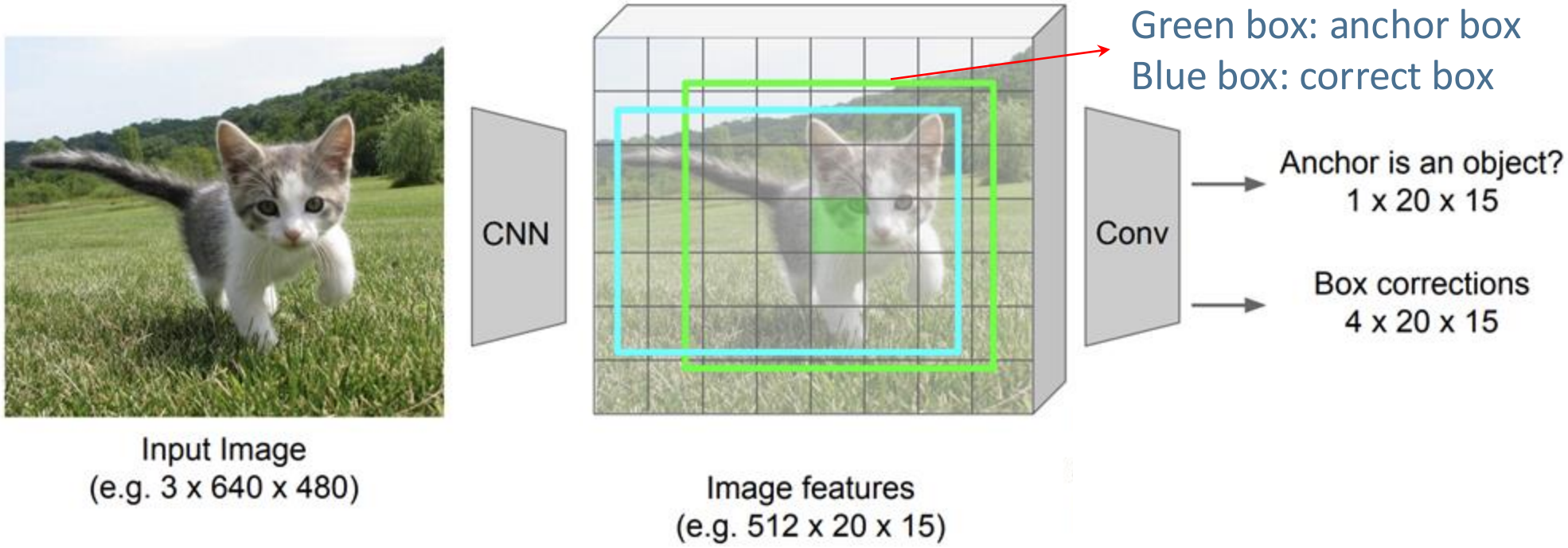
- Region Proposal Network
  - Imagine an **anchor box** of fixed size at each point in the feature map
  - At each point, predict whether the corresponding anchor contains an object (binary classification)



*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Region Proposal Network
  - Imagine an **anchor box** of fixed size at each point in the feature map
  - For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per feature)



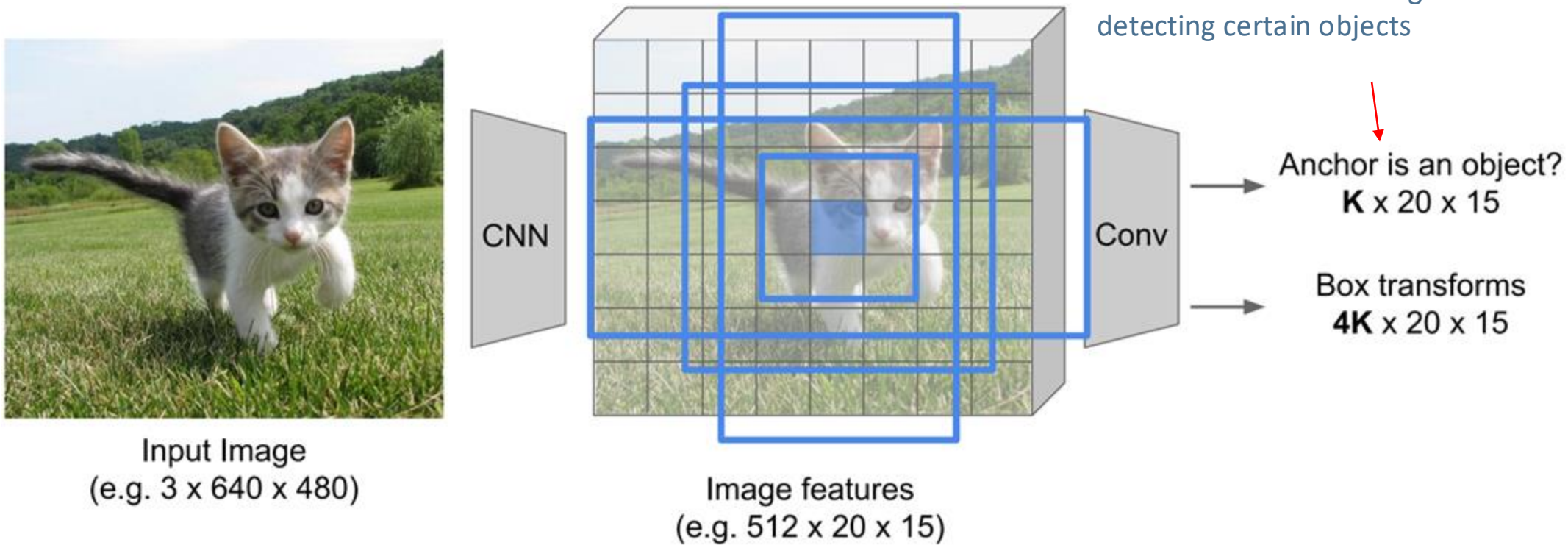
*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Region Proposal Network

Anchor boxes - predefined possible **K** regular boxes at each feature location

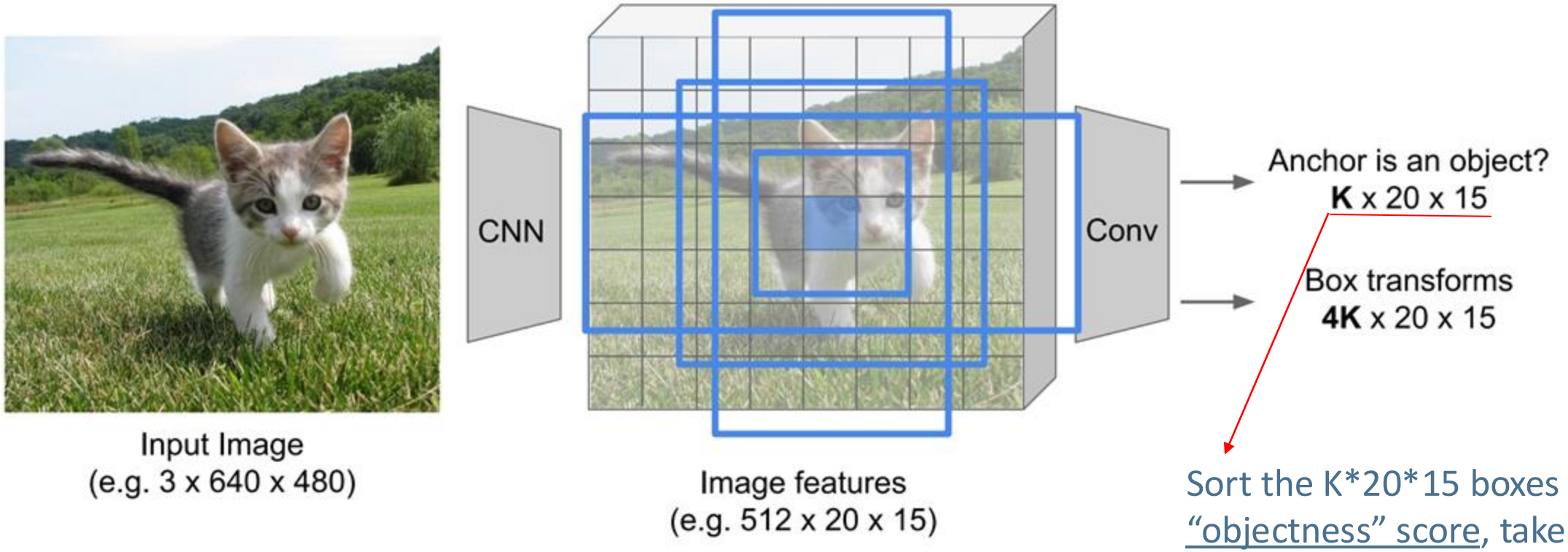
So not just 5x5, but also 7x4, 4x7, 3x3, etc.
Different ratios boxes might be easier for detecting certain objects



*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection- Two-stage method

- Region Proposal Network
  - **Anchor boxes** - predefined possible **K** regular boxes at each feature location

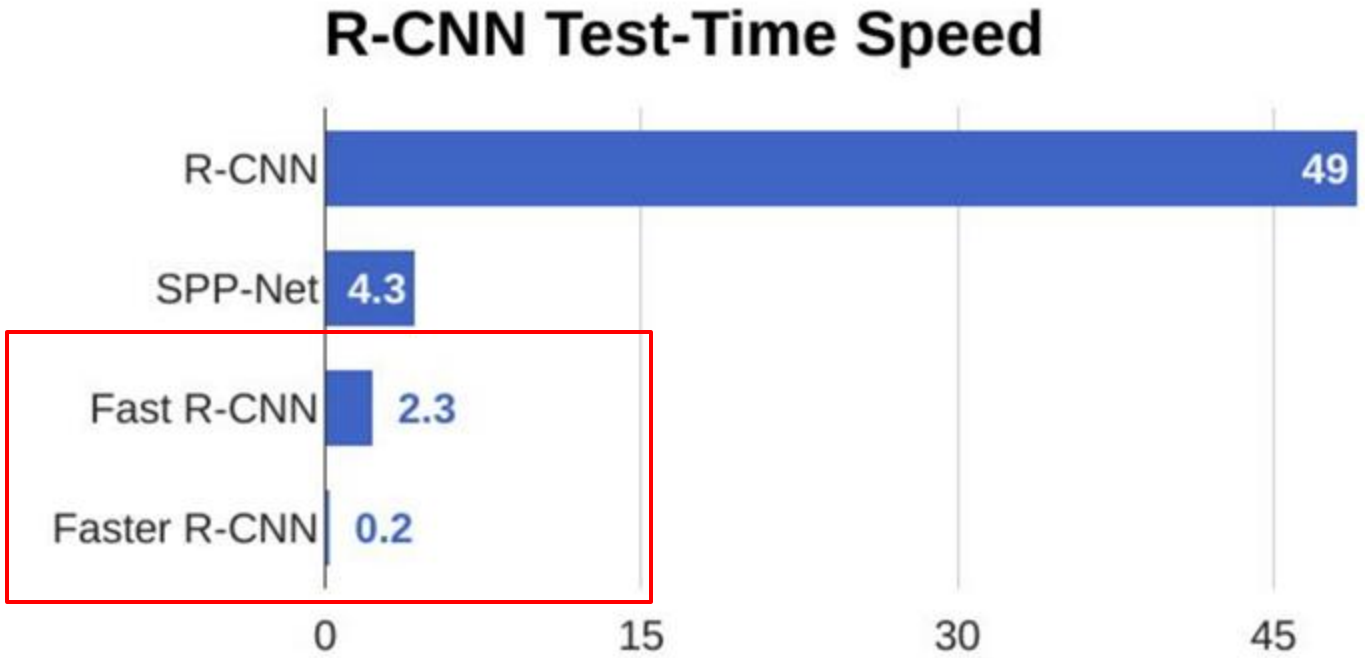


*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Faster R-CNN
  - Use CNN features for region proposal!
  - Called Region Proposal Network (RPN)
    - Use predefined **anchor boxes**
  - Other parts the same as Fast R-CNN

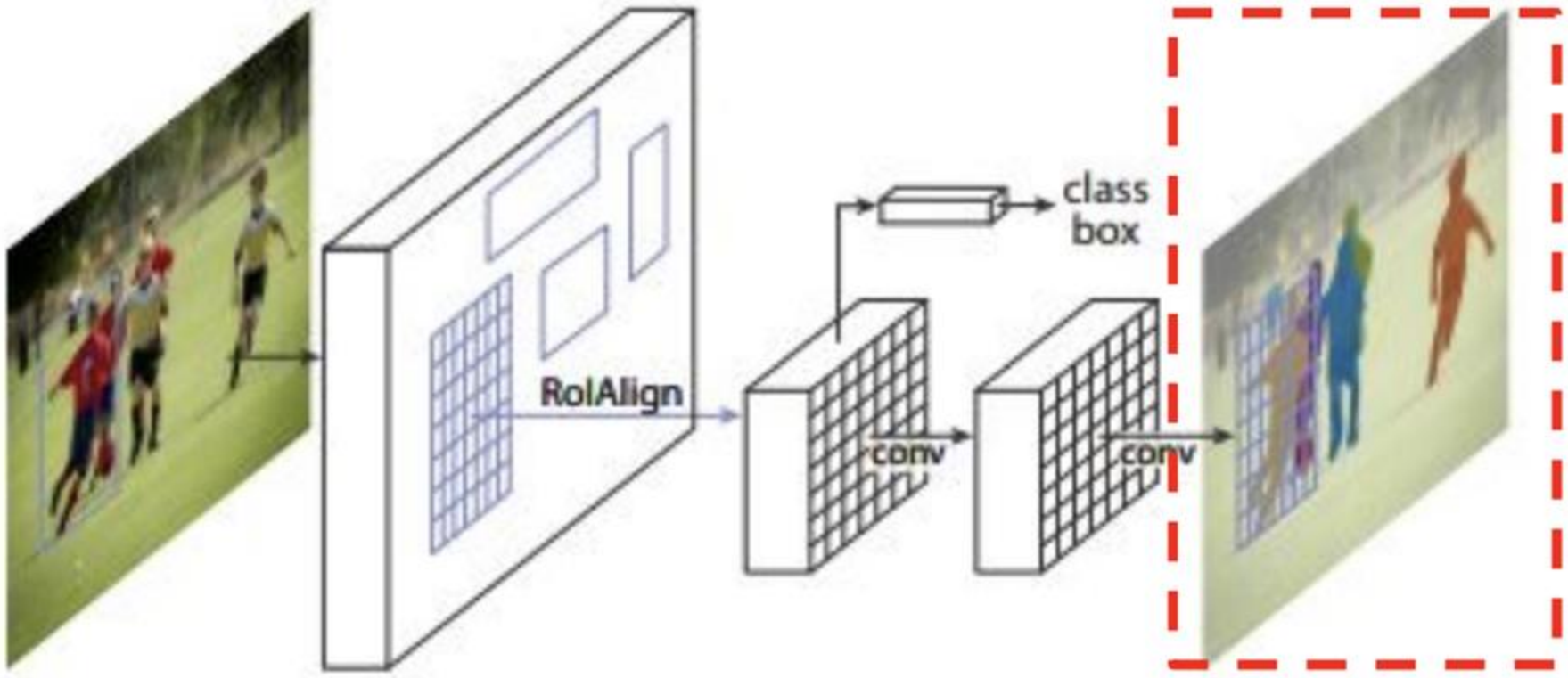
Now we save the 2 second from selective search..



*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS 2015

Object Detection-Two-stage method

- Mask R-CNN
  - Add a parallel branch to Faster R-CNN for predicting a segmentation mask for each object.
  - Mask R-CNN outputs **a binary mask** for each RoI on top of the Faster R-CNN

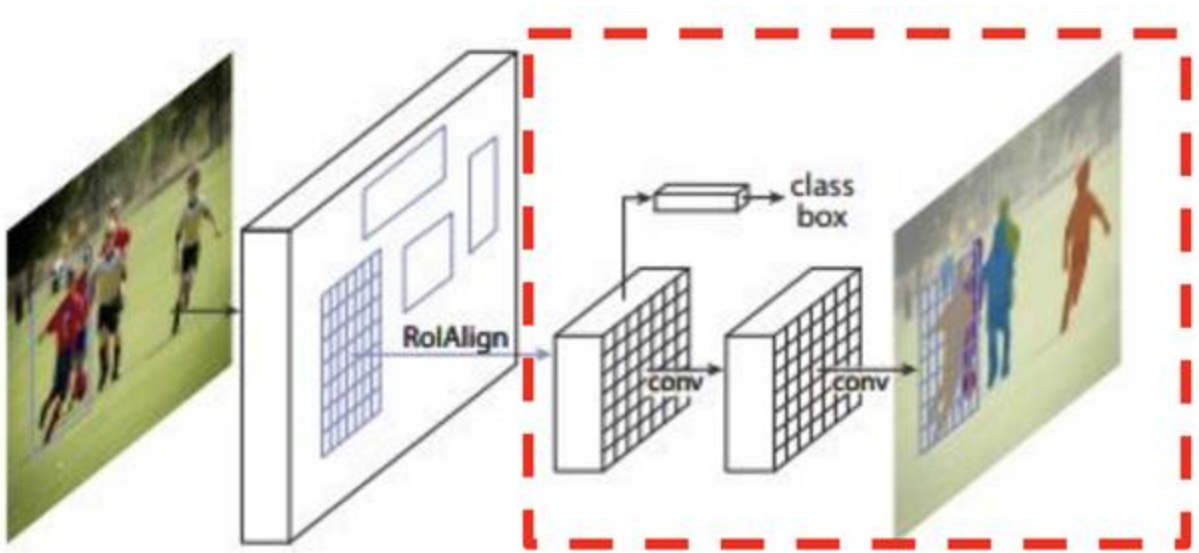


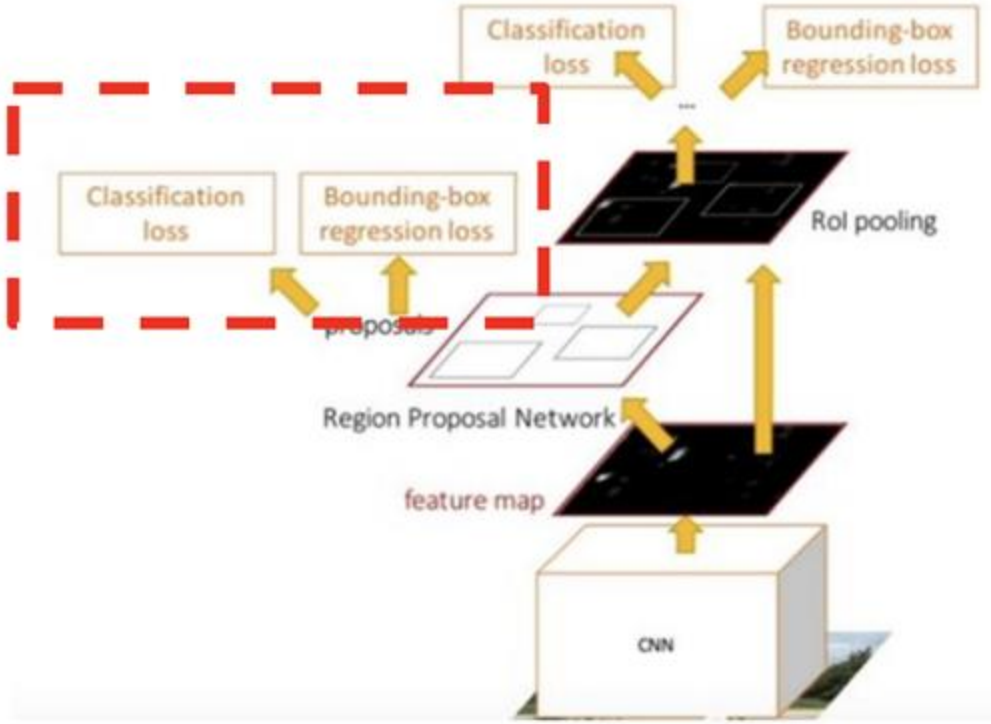
So boxes are not accurate enough for objects. We need pixel-wise detections, i.e., masks

*Mask R-CNN. ICCV 2017

Object Detection-Two-stage method

- Mask R-CNN
  - Mask R-CNN adopts the same two-stage procedure with the identical RPN first stage
  - Mask R-CNN outputs a **binary mask** for each RoI for each class in parallel to predicting the class and box offset in the second stage





*Mask R-CNN. ICCV 2017

Object Detection-Two-stage method

- Mask R-CNN
  - Loss function
  - **The mask branch predicts encodings for K classes of each RoI** so that the network can generate masks without competition, which decouples mask and class prediction
    - So you can classify the object box first, then select that the mask for that object class

$$L = L_{cls} + L_{box} + L_{mask}$$

*Mask R-CNN. ICCV 2017

Object Detection-Two-stage method

- Mask R-CNN
  - Experiment - Instance segmentation

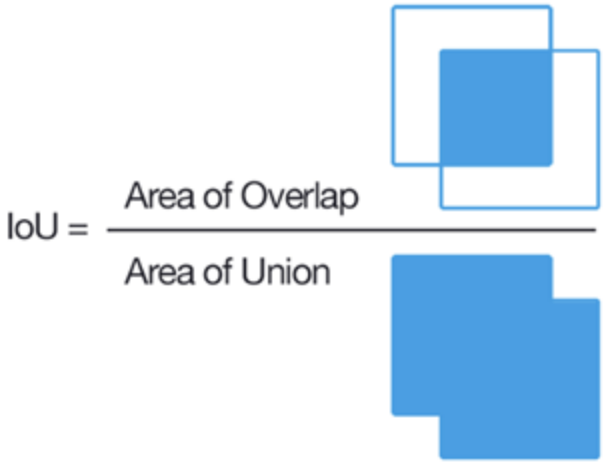
*Mask R-CNN. ICCV 2017

Object Detection-Two-stage method

- Two-stage
  - Region proposal - get regions-of-interest (RoIs) - Region Proposal Network (RPN)
    - Run once per image
  - Do classification and regression on the RoIs
    - Run on every RoIs
    - We can also do instance segmentation so we get a mask of the object
- Key
  - Feature sharing
  - RoI feature cropping
- Training - Multi-task losses
  - RPN classification and regression
  - Box classification and regression
- Post-processing and evaluation
  - Intersection-Over-Union
  - Non-maximum suppression
  - Average Precision

Object Detection - Box Overlapping

- During the training and evaluation, we would like to determine **positives and negatives** from the object hypotheses with **ground-truth bounding boxes**
- Intersection-over-Union (IoU) is used to determine the overlapping between two bounding boxes
- IoU = 1 means 100% overlapping between two boxes, and IoU = 0 means 0% overlapping between two boxes



Object Detection - Non-maximum Suppression

- Problem: Sometimes the network predicts multiple bounding boxes for one object. How to select the best one?
  - NMS! - Basically the high score boxes will “suppress” lower score boxes they overlap. (no merging boxes)

For each class, start with the highest confidence

0. Before NMS

1. Select BBox with highest score

2. Calculate IOU

3. Remove BBox if IOU > threshold

Object Detection - Evaluation Metric

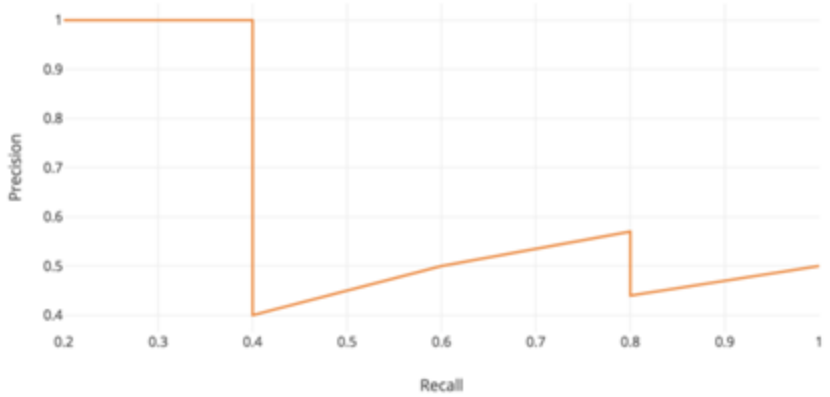
- **Detection Correctness (IoU Threshold):**
  - A predicted bounding box is a **True Positive (TP)** if its Intersection-over-Union (IoU) with a ground-truth box exceeds a threshold (typically **> 0.5**).
  - Otherwise, it is a **False Positive (FP)**.
- **Evaluation Metrics:**
  - **Precision:** Of all boxes predicted, how many are correct?
    Recall: Of all ground-truth objects, how many did we find?
    The Precision-Recall Trade-off:
  - Predictions are ranked by confidence score.
  - As we include more predictions (lower confidence threshold):
    - **Recall Increases** (we find more true objects).
    - **Precision Decreases** (we also include more false alarms).

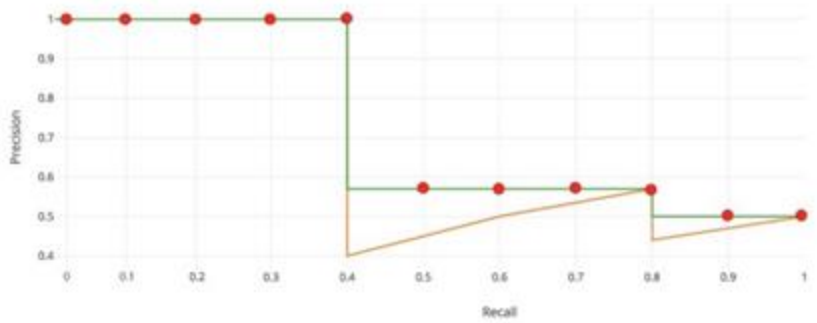
Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

$$\text{Preciesion} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Object Detection - Evaluation Metric

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0





Summarize the entire Precision-Recall curve into one number to compare models easily.

The Method: Average Precision (AP):

- Plot the Precision-Recall curve as the confidence threshold varies.
- Sample the curve at multiple recall levels (e.g., from 0.0 to 1.0).
- At each recall level, use the **maximum precision** achieved at that recall or higher.
- The **AP** is the average of these maximum precision values across all recall levels.

From AP to mAP:

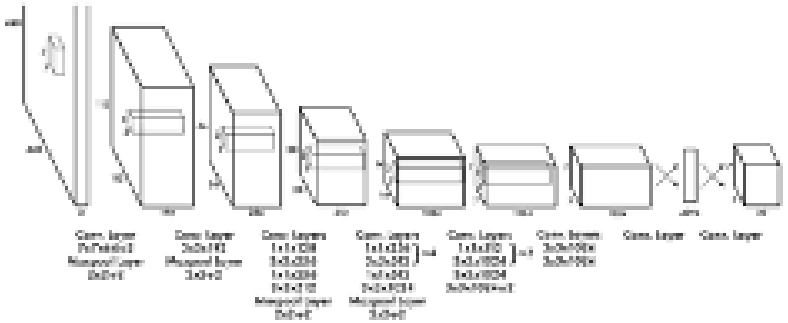
- **AP** evaluates performance on a **single object class**.
- **mAP (Mean Average Precision)** is the average of AP scores across **all object classes** (e.g., 60 classes in a dataset), providing one overall performance score.

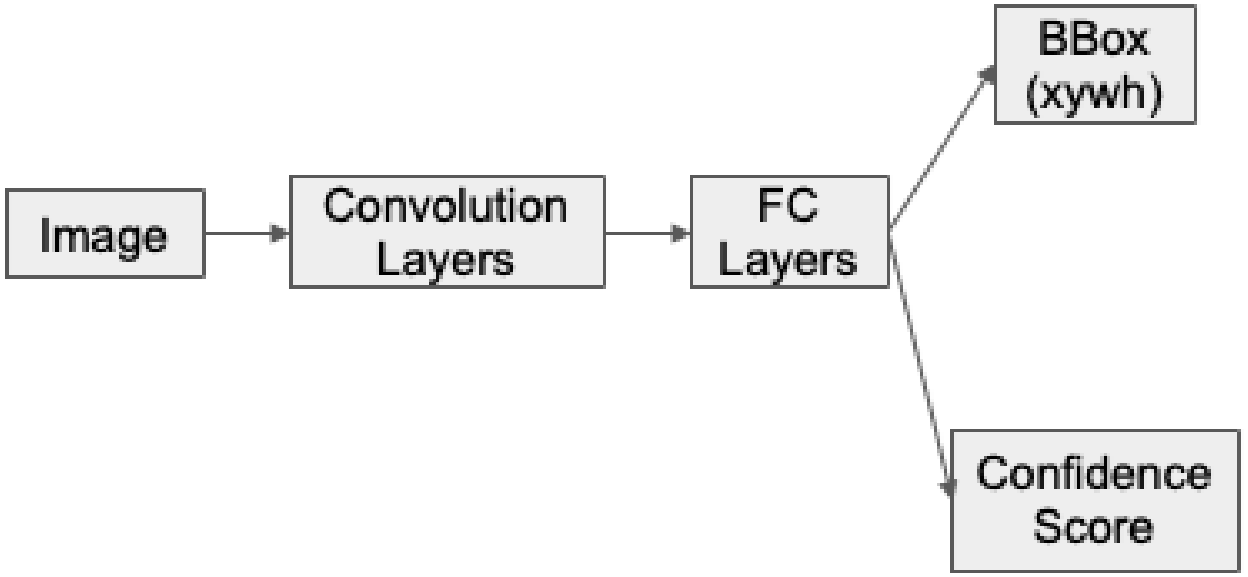
*https://docs.kolena.com/metrics/average-precision/

Object Detection - One-stage Model

Key concept: Frame object detection as a single regression problem from image pixels to bounding box coordinates and class probabilities.

- **YOLO**
  - an end-to-end (image→2d-bbox) regression problem (“unified detection”)
  - one-stage detector



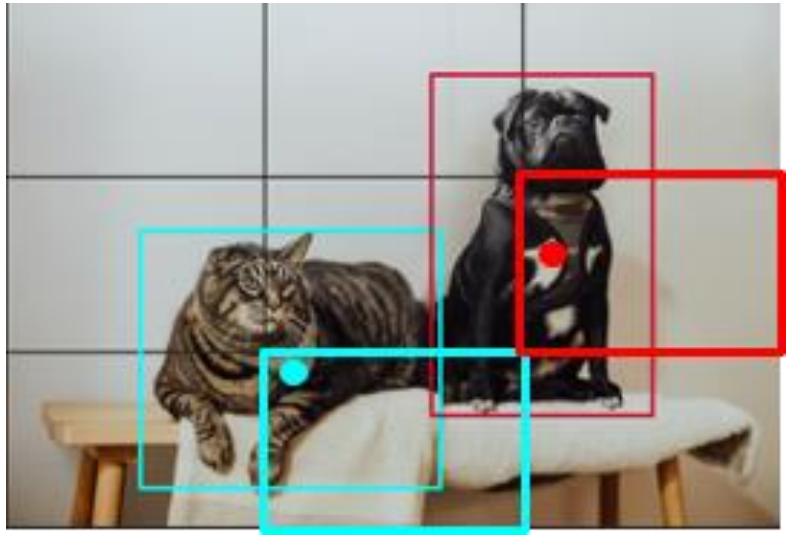


source: You Only Look Once: Unified, Real-Time Object Detection. 2015

Object Detection - One-stage Model

- YOLO
  - Divide input image into an SxS grid (or Height x Width)
  - Each grid is responsible for detecting objects whose centers lie in that grid
  - Each grid predicts B bounding boxes (regression) and their confidence
  - In total, final prediction is a S x S x (B*5+C) tensor
    - [x, y, w, h, conf]

C = #classes
S = grid size
B = #bboxes



Object Detection - One-stage Model

- YOLO - Comparison
  - FPS: frame-per-second, measuring actual inference speed
  - mAP: mean average precision, higher better

		mAP	FPS
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

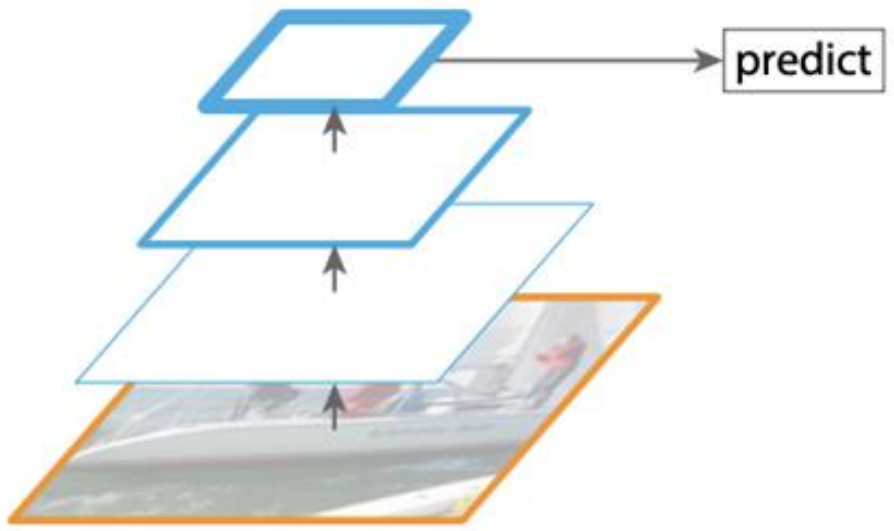
Trading speed for mAP

Object Detection - One-stage Model

- Improvements of YOLO since 2015…
- **YOLOv2** YOLO9000:Better, Faster, Stronger. Joseph Redmon & Ali Farhad. (2017)
- **YOLOv3** YOLOv3: An Incremental Improvement. Joseph Redmon & Ali Farhad. (2018)
- **YOLOv4** YOLOv4: Optimal Speed and Accuracy of Object Detection. Alexey Bochkovski, Chien-Yao Wan & Hong-Yuan Mark Liao. (2020)
  - new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, etc.
- **YOLOv5** (GitHub)
- **YOLOv6**
- **YOLOv7, 2022 - CVPR 2023**
- **YOLOv8?? - https://github.com/ultralytics/ultralytics**
- **YOLOv9 by YOLOv4 authors in 2024**

Object Detection - Feature Pyramid Model

- Single feature map may suffer from low-resolution



(b) Single feature map

Challenge: Objects appear at different scales. How does a detector handle a tiny person vs. a large car?

source: Feature Pyramid Networks for Object Detection. CVPR 2017

Object Detection - Feature Pyramid Model

- Single feature map may suffer from low-resolution

224x224 input

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

2x down

2x down

2x down..

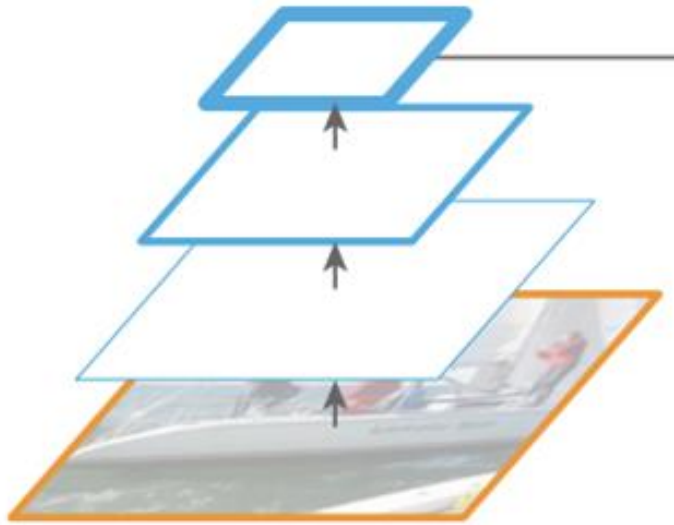
32x down
In total.
224x224
becomes
7x7

The output size of each stage of ResNet

Object Detection - Feature Pyramid Model

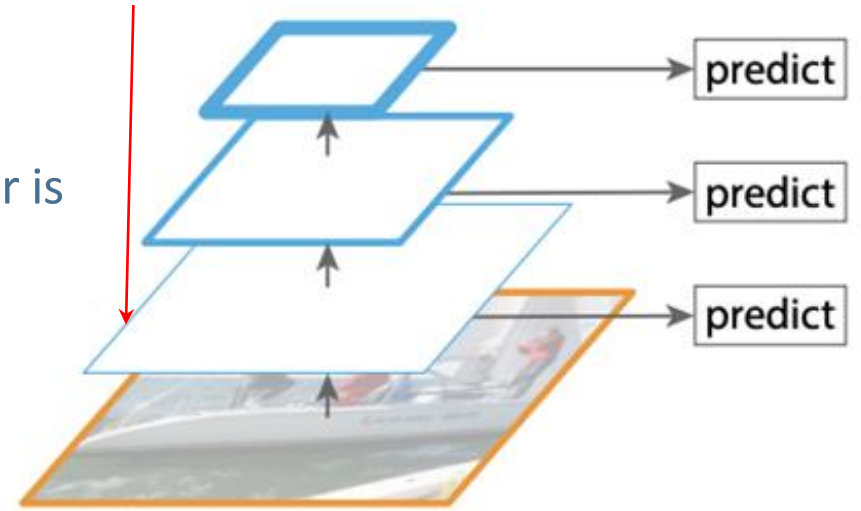
- Single feature map may suffer from low-resolution
  - Use multiple feature maps according to object size

Smaller boxes should use higher resolution features. For example from here



(b) Single feature map

But this layer is too shallow

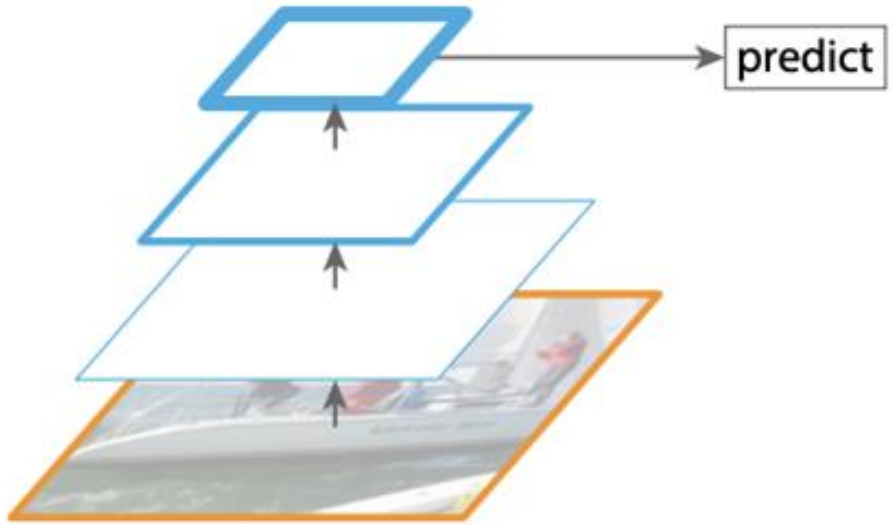


(c) Pyramidal feature hierarchy

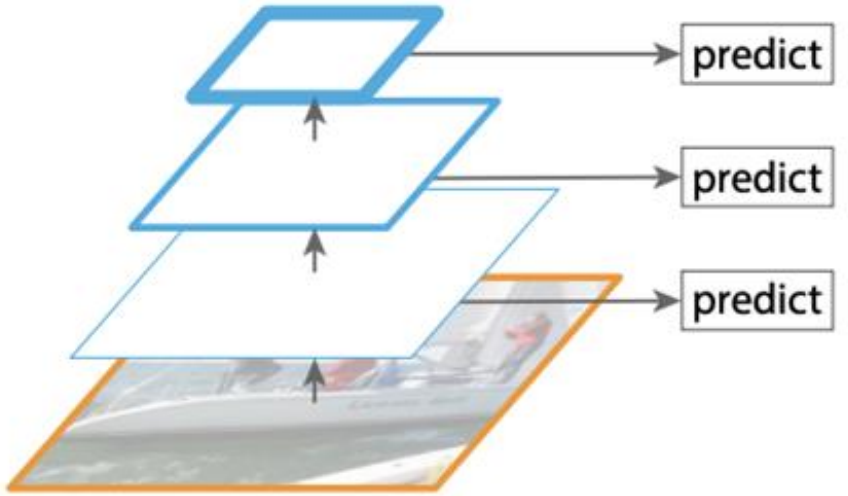
Object Detection - Feature Pyramid Model

- Single feature map may suffer from low-resolution
  - Use multiple feature maps according to object size

But deeper layer may have better semantics. Can we have both?



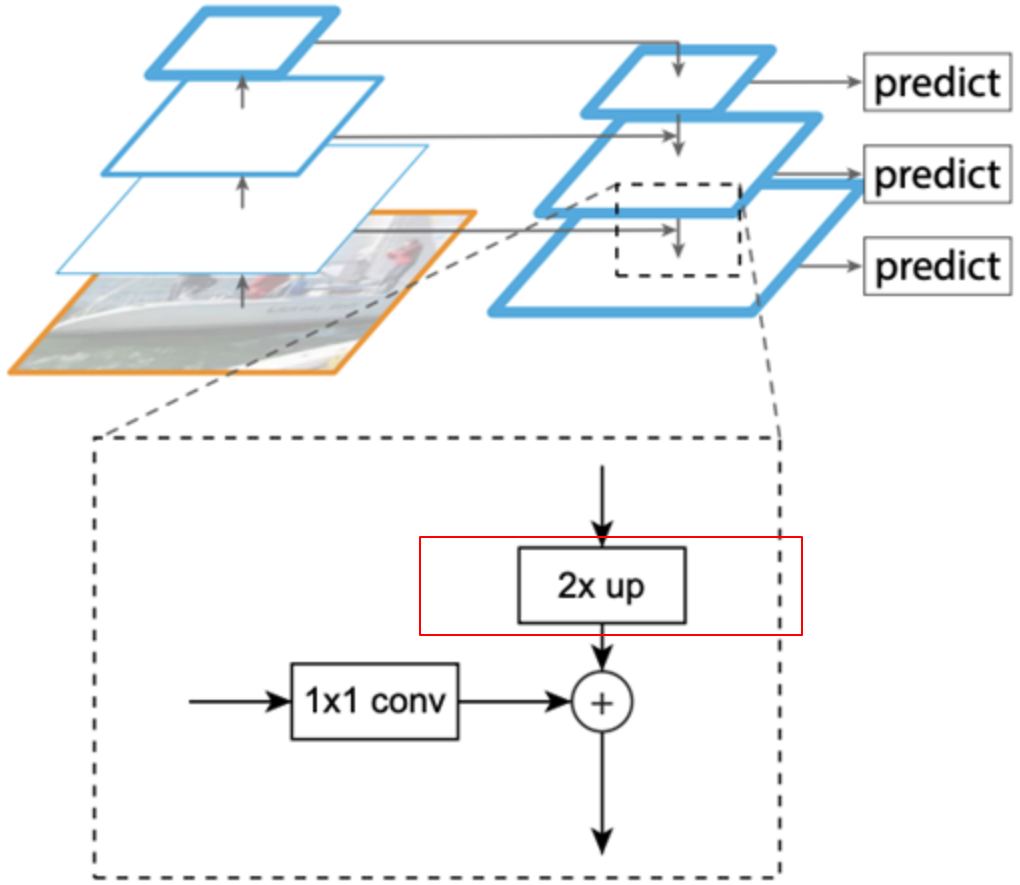
(b) Single feature map



(c) Pyramidal feature hierarchy

Object Detection - Feature Pyramid Model

- Single feature map may suffer from low-resolution
  - Fuse different feature maps!

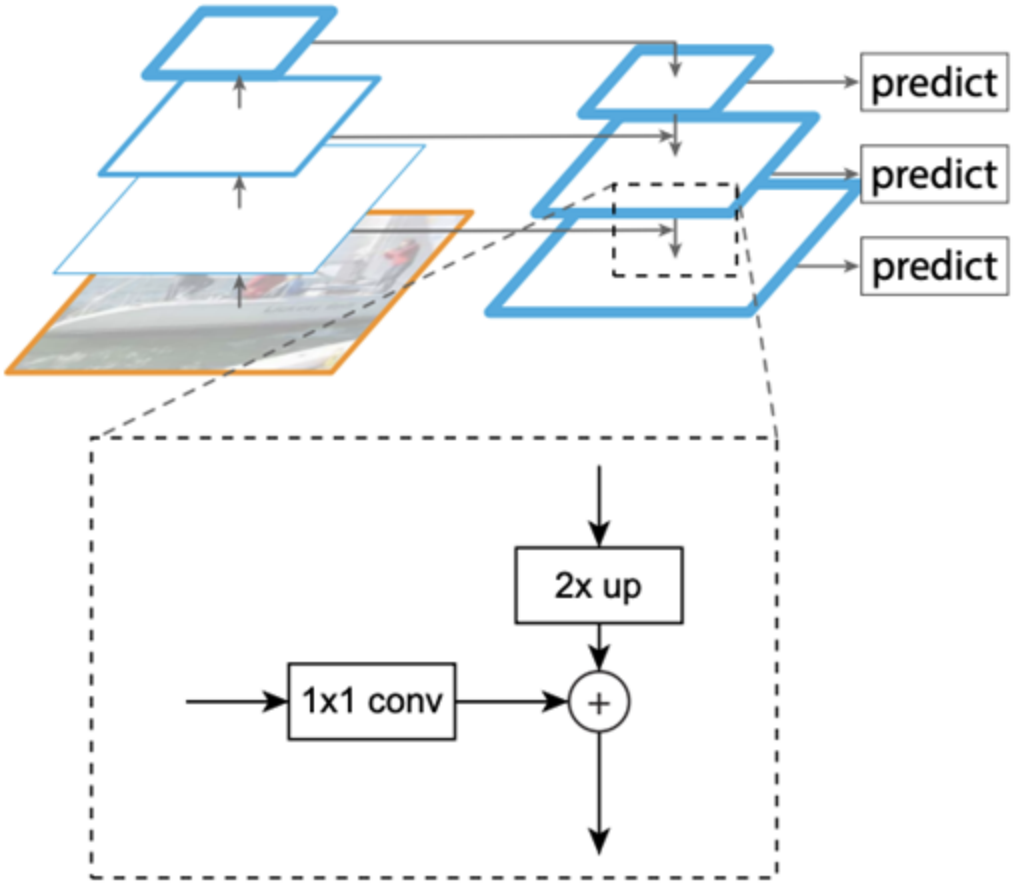


This uses simple nearest neighbor unpooling (repeating values) to up-sample the feature map.

Object Detection - Feature Pyramid Model

- Single feature map may suffer from low-resolution
  Fuse different feature maps!

Now then the faster RCNN, Mask RCNN can simply **select different layer's CNN feature** according to the bounding box size. All other parts stay the same and detection results are better!

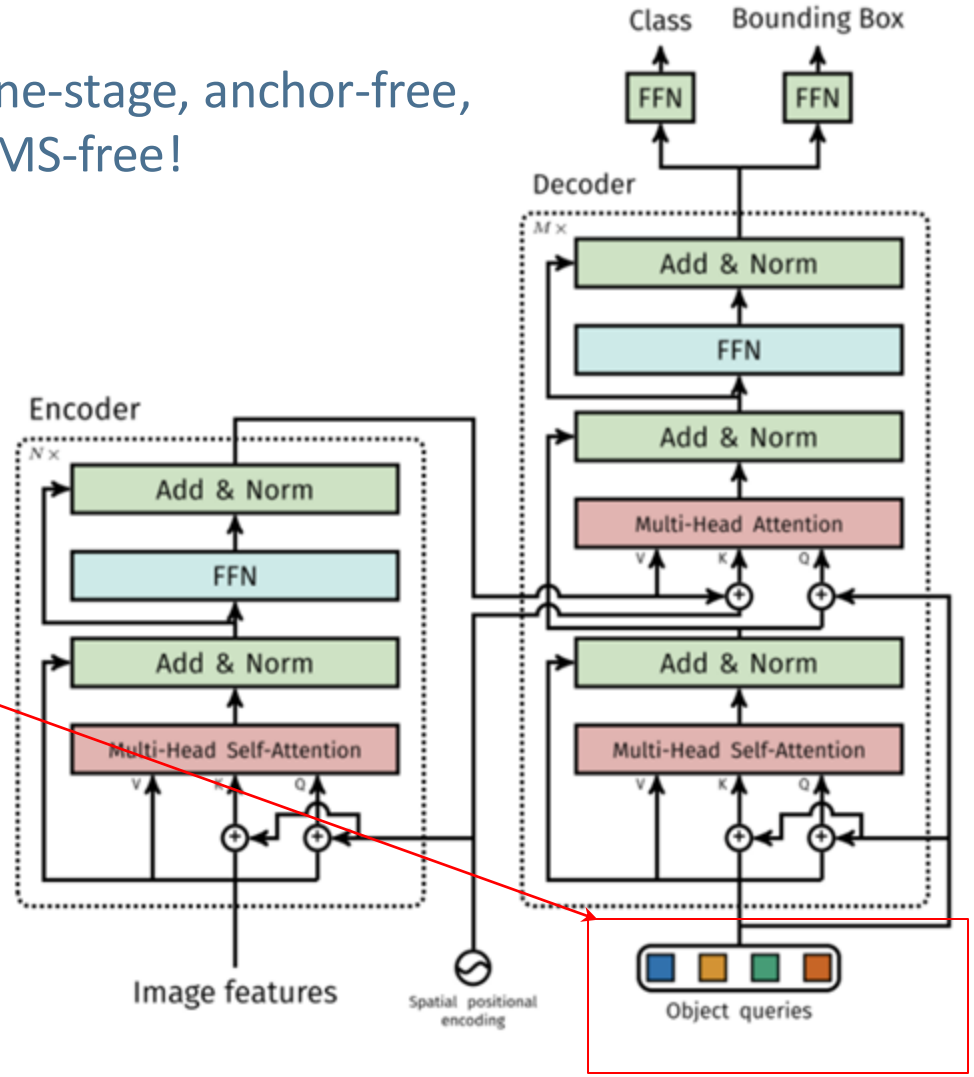


Object Detection - Transformer

DETR: End-to-End Object Detection with Transformers

- The DEtection TRansformer (DETR) predicts all objects at once, and is trained end-to-end and **abandons the anchor boxes** and **non-maximum suppression** during testing
- DETR predicts a **fixed-size set of N predictions in a single pass**. N is chosen to be significantly larger than the typical number of objects (100 queries, etc.)
- During training, the predictions are online assigned to different ground truth boxes with a bipartite matching algorithm (Hungarian)

One-stage, anchor-free, NMS-free!

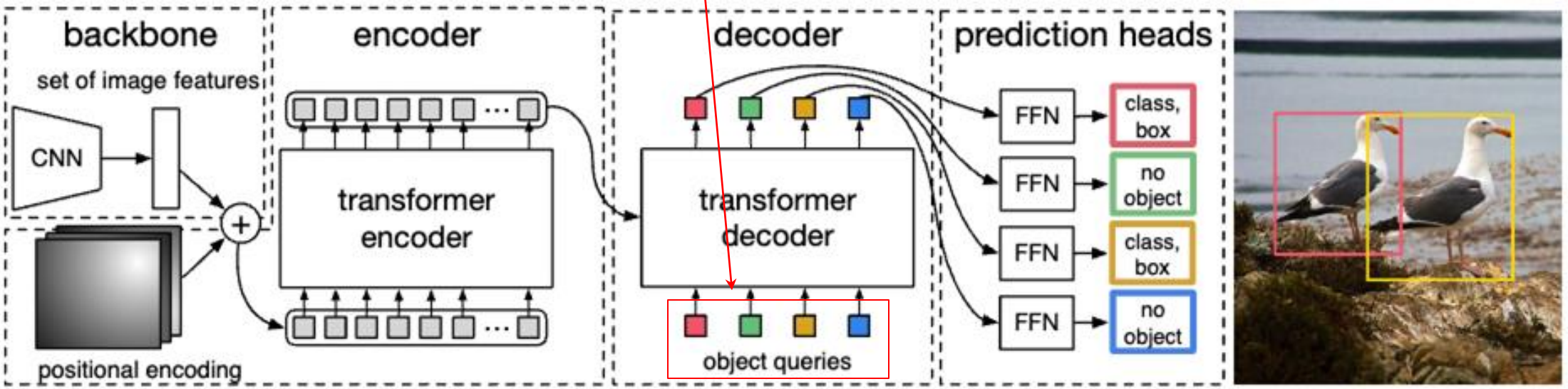


Object Detection - Transformer

- **Backbone.** Given an input image $x_{\text{img}} \in \mathbb{R}^{H_0 \times W_0 \times 3}$, an CNN backbone obtains a visual feature map $z_0 \in \mathbb{R}^{H \times W \times C}$ (H = H0/32, W = W0/32 in the original paper), which is reshaped to HW × C feature sequence

32x downsampled

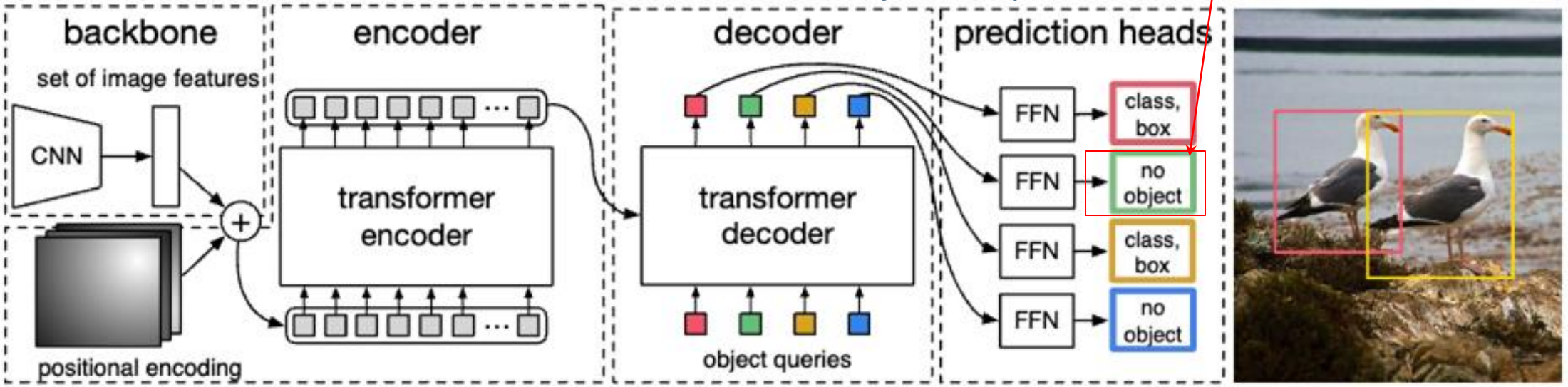
- **Transformer encoder.** The HW × C features with additive positional encoding are fed into the Transformer Encoder to conduct self-attention for multiple times
- **Transformer decoder.** It takes **N object query vectors** as inputs and decodes them **in parallel.** The query vectors are initialized randomly and learned by back-propagation



Object Detection - Transformer

- **Prediction FFN.** The head predicts the normalized center coordinates, height and width of the box w.r.t. the input image, and the class confidences
- As N is greater than the actual number of objects, an additional **background (Ø) class** is also introduced

So the fixed 100 object queries are learned "object templates"



Object Detection

- Recent improvement
  - Better training, reparametrization
    - Yolov7
  - Object detection without box representation (no squares [x, y, w, h])
    - Points
      - CenterNet: Keypoint Triplets for Object Detection. 2019
    - Polar coordinates
      - PolarMask: Single Shot Instance Segmentation with Polar Representation. 2019
  - Use transformers
    - DETR: End-to-End Object Detection with Transformers. 2020
    - DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. 2022
  - Open-vocabulary! With CLIP and stable diffusion model
    - Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. CVPR 2023

Object Tracking

- Problem
  - Tracking the same objects/features across multiple video frames
- Types
  - **Tracking without detection**
    - Manually given a bounding box, then track the box
    - Single Object Tracking (SOT, a.k.a. VOT)
      - Traditional vision method based and deep learning based (SiamRPN, CVPR 2019)
      - Not very useful (See DJI active track)
  - **Tracking-by-detection (our focus)**
    - Multi-Object Tracking (MOT)
      - Online tracking & offline tracking
      - Single camera
      - Multi-camera

- **Transition from Detection to Tracking**
  - Detection gives you (bbox, class) per frame.
  - Tracking answers: "Is this object in frame t the same as that object in frame t+1?" -> Assigns an ID.

Object Tracking

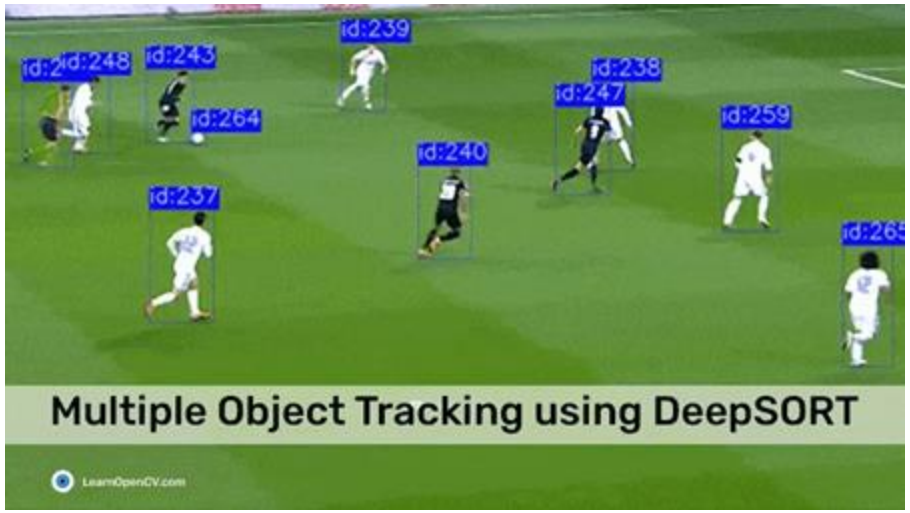
Single Object Tracking (SOT) Examples

Single object tracking is used in real-world application such as DJI Active Track / Quick shot (Click a point to track)

Object Tracking

Multi-Object Tracking (MOT) Examples

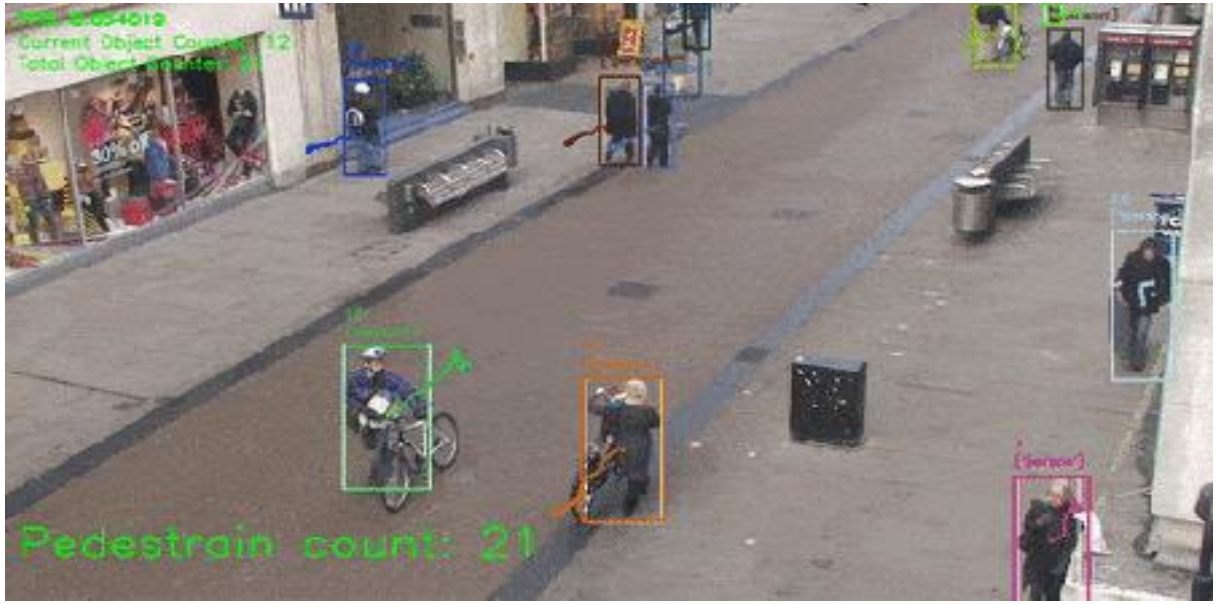
- Track players in soccer
- The statistics of each player can then be automatically summarized.
- For example, the distance covered.

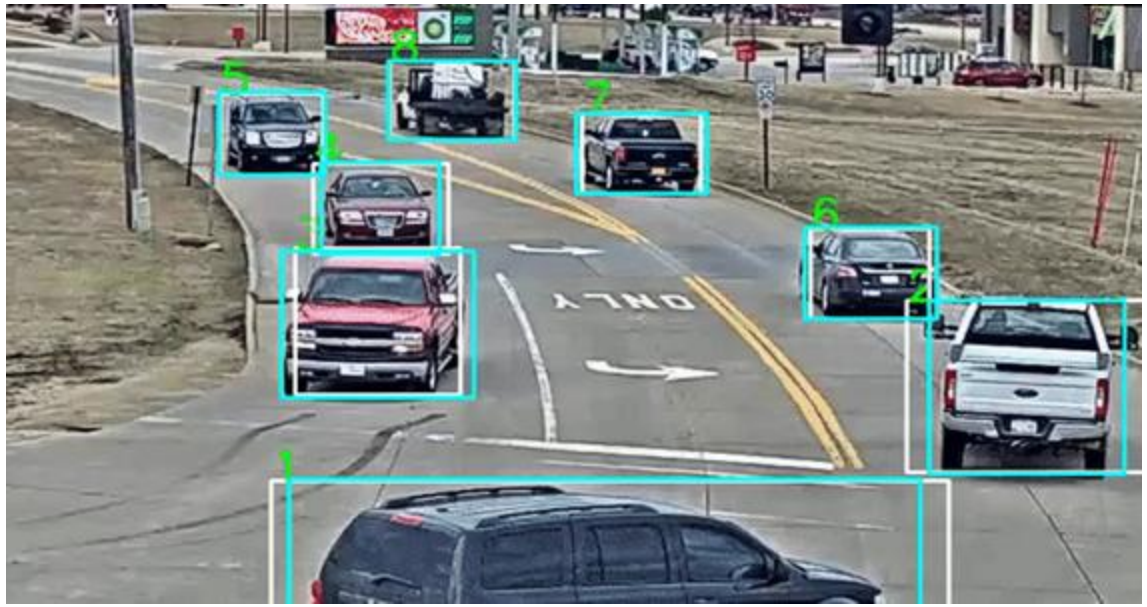


Object Tracking

Multi-Object Tracking (MOT) Examples

- Person counting from a camera
- Vehicle tracking (Tracking is widely used in traffic cameras.Compared to person tracking, vehicles are easier and with less diverse background.)





Object Detection - Tracking-by-Detection

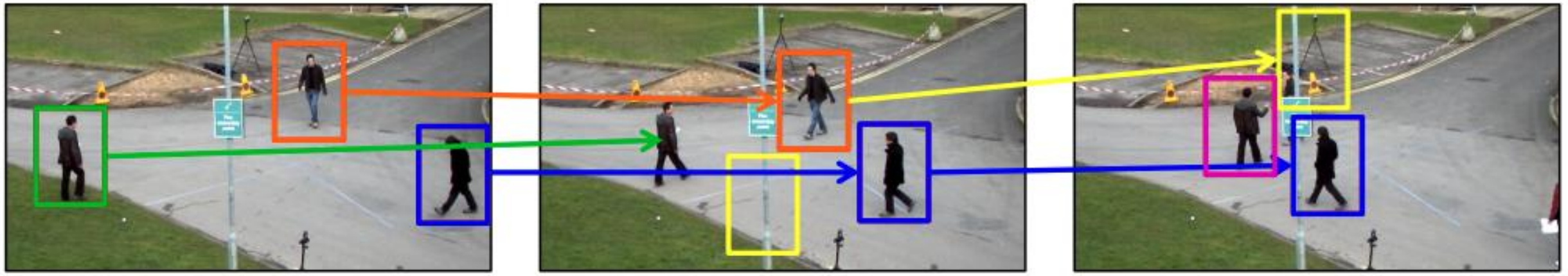
- Challenges
  - Variations due to geometric changes (pose, scale)
  - Variations due to photometric factors (change of lighting conditions, appearance)
  - Occlusions
  - Non-linear motion
  - Limited resolution, blurry, far away (so even detections might be missing)
  - Appearance is often very similar

Object Tracking - Tracking-by-Detection

- Problem
  - Input: object boxes
  - Objective: estimate **object state** over time
    - **State**: box position. This is a naming convention from robotics
- Key ingredients
  - Detection
  - Tracking
    - **State estimation**
      - predict where the boxes might be
    - **Data association**
      - Similarity measurement

Object Tracking - Tracking-by-Detection

- Problem
  - Input: target boxes
  - Objective: estimate target state over time (given the a bunch of detected boxes across time)
  - State: box position
- First thing to note
  Detections are not perfect!



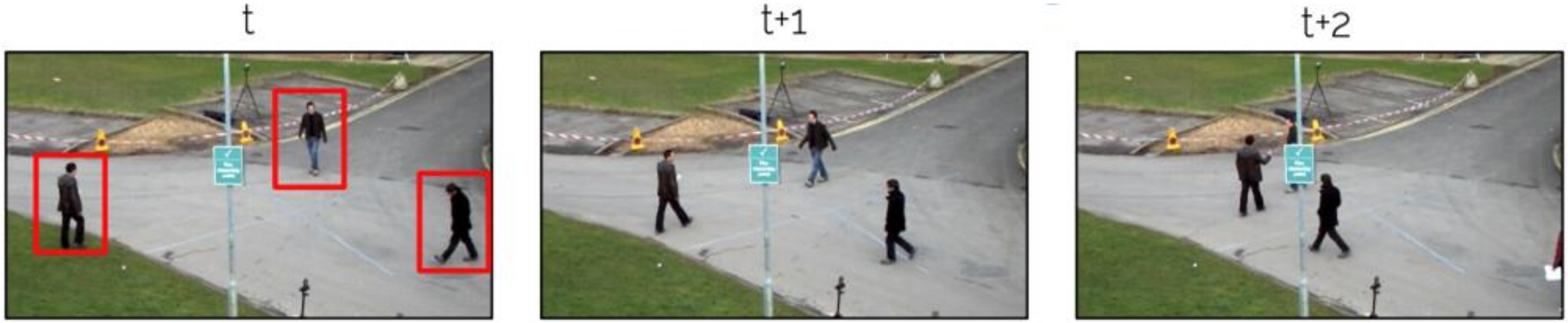
Find detections that match and form a trajectory

Online vs Offline Tracking

- Online tracking
  - Processes two frames at a time
  - For real-time applications
  - Prone to drifting
    - Hard to recover from errors or occlusions
- Offline tracking
  - Processes a batch of frames
  - Good to recover from occlusions
  - Not suitable for real-time applications
- We will only cover online tracking method in this class
  - As this is the most used method in real-world applications like surveillance and robots

Object Tracking - Online Tracking

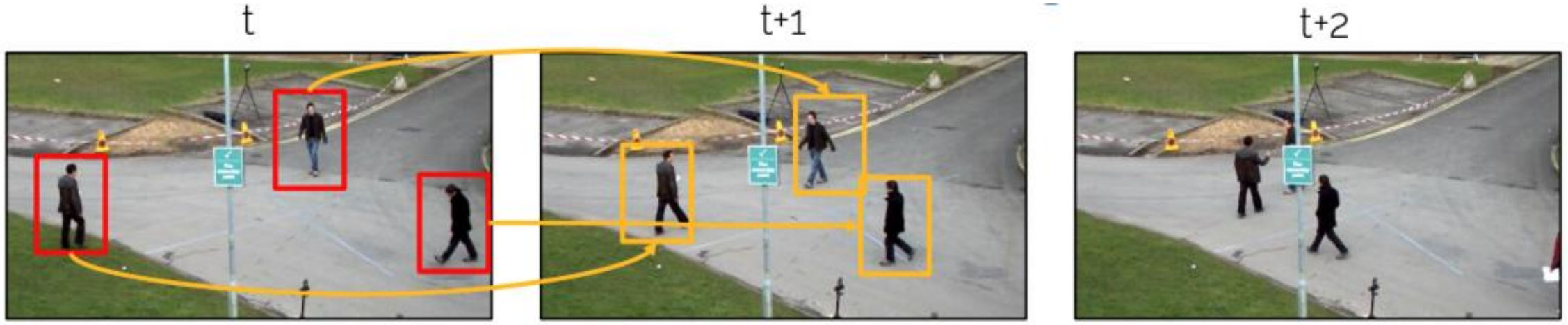
Example Steps



1. Track initialization (e.g. using an object detector)

Object Tracking - Online Tracking

Example Steps

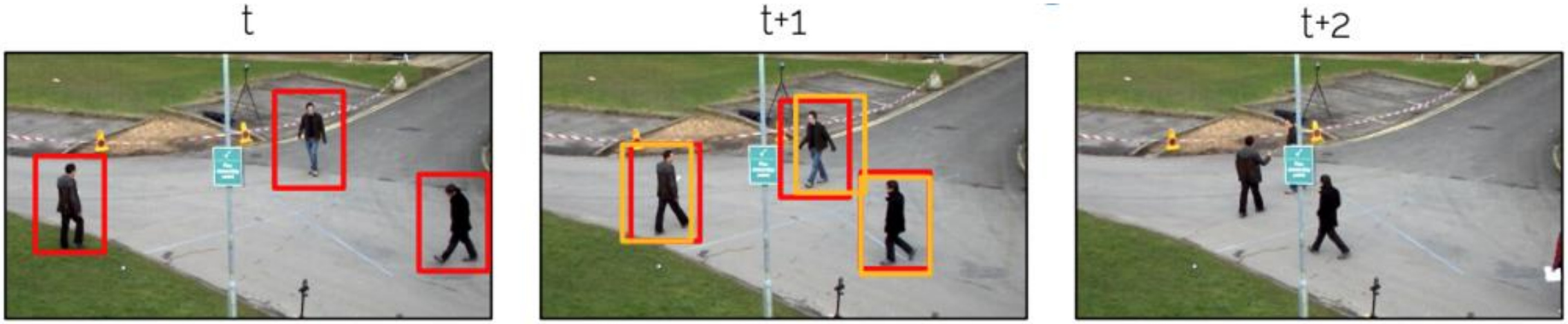


1. Track initialization (e.g. using an object detector)
2. Prediction of the next position

For example, we can predict where the track's box will be by taking into its previous moving trajectory

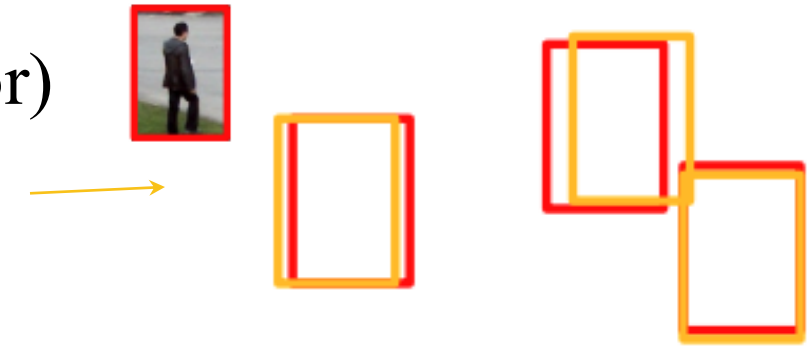
Object Tracking - Online Tracking

Example Steps



1. Track initialization (e.g. using an object detector)
2. Prediction of the next position
3. Matching predictions with detections

   Data association based on appearance, location, etc.



Object Tracking

Prediction of the next position

State and Kalman filter, are common terms in robotics. We will see "state" again in later lectures

- **Kalman filter**
  - **State** space of a object box xy $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$
    - 4 dim box coordinates
    - 4 dim their respective velocity
    - The Kalman filter for each track will have 8 scalars for the mean of these variables, and a 8x8 matrix of their covariance (which means it assumes a multivariate gaussians)

This is assumed to be changing over time

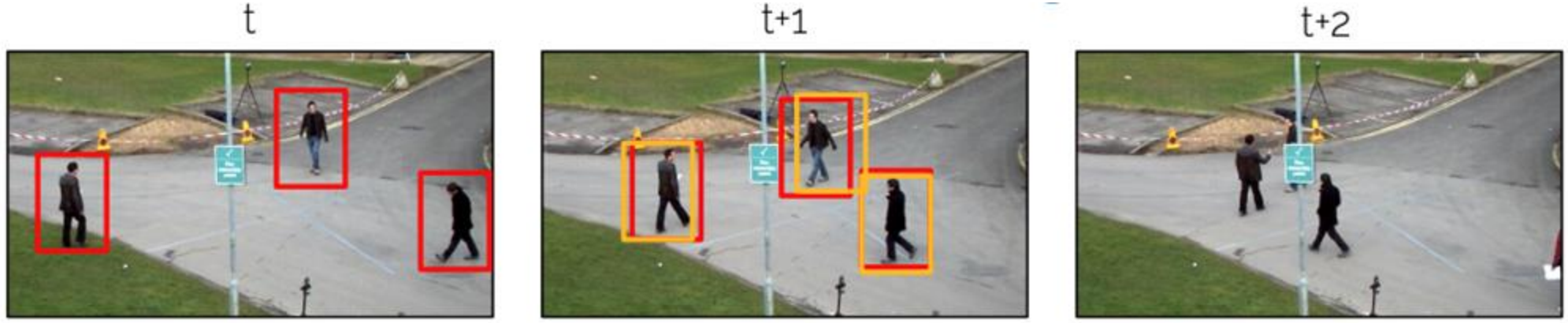
  - Given 4 dim box coordinates as a new observation of the current timestep
    - Update the track's current KF states (8 mean and 8x8 covar)
      - Based on previous states and **some calculation**
        - This involves some computation of Kalman gain, you can check the code below. Not going into this detail today
    - Just consider it as **a linear model that extrapolate tracks** (object states) assuming constant velocity

Object Tracking

Comparing detected boxes and predicted boxes

- Blue box: detected boxes
- White box: predicted boxes
- See the white box slowly enlarges

Object Tracking



1. Track initialization (e.g. using a object detector)
2. Prediction of the next position (motion model)
3. **Matching predictions with detections**

   Data association based on appearance, location, etc.

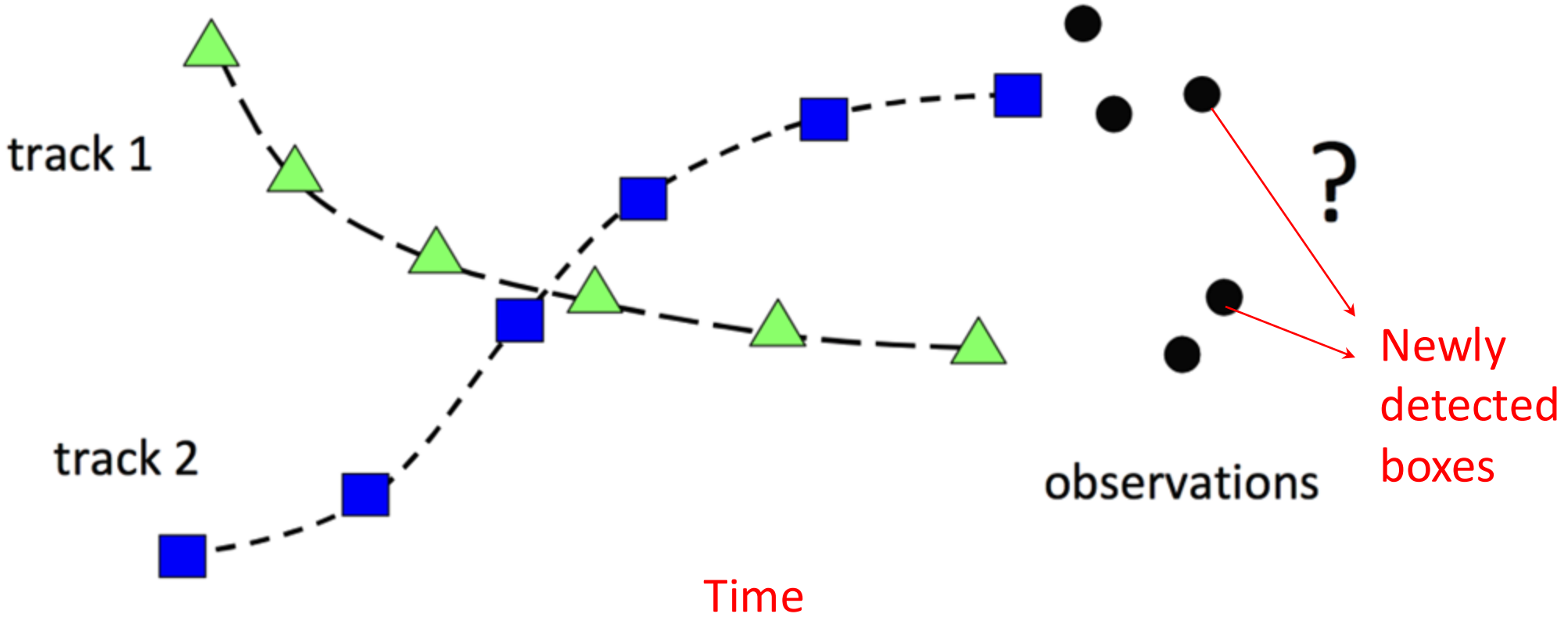
Data Association

- Given each estimated track state (potential target box locations)
  - Assign tracks with new boxes (new observations)
  - Which is a discrete combinatorial optimization problem

Object Tracking

Data Association

- Given each estimated track state (potential target box locations)

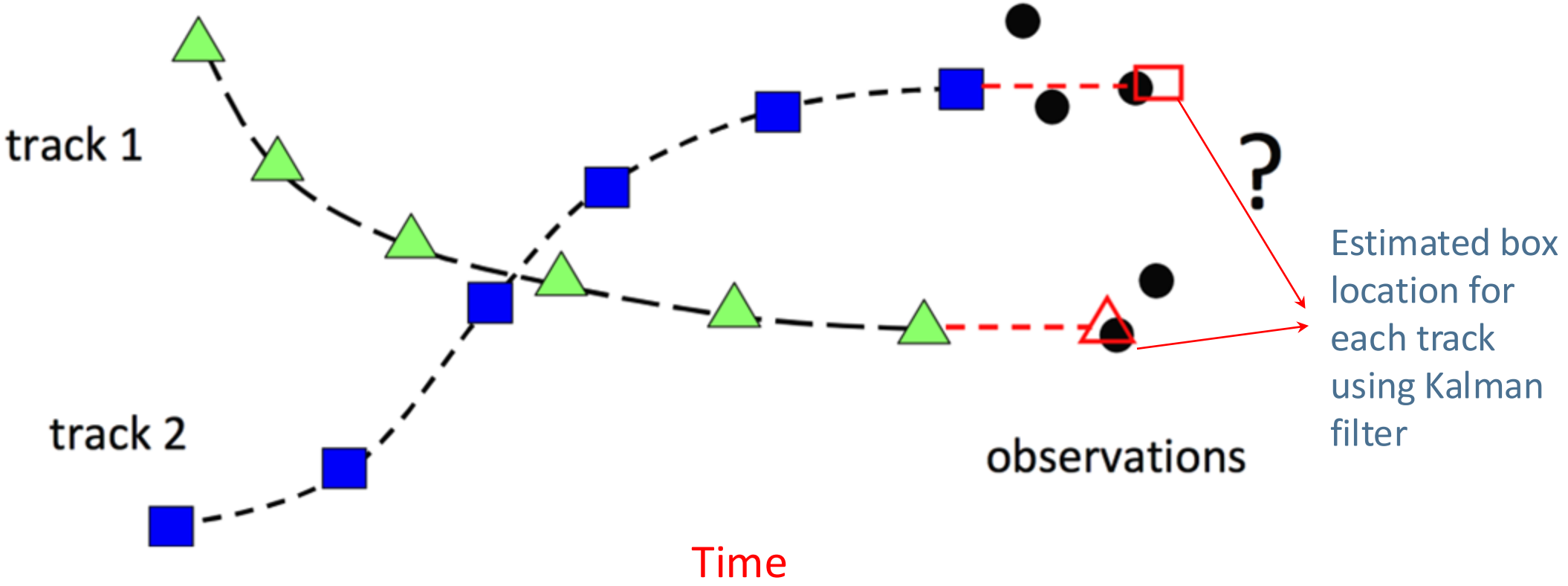


Credit: Collins, PSU

Object Tracking

Data Association

- Given each estimated track state (potential target box locations)

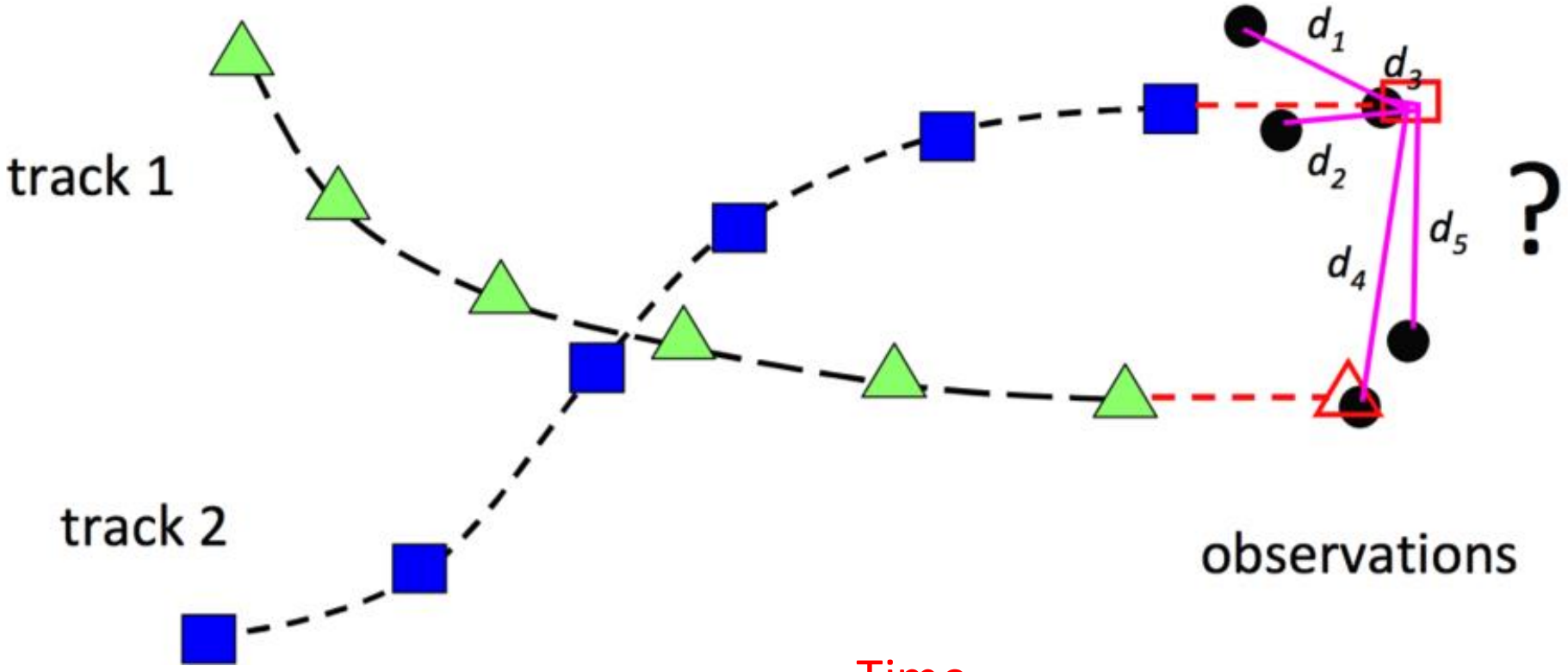


Credit: Collins, PSU

Object Tracking

Data Association

- Given each estimated track state (potential target box locations)



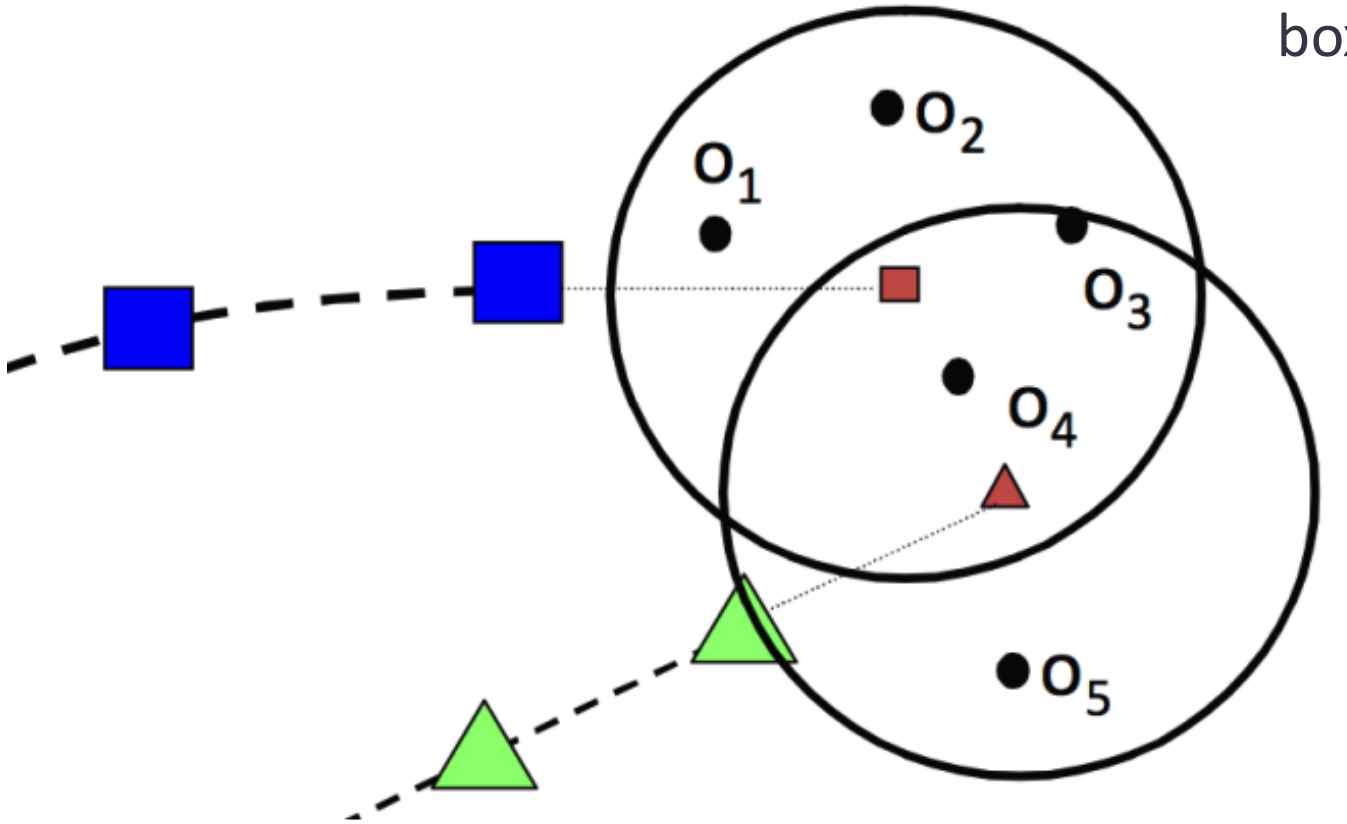
How to assign estimated box to new detected box?

Credit: Collins, PSU

Object Tracking

Data Association

- Given each estimated track state (potential target box locations)



New boxes	Track 1	Track 2
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

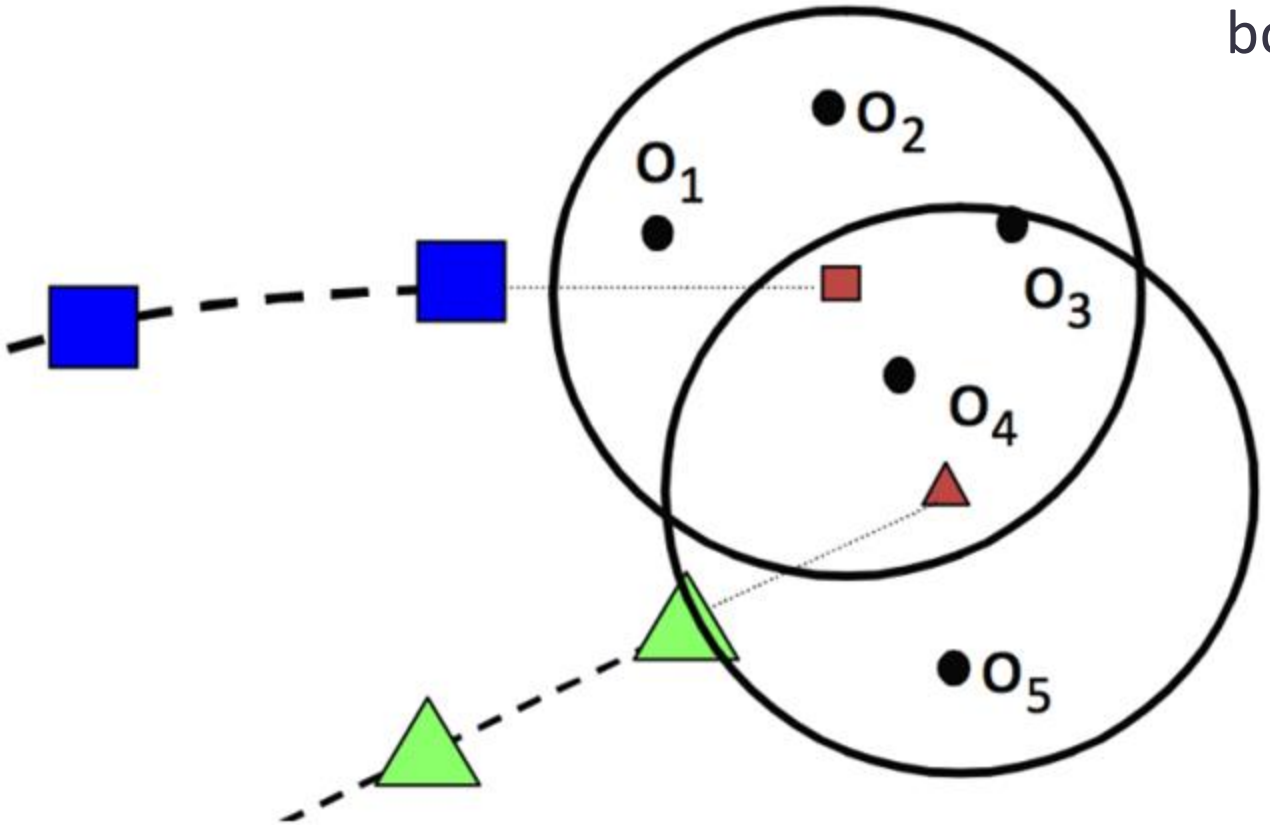
We have 5 new observed boxes

Suppose these are the matching scores (closer the higher, etc.)

Object Tracking

Data Association

- Given each estimated track state (potential target box locations)



New boxes	Track	Track
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

We can only do 1-to-1 assignment.

Using greedy method, Track 2 will be assigned to box 5

Non-optimal!

Total score is not maximized

Object Tracking

Data Association

- Assignment problem mathematical definition

Suppose we want to match n items to n(m) items
X is the assignment matrix of shape n-by-n(m)
W is the similarity matrix of shape n-by-n(m)
Maximize the total similarity score (weighted sum)

$$\text{maximize:} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij}$$

$$\text{subject to:} \quad \begin{aligned} &\textstyle\sum_j x_{ij} = 1; \quad i = 1,2,\ldots,n \\ &\textstyle\sum_i x_{ij} = 1; \quad j = 1,2,\ldots,n \\ &x_{ij} \in \{0,1\} \end{aligned}$$

constraints that say
X is a permutation matrix

So only 1 entry per row

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Object Tracking

Data Association

- **Hungarian Algorithm**
  - For assignment problem, also called **weighted bipartite matching problem**
    - Bipartite graph

      Has two sets of **nodes** S, T;

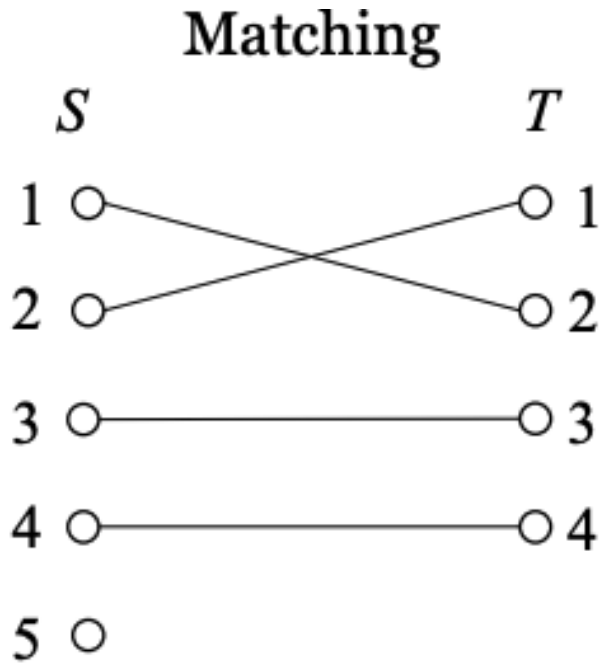
      V = union(S, T)

      And a set of **edges** E connecting them

      A matching on a bipartite graph is to find E that no two edges are connected to the same node

      So some nodes can be left out
  - Complexity - **<u>guaranteed optimal solution</u>**
    - O(n^3), where n is the number of elements in each set.

We skip the algorithm detail. You just need to know what it is for and when to use it. Tools in scipy compute for you.



Object Tracking

Data Association

- Example result - matching 5 nodes with other 5 nodes

Given the similarity matrix

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Greedy Solution

Score=3.77

Go from row 1 to row 5, select the one with the highest score

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

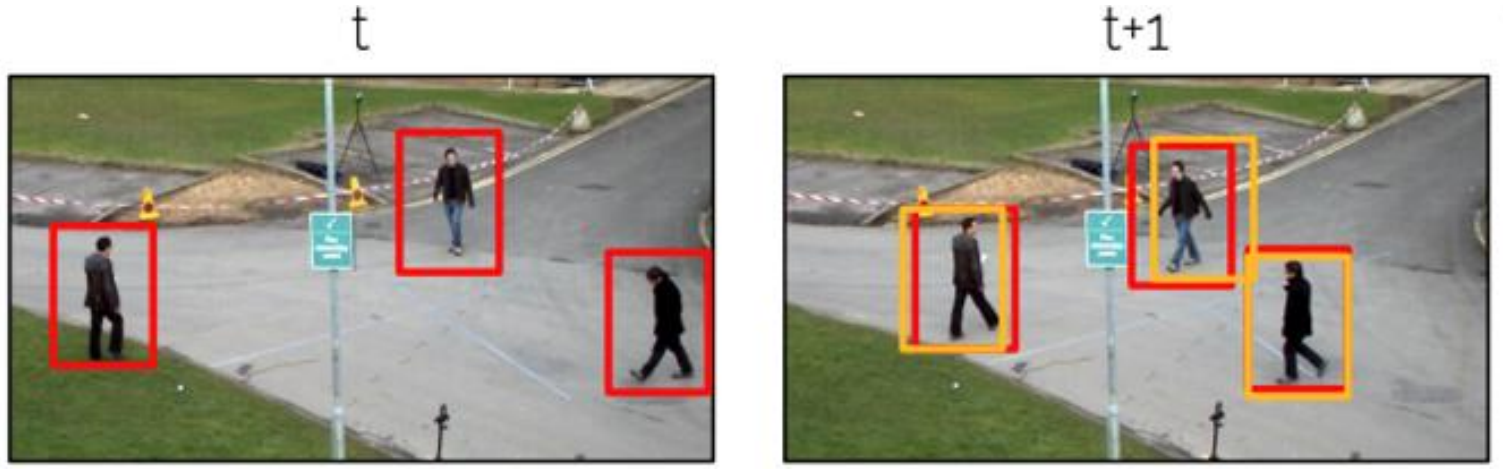
Optimal Solution

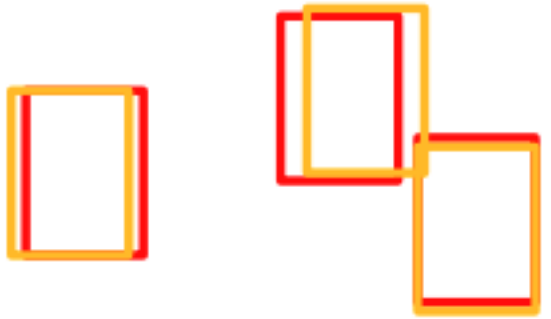
Score=4.26

Hungarian algorithm (An iterative process that adjusts the matching)
Avoids a bad match

Object Tracking - Online Tracking

How is the cost matrix computed?



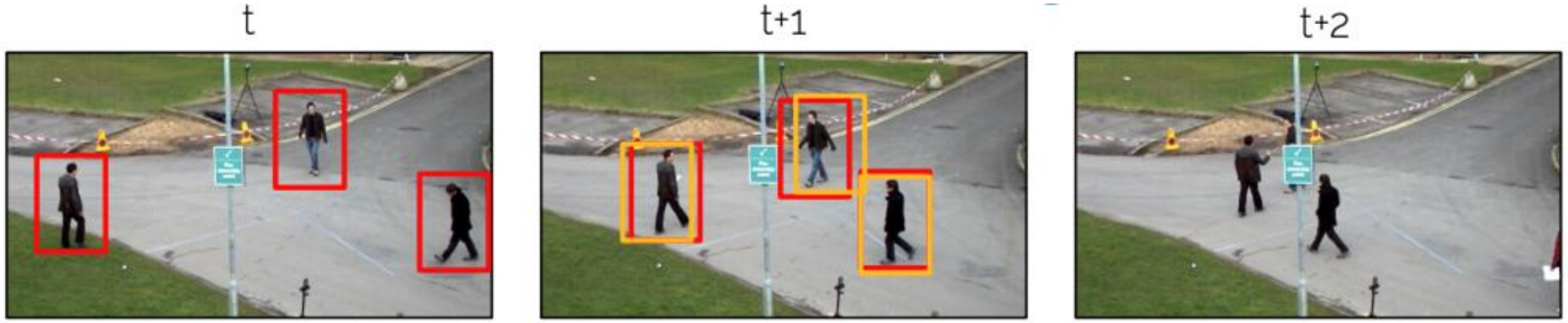


- 3. Matching predictions with detections
  - Data association based on appearance, **location**, etc.

Compute IOU between predicted boxes and observed boxes

Object Tracking

How is the cost matrix computed?

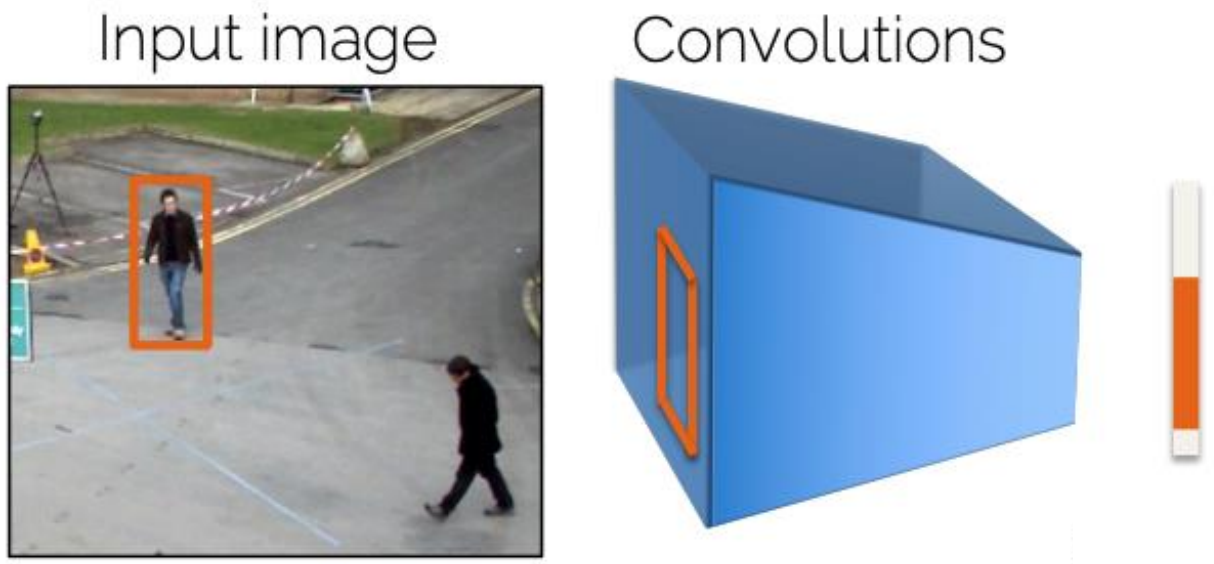


- 3. Matching predictions with detections
  - Data association based on **appearance**, location, etc.

Object Tracking

Appearance Feature for matching

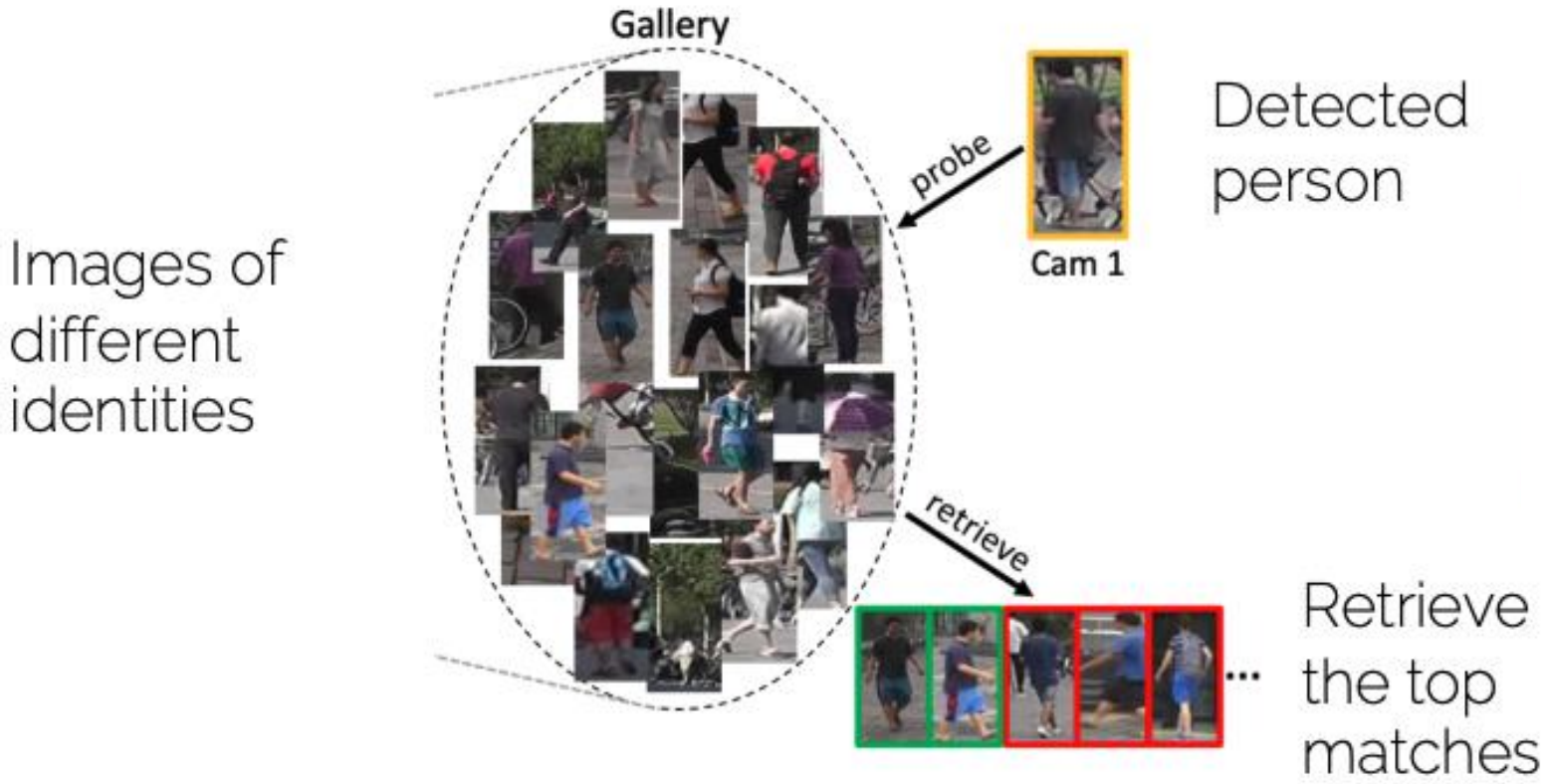
- Matching predictions with detections
  - Data association based on **appearance**, location, etc.
  - We can use **ROIAlign** to get feature vector of the object from object detection model - Save computation!
    - Then we can compute cosine distance between two vectors



Object Tracking

Better Appearance Feature - Person Re-Identification

- Use **metric learning/similarity learning/contrastive learning** to get better discriminative person features
  - Person Re-ID is an active area of research for many years



Object Tracking - Online Tracking

Summary

- 1. Track initialization (e.g. using a object detector)
  - YOLO, Faster RCNN, etc.
- 2. Prediction of the next position
  - Kalman filter
- 3. Matching predictions with detections
  - Data association based on appearance, location, etc.

Upgrade your detector and you will get a significant boost in performance

Super fast

Hungarian Algorithm

The SORT method: Simple Online and Realtime Tracking, 2016

The DeepSORT method: Simple Online and Realtime Tracking with a Deep Association Metric, 2017

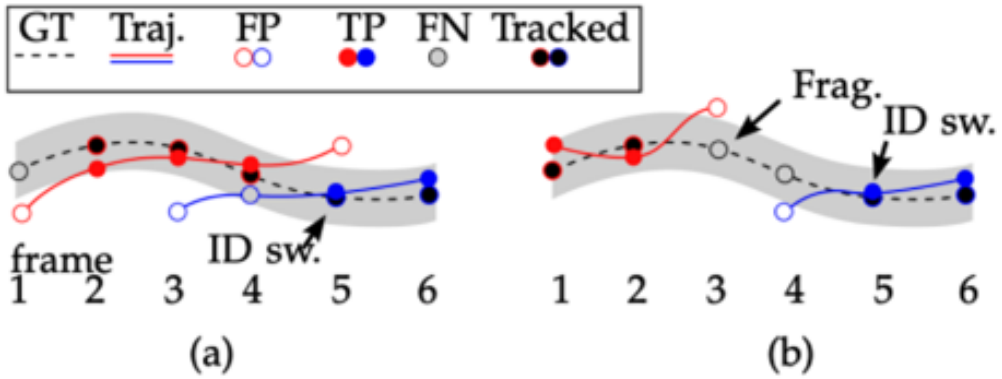
MOT - Evaluation

- Compute a set of measures per frame
  - Perform matching between predictions and ground truth (use exactly the same Hungarian algorithm)
  - FP = False positives
  - FN = False negatives (missing detections)
  - **IDsw: identity switches**

The first two are the same as object detection evaluation

Object Tracking

MOT - Evaluation



- How do we compute ID switches? Here are two common scenarios
  - (a) An ID switch is counted because the ground truth track **is assigned first to red, then to blue**
  - (b) You count an ID switch (red and blue both assigned to the same ground truth), but also a fragmentation (Frag./False Negative) because the ground truth coverage was cut

MOT - Evaluation - MOTA

- Compute a set of measures per frame
  - Perform matching between predictions and ground truth (we will use exactly the same Hungarian algorithm)
  - FP = False positives
  - FN = False negatives (missing detections)
  - IDsw: identity switches

Multi-object tracking accuracy

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t},$$

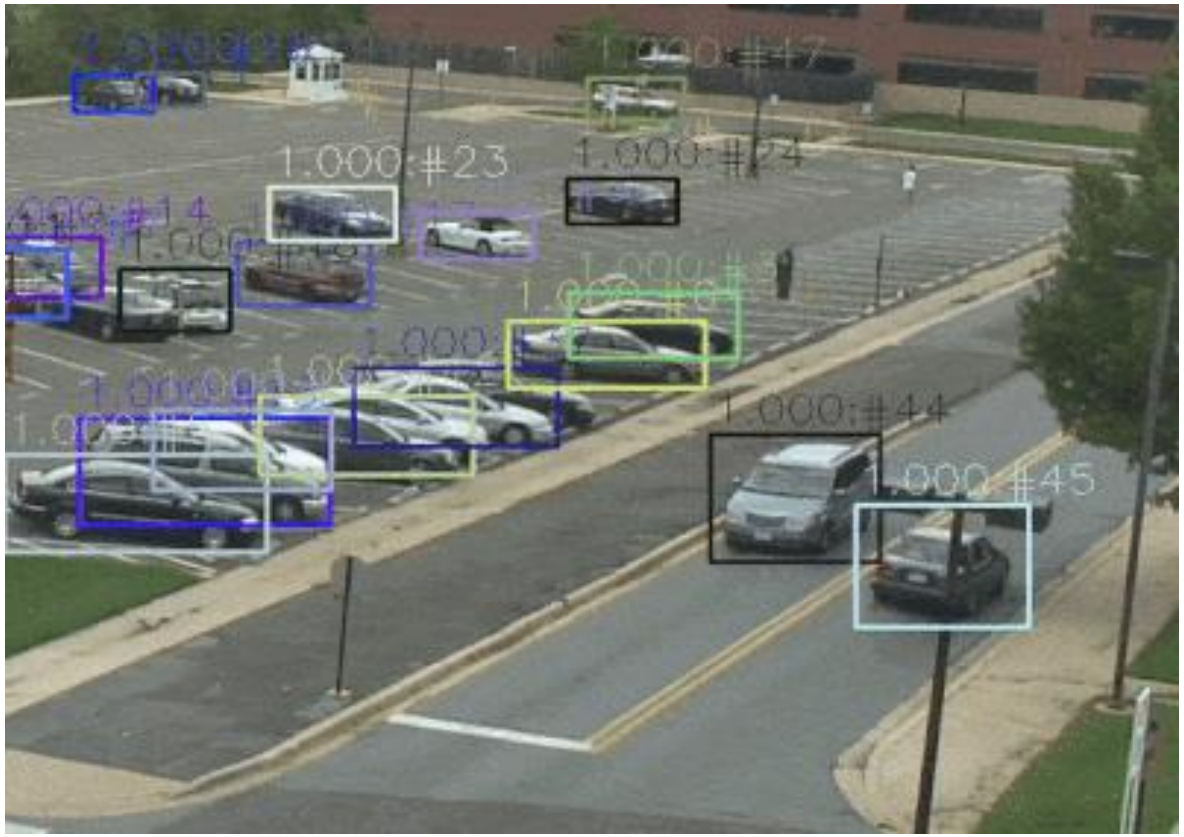
Ground truth

One minus the mistakes the tracker make.
So higher the better

Object Tracking

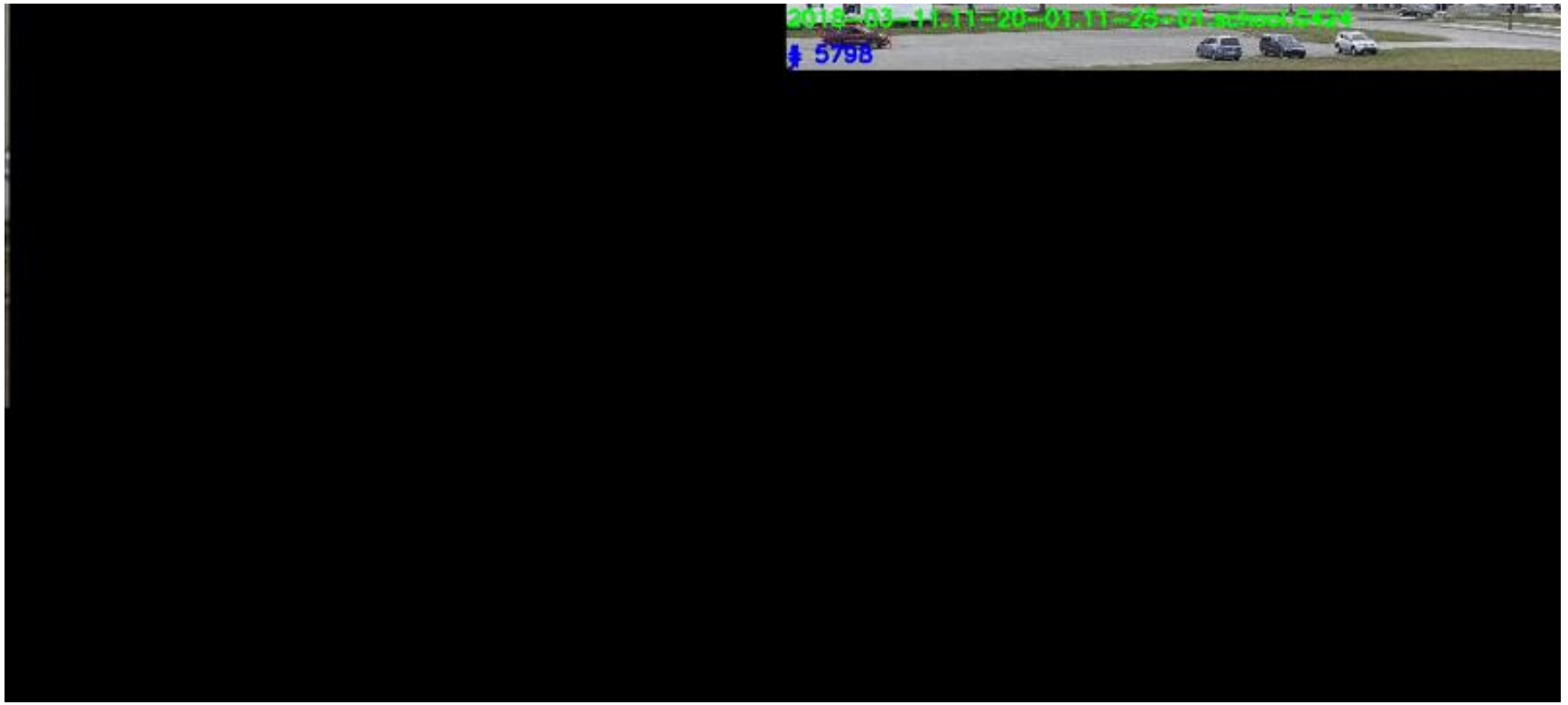
Tracking-by-Detection - Single Camera MOT

- Example
  - Motion
  - Change in scale
  - Occlusion
  - See the missing few boxes of the parked cars but the ID stays the same



Tracking-by-Detection - Multi-Camera MOT

- Example
  - Need to perform ReID on multiple cameras



Object Tracking

Long-term MOT with trajectory prediction

We can combine trajectory prediction with MOT…

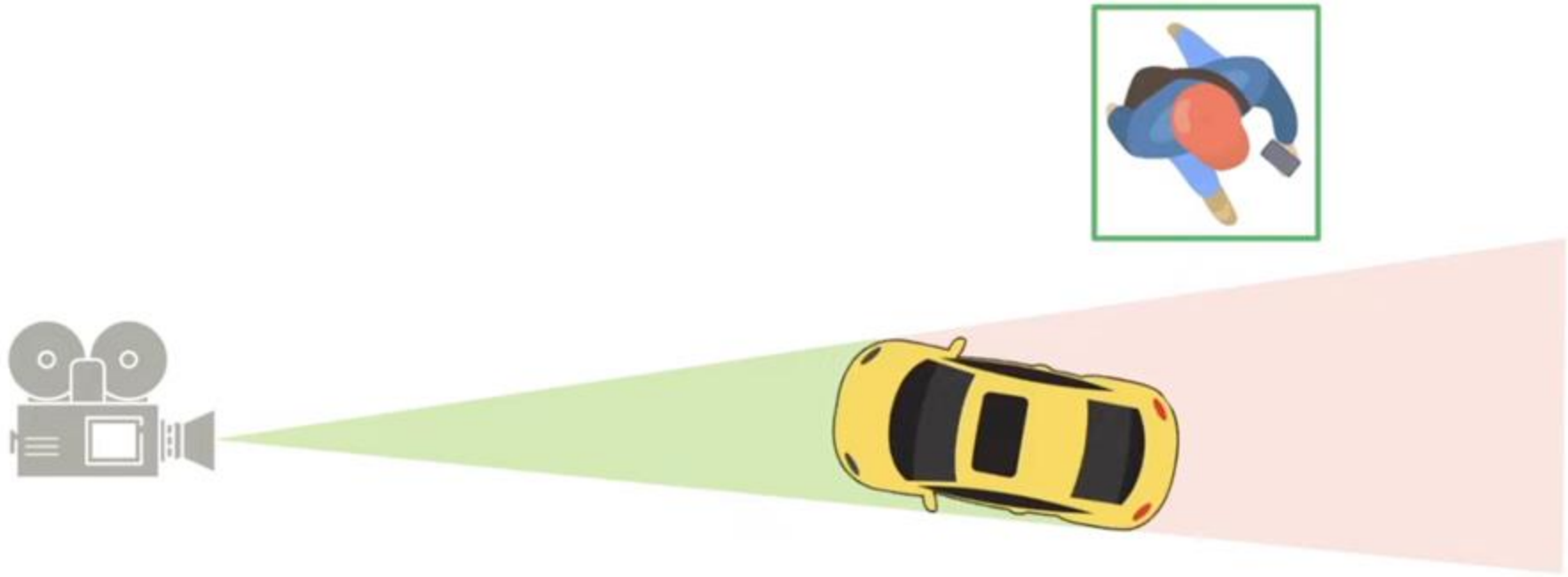
- Multi-object tracking suffers from long-term occlusions
  - Recall MOT

Quo Vadis: Is Trajectory Forecasting the Key Towards Long-Term Multi-Object Tracking? NeurIPS 2022

Object Tracking

Long-term MOT with trajectory prediction

- Multi-object tracking suffers from long-term occlusions

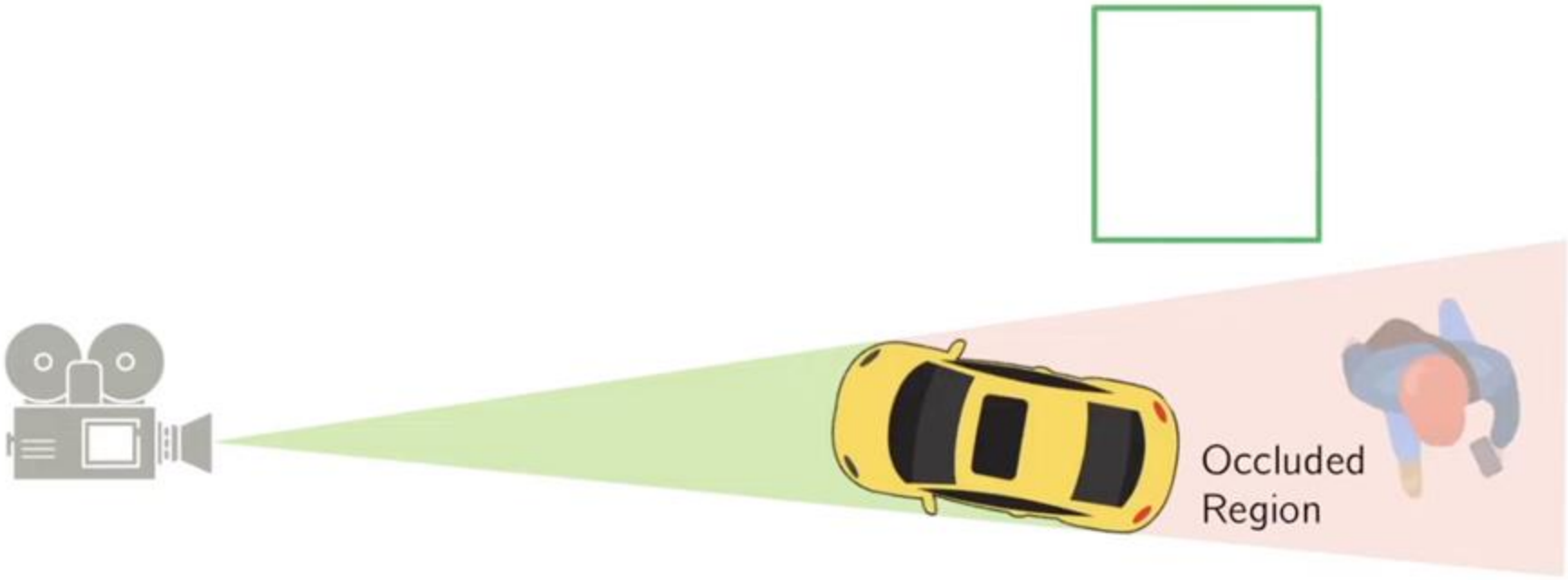


Quo Vadis: Is Trajectory Forecasting the Key Towards Long-Term Multi-Object Tracking? NeurIPS 2022

Object Tracking

Long-term MOT with trajectory prediction

- Multi-object tracking suffers from long-term occlusions

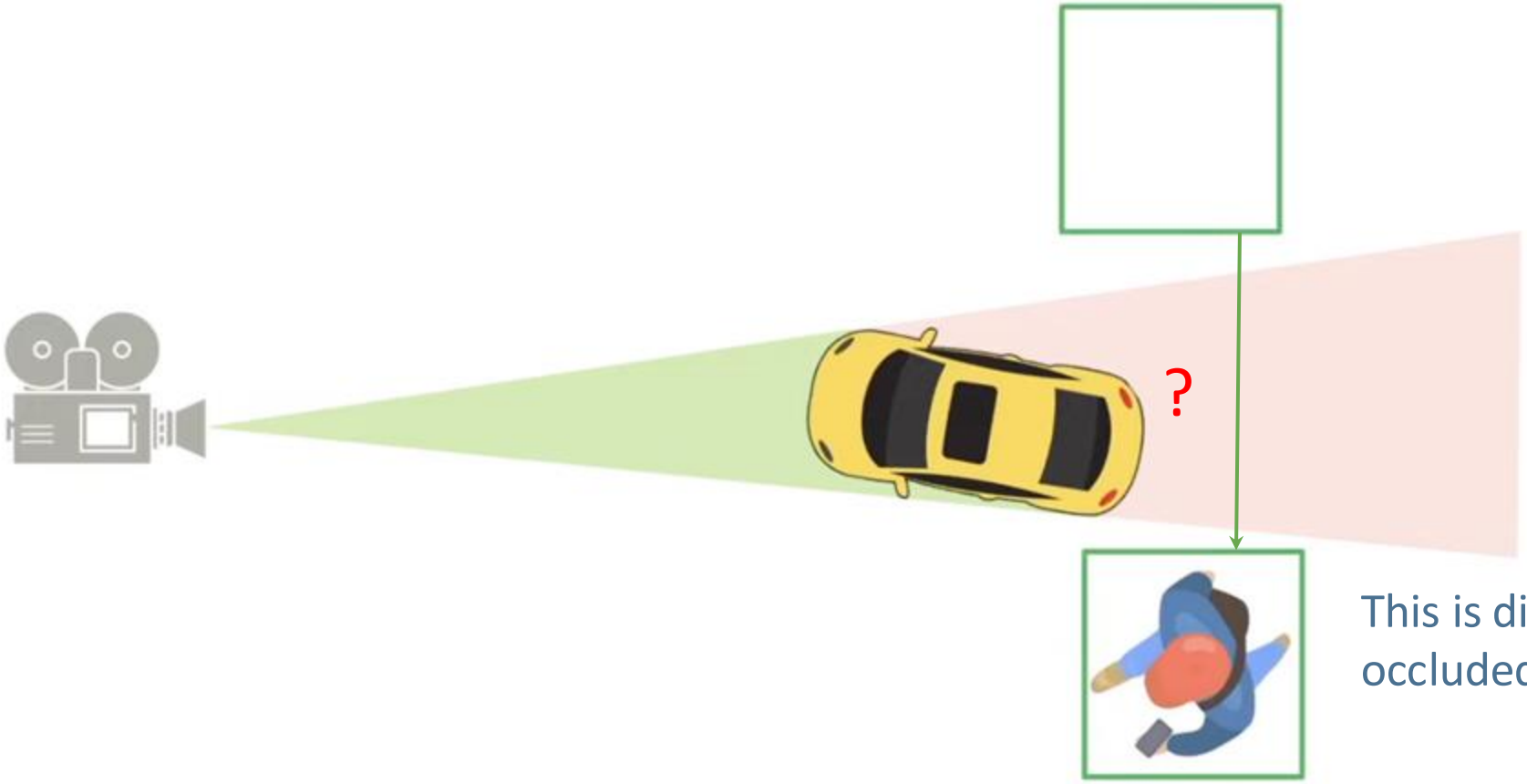


Quo Vadis: Is Trajectory Forecasting the Key Towards Long-Term Multi-Object Tracking? NeurIPS 2022

Object Tracking

Long-term MOT with trajectory prediction

- Multi-object tracking suffers from long-term occlusions

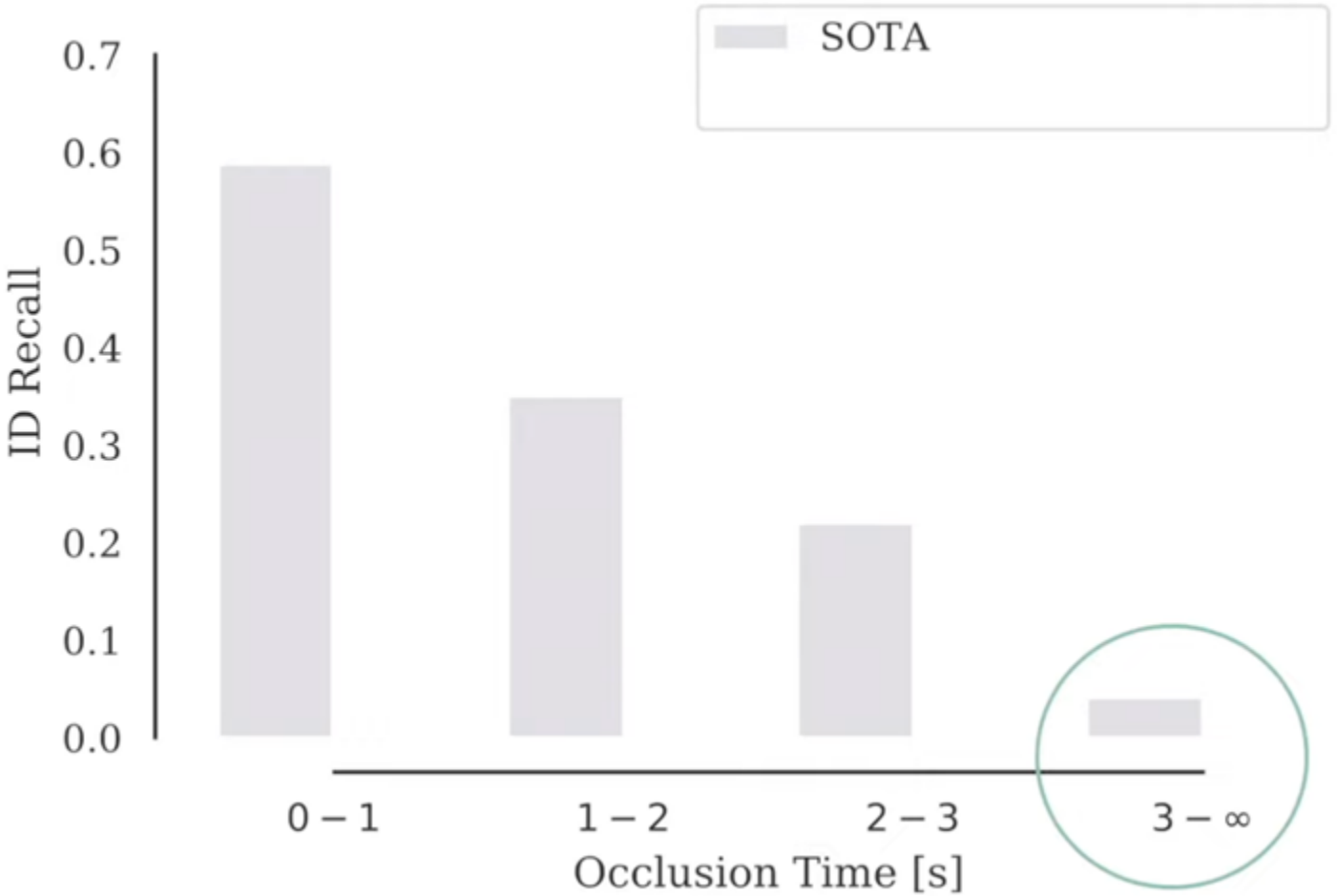


This is difficult if the occluded time is too long

Object Tracking

Long-term MOT with trajectory prediction

- Multi-object tracking suffers from long-term occlusions



Performance of top-8-trackers on MOT17 benchmark

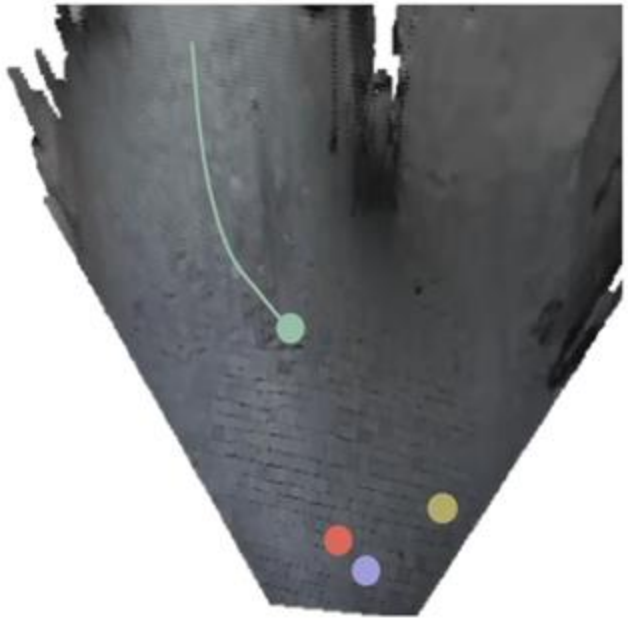
Object Tracking

Long-term MOT with trajectory prediction

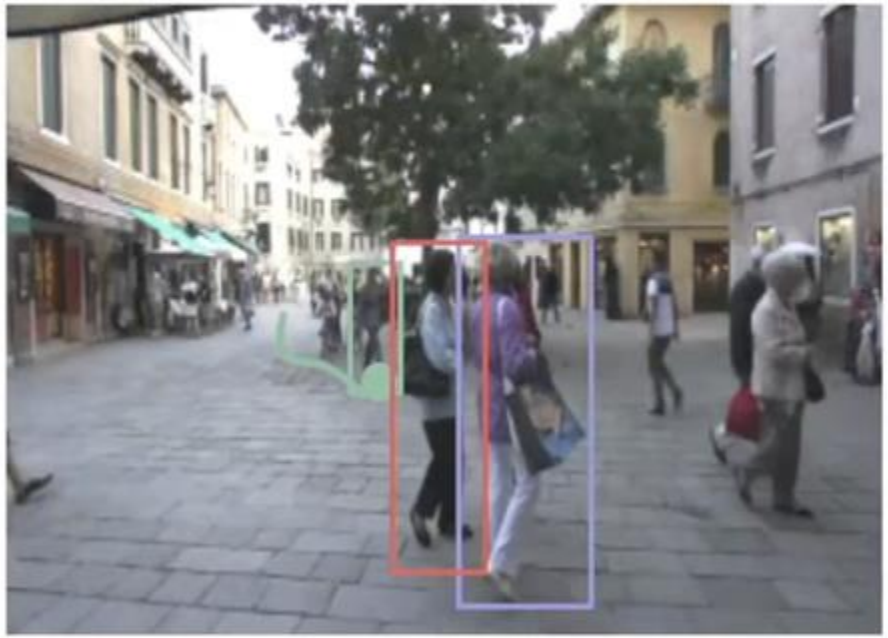
- Perform trajectory prediction in bird's view when occlusion in MOT occurs
  - Step 1. Project image and the detected person location into BEV view

This assumes everything is on the ground

Scene Bird's-Eye View



Scene Image

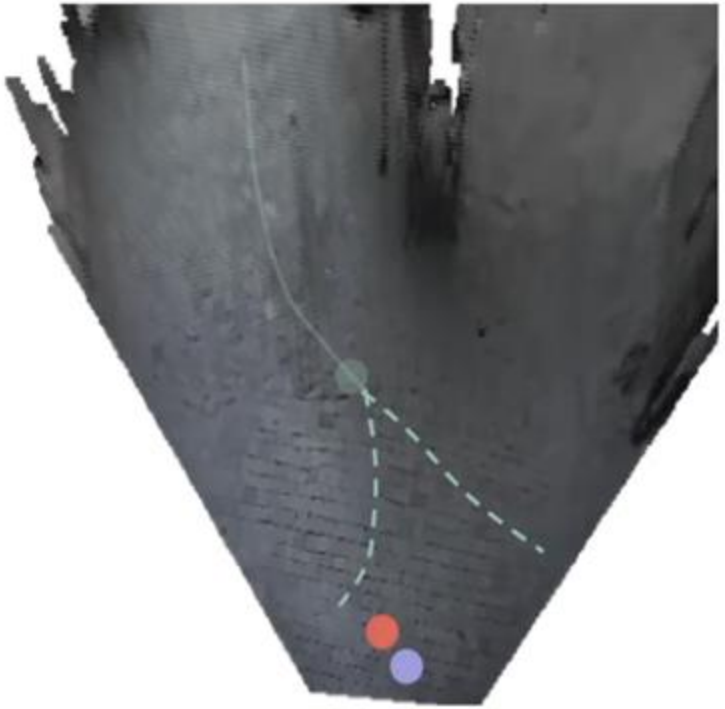


Object Tracking

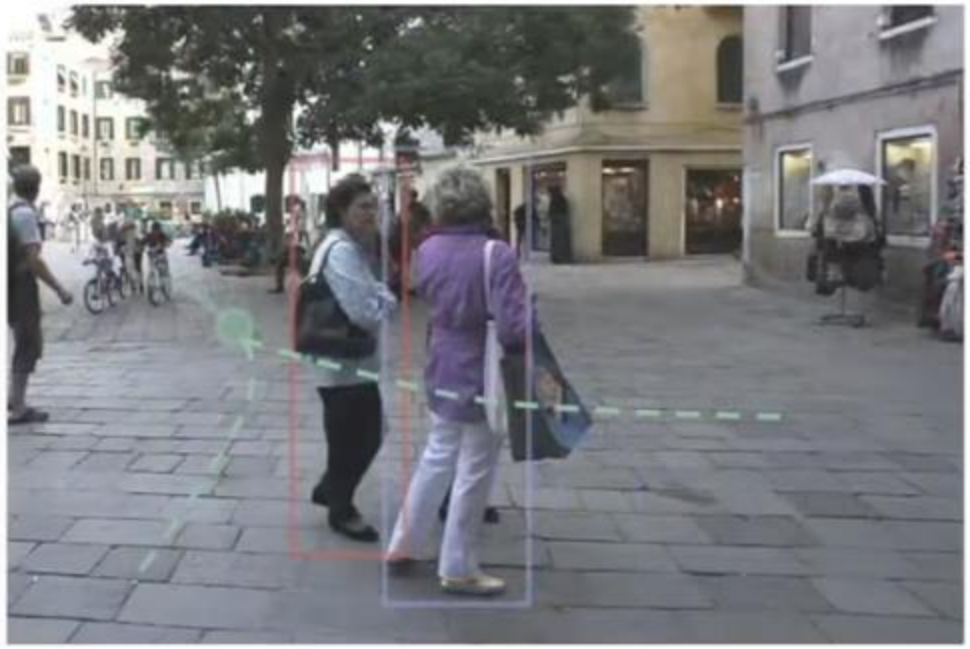
Long-term MOT with trajectory prediction

- Perform trajectory prediction in bird's view when occlusion in MOT occurs
  - Step 2. **Predict trajectory** when occlusion occurs
    - This essentially reduces the search space for re-identification of the person

Scene Bird's-Eye View



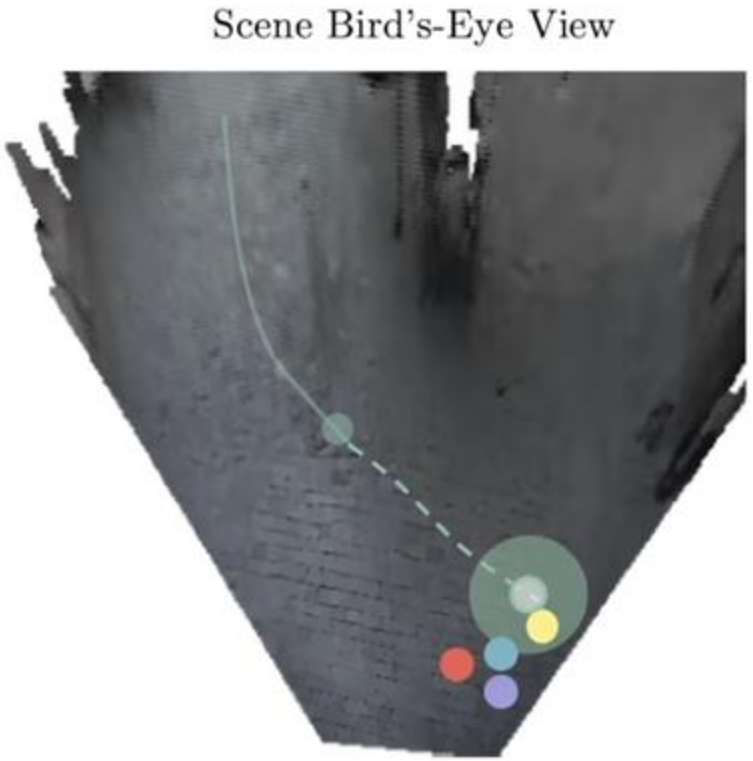
Scene Image

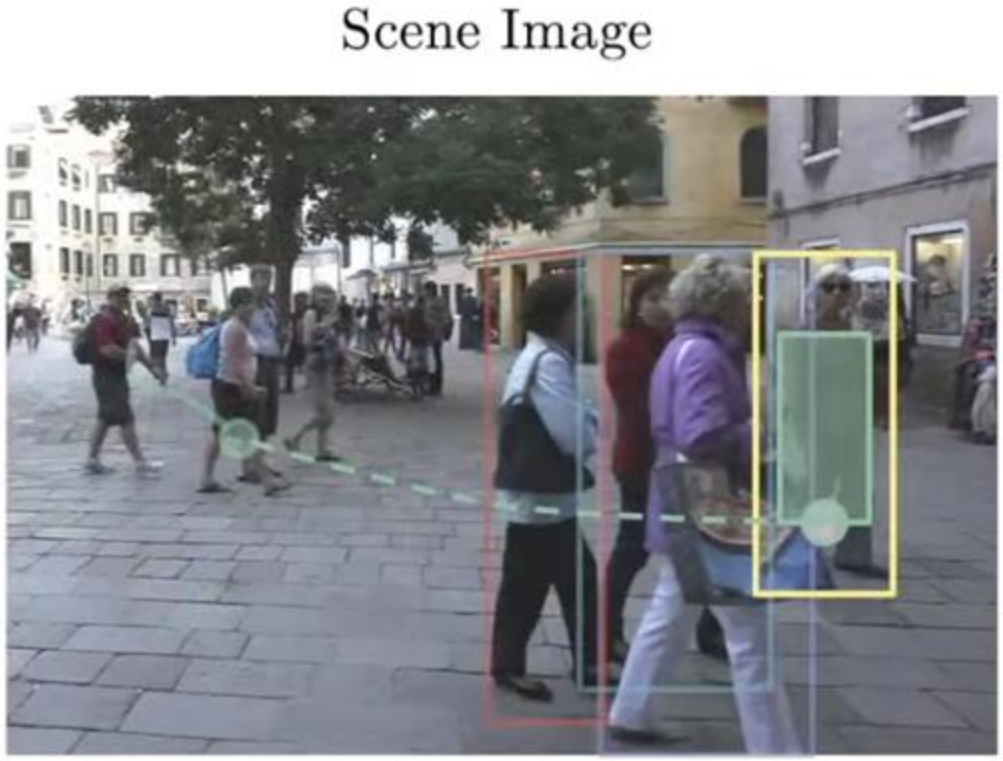


Object Tracking

Long-term MOT with trajectory prediction

- Perform trajectory prediction in bird's view when occlusion in MOT occurs
  - Finally, use appearance matching when the person appears





Object Tracking

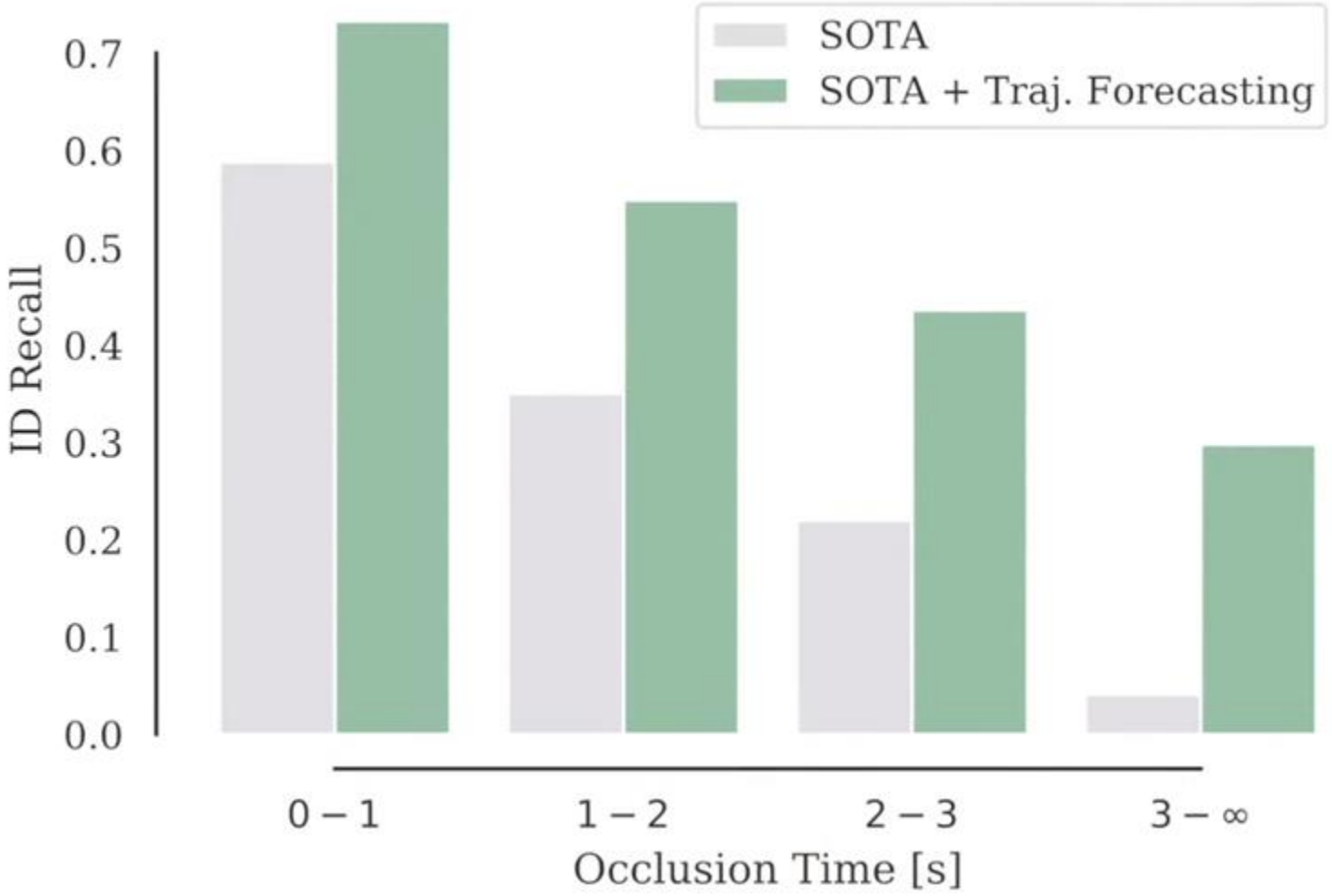
Long-term MOT with trajectory prediction

- Perform trajectory prediction in bird's view when occlusion in MOT occurs
  - Full example

Object Tracking

Long-term MOT with trajectory prediction

- Experiments



Resources

1. Papers:

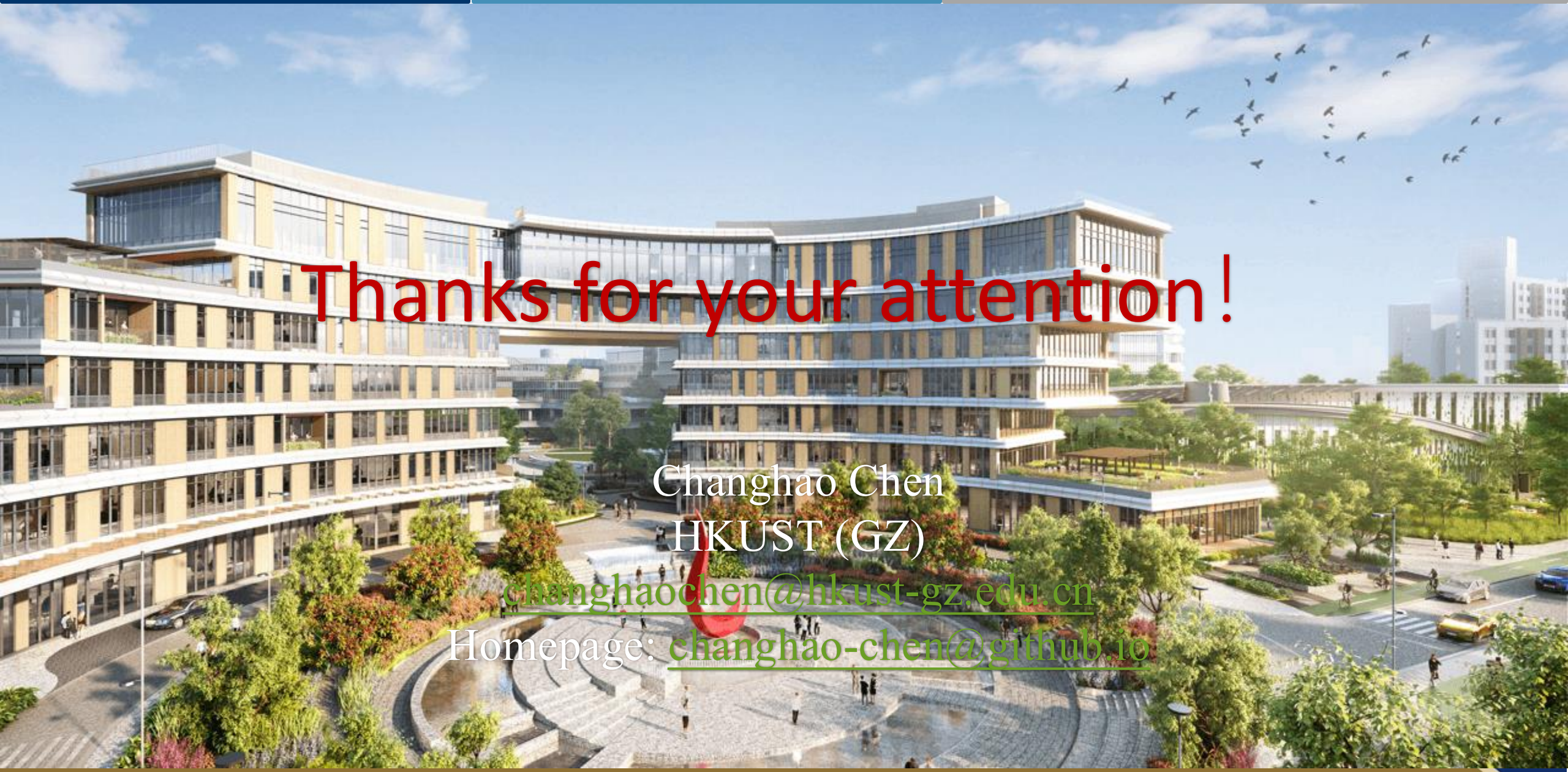
- Rich feature hierarchies for accurate object detection and semantic segmentation
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- You Only Look Once: Unified, Real-Time Object Detection
- End-to-End Object Detection with Transformers
- Simple Online and Realtime Tracking
- Simple Online and Realtime Tracking with a Deep Association Metric

2. Opensource Code:

Detectron2, YOLO

3. Benchmark Datasets:

COCO, Pascal VOC, MOTChallenge.



Thanks for your attention!

Changhao Chen
HKUST (GZ)
changhaochen@hkust-gz.edu.cn
Homepage: changhao-chen@github.io