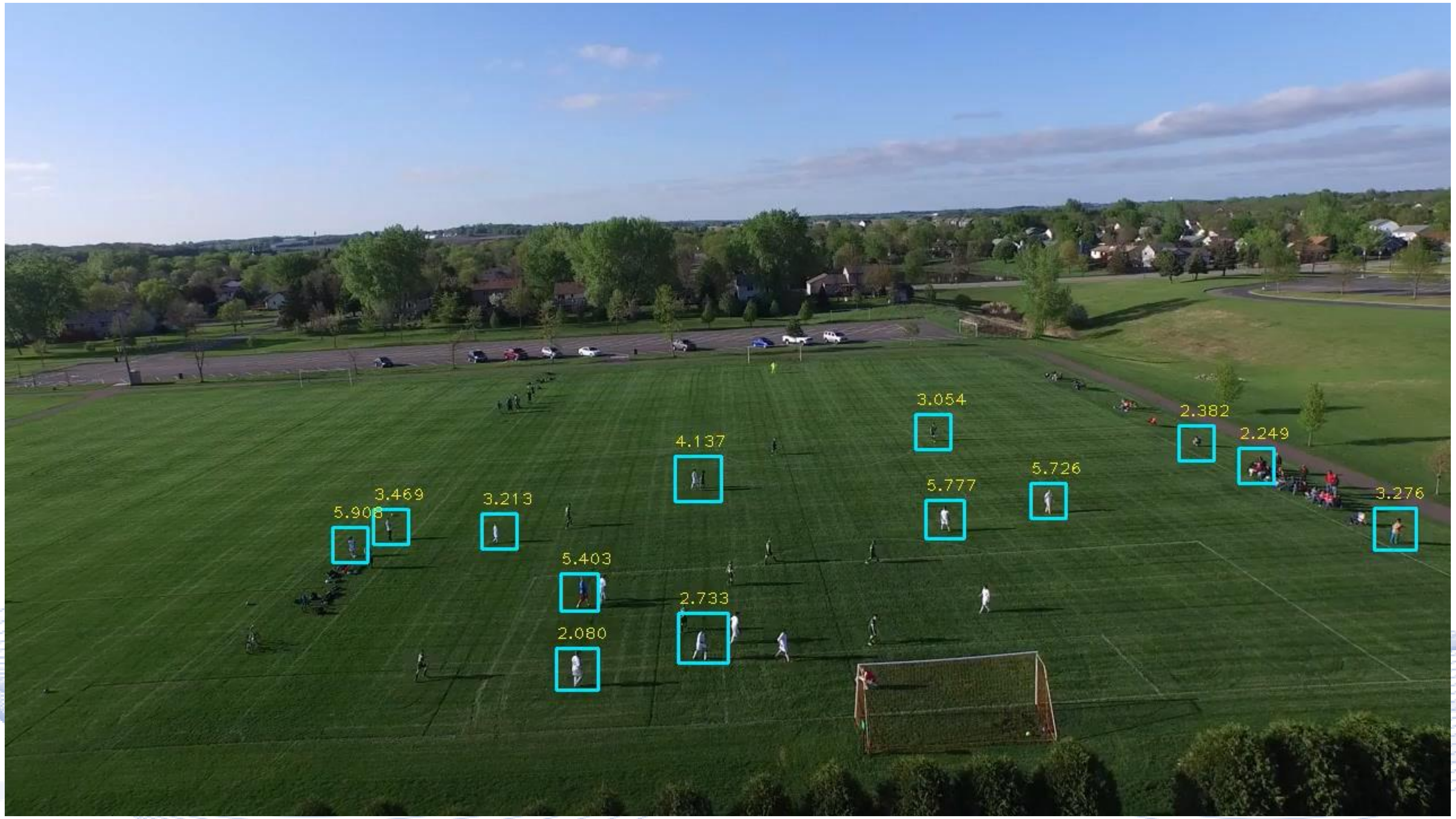
Deep Object Detection

V. Nousi, P. Mentesidis, E. Patsiouras, A. Tefas, I. Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 5.0

Object Detection for UAV sports cinematography

Artificial Intelligence &
Information Analysis Lab

Object Detection for UAV sports cinematography



Object Detection for UAV sports cinematography

Target/object examples: athletes, boats, bicycles.

Object Detection

Many deep object detectors exist that are pretrained on IMAGENET:

- Faster R-CNN [REN2015]
- Single Shot Detection (SSD) [LIU2016]
- You Only Look Once (YOLO) [RED2016], [RED2017], [RED2018].

The performance of such detectors for on-drone deployment has been tested extensively [NOU2018].

Object Detection

- Object detection = classification + localization:
- Find ***what*** is in a picture as well as ***where*** it is.

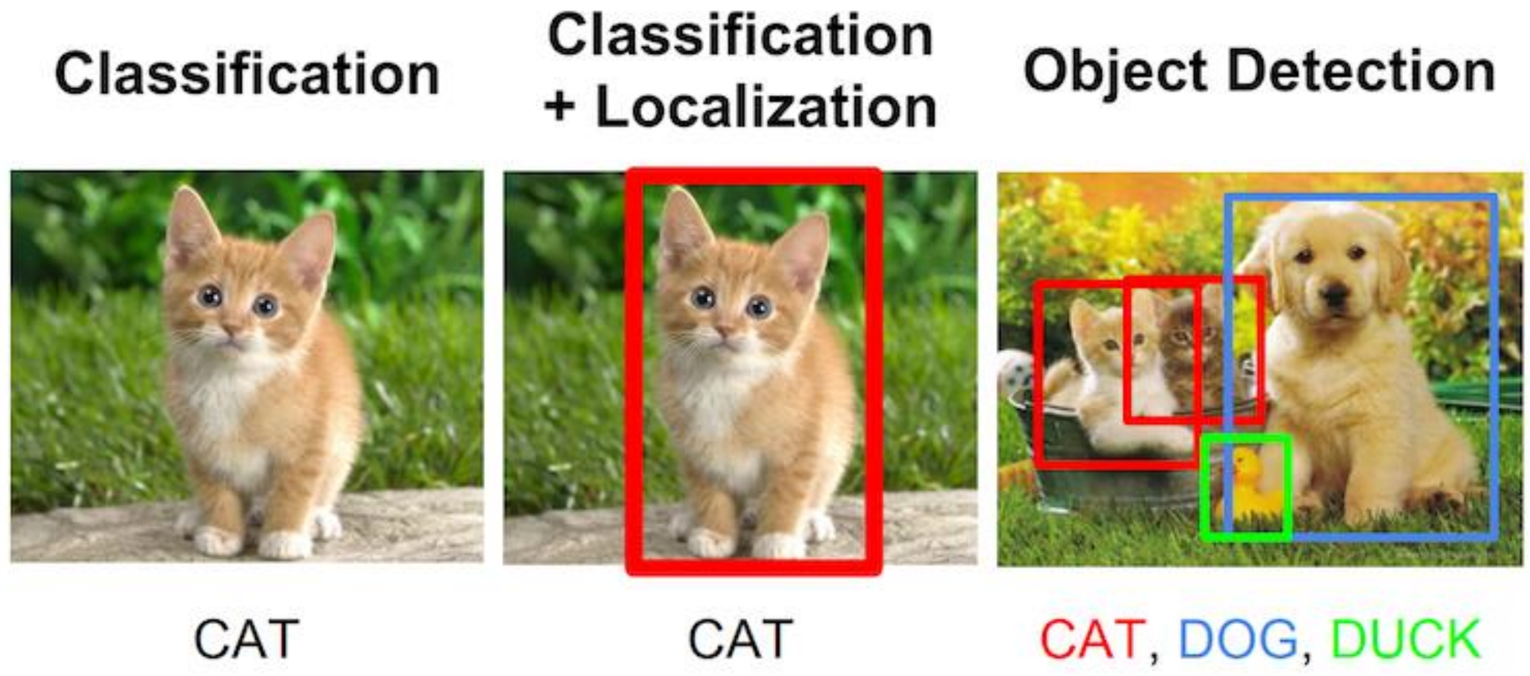


Figure: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Object Detection

- ***Input***: an image.
- ***Output***: ***bounding boxes*** containing depicted objects.
  - Each image may contain a different number of detected objects.
- Old approach: train a specialized classifier and deploy in ***sliding-window style*** to detect all object of that class.
  - Very inefficient, quite ineffective.
- ***Goal***: combine classification and localization into a ***single architecture for multiple, multiclas object detection***.

Classification/Recognition/ Identification

- Given a set of classes $\mathcal{C} = \{\mathcal{C}_i, i = 1, \dots, m\}$ and a sample $\mathbf{x} \in \mathbb{R}^n$, the ML model $\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta})$ predicts a class label vector $\hat{\mathbf{y}} \in [0, 1]^m$ for input sample $\mathbf{x}$, where $\boldsymbol{\theta}$ are the learnable model parameters.

- Essentially, a probabilistic distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ is computed.
- Interpretation: likelihood of the given sample $\mathbf{x}$ belonging to each class $\mathcal{C}_i$.

- Single-target classification:
  - classes $\mathcal{C}_i, i = 1, \dots, m$ are mutually exclusive: $||\hat{\mathbf{y}}||_1 = 1$.
- Multi-target classification:
  - classes $\mathcal{C}_i, i = 1, \dots, m$ are not mutually exclusive : $||\hat{\mathbf{y}}||_1 \geq 1$.

Classification/Recognition/ Identification

- A sufficient large training sample set $\mathcal{D}$ is required for Supervised Learning (regression, classification):

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}.$$

- $\mathbf{x}_i \in \mathbb{R}^n$: n –dimensional input (feature) vector of the i-th training sample.
- $\mathbf{y}_i$: its target label (output).
- Target vector $\mathbf{y}$ can be:
  - real-valued vector: $\mathbf{y} \in [0,1]^m, \mathbf{y} \in \mathbb{R}^m$;
  - binary-valued vector $\mathbf{y} \in \{0,1\}^m$ or even categorical.

Classification/Recognition/ Identification

- ***Training***: Given N pairs of training samples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in [0,1]^m$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min_{\boldsymbol{\theta}} J(\mathbf{y}, \hat{\mathbf{y}})$.

- ***Inference/testing***: Given N_t pairs of testing examples $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N_t\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in [0,1]^m$, compute (predict) $\hat{\mathbf{y}}_i$ and calculate a performance metric, e.g., classification accuracy.

Regression

Given a sample $\mathbf{x} \in \mathbb{R}^n$ and a function $\mathbf{y} = \boldsymbol{f}(\mathbf{x})$, the model predicts ***real-valued quantities*** for that sample: $\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\hat{\mathbf{y}} \in \mathbb{R}^m$ and $\boldsymbol{\theta}$ are the learnable parameters of the model.

- **Training**: Given N pairs of training examples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^m$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min_{\boldsymbol{\theta}} J(\mathbf{y}, \hat{\mathbf{y}})$.

- **Testing**: Given N_t pairs of testing examples $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N_t\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^m$, compute (predict) $\hat{\mathbf{y}}_i$ and calculate a performance metric, e.g., MSE.

Regression

- **Regression:**
  - Example: In object detection, localize the object:
  - regress object ROI parameters: ROI center (x_c, y_c), width w, height h).
  - **Function approximation:** it is essentially regression, when the function $\mathbf{y} = \boldsymbol{f}(\mathbf{x})$ is known.

Object Detection

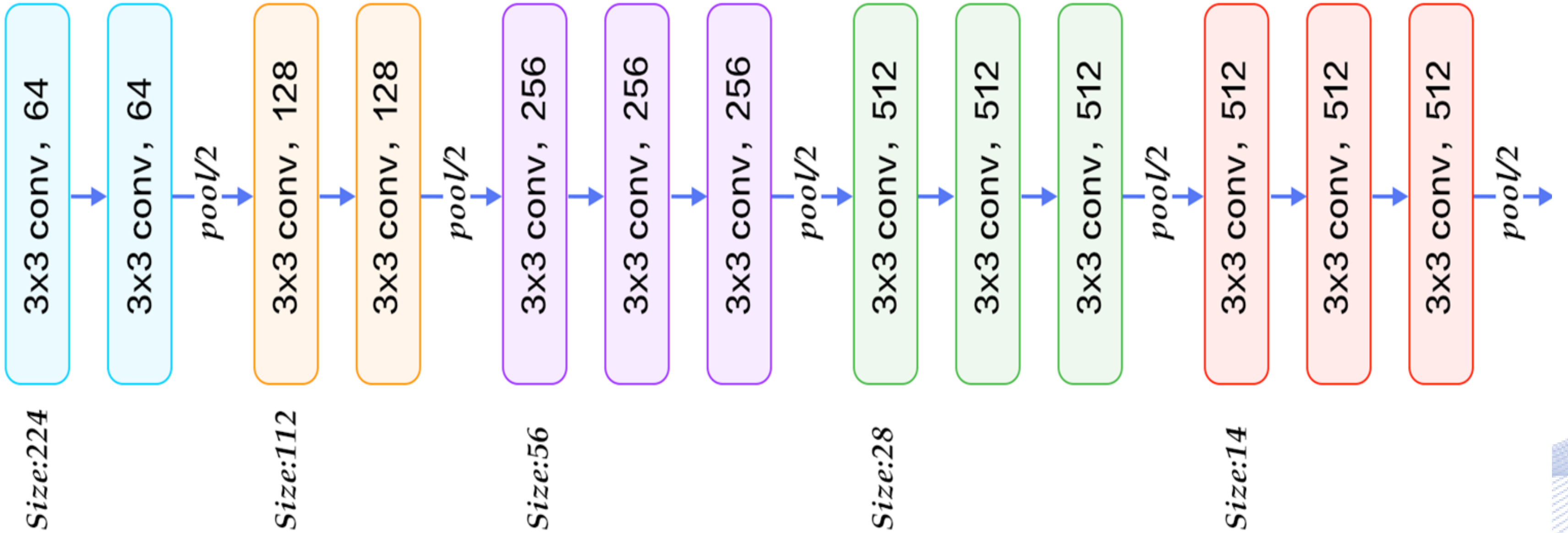
Object detection is a ***multitask machine learning*** problem:

- combination of classification and regression.
- Given a set of classes $\mathcal{C} = \{\mathcal{C}_i, i = 1, \ldots, m\}$ and an image sample $\mathbf{x} \in \mathbb{R}^n$, the model predicts (for one object instance only) an output vector $\hat{\mathbf{y}} = [\hat{\mathbf{y}}^T{}_1 | \hat{\mathbf{y}}^T{}_2]^T$ consisting of:
  - A class vector $\hat{\mathbf{y}}_1 \in [0, 1]^m$ and
  - A bounding box parameter vector $\hat{\mathbf{y}}_2 = [x, y, w, h]^T$ corresponding to object ROI.
- Optimization of a joint cost function:

$$\min_{\boldsymbol{\theta}} J(\mathbf{y}, \hat{\mathbf{y}}) = \alpha_1 J_1(\mathbf{y}_1, \hat{\mathbf{y}}_1) + \alpha_2 J_2(\mathbf{y}_2, \hat{\mathbf{y}}_2).$$

- The above vector pair will be computed for every possible target detected in the image sample $\mathbf{x}$.

Object Detection with CNNs



Object detection: CNN pipeline for bounding box regression.

CNN Object Detection

Region proposal-based detectors

- R-CNN, Fast R-CNN, Faster R-CNN
- R-FCN

Single Stage Detectors

- YOLO
- SSD
- YOLO v2, v3, v4
- RetinaNet, RBFnet
- CornerNet, CenterNet

Transformer Detectors

- DETR.

R-CNN

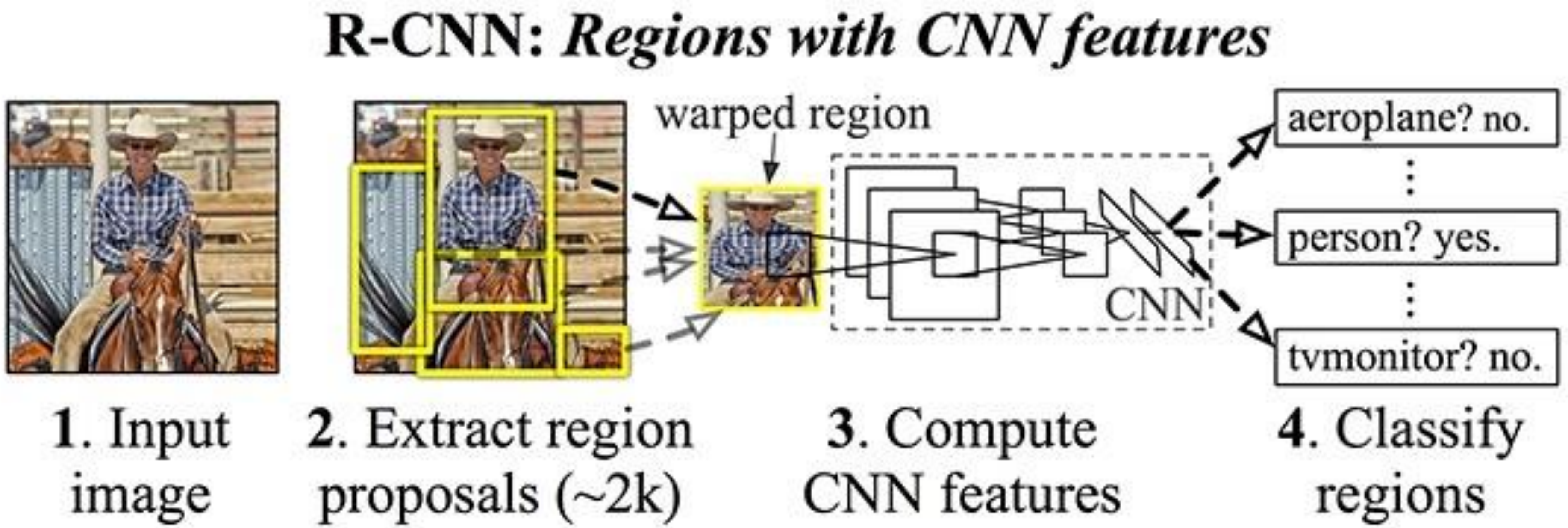
Regions with CNN features (***R-CNN***).

- Three step approach:

1. Extract ***region proposals*** using an external proposal method (i.e., Selective Search). Cropped and resized proposed input image regions form ***crops***, always having the same size.
2. Extract ***CNN features*** for each crop.
3. a) ***Classify*** features with an SVM.

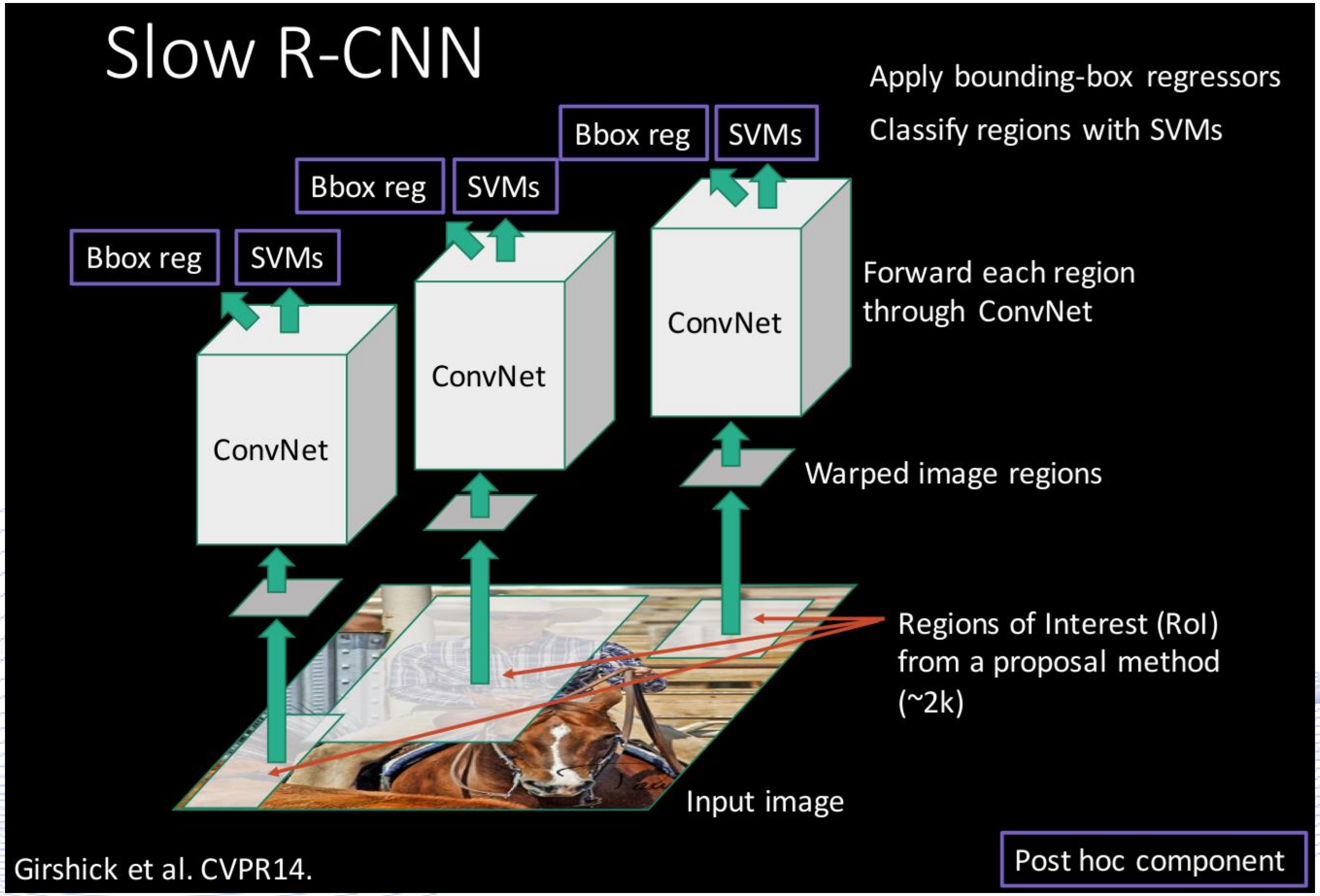
   b) ***Regress*** Region Of Interest (ROI) width w and height h, based on the proposed and validated crops.

R-CNN



R-CNN structure [GIR2014].

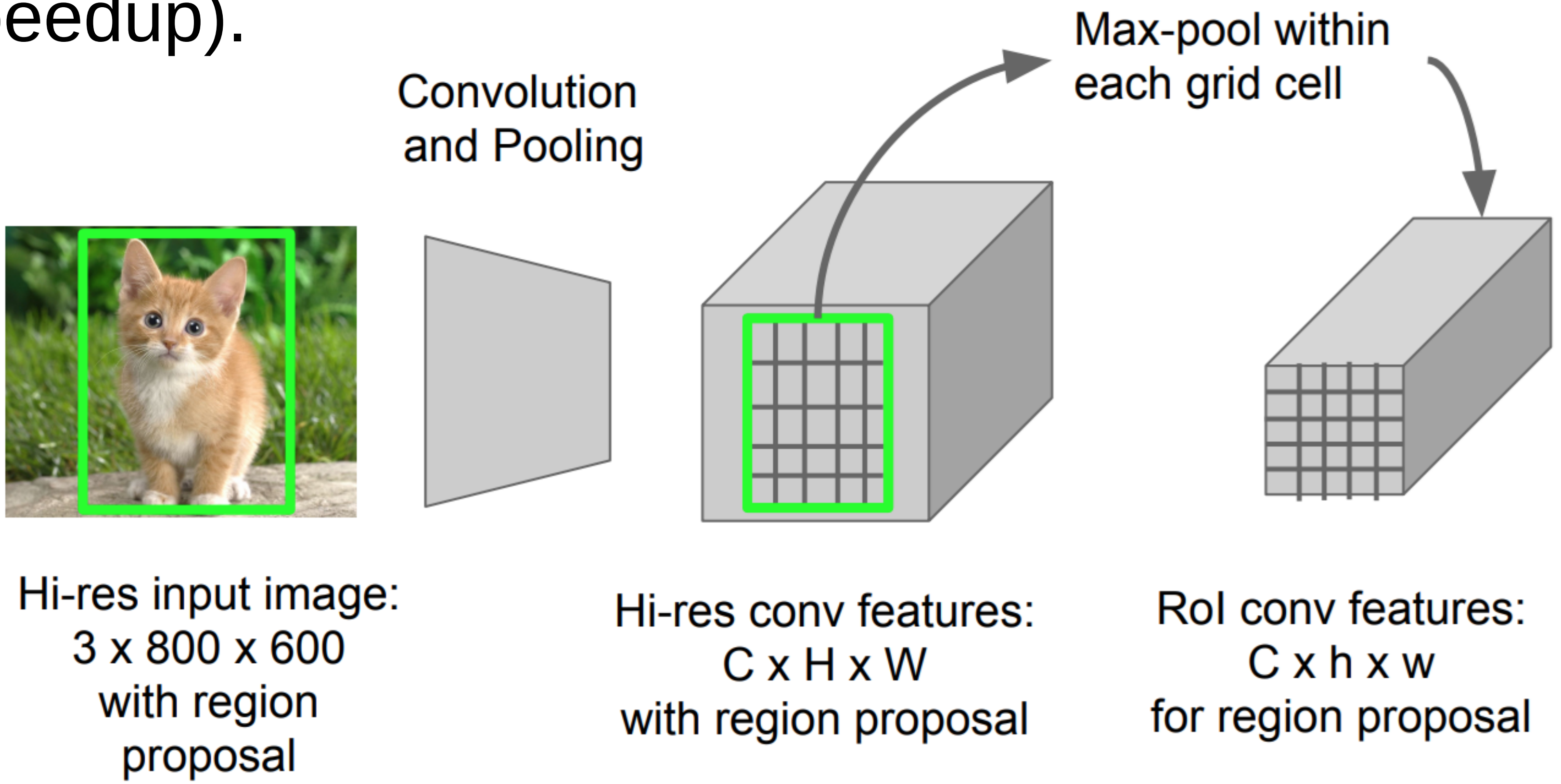
R-CNN



Slow R-CNN
Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.
Post hoc component

Fast R-CNN

Input image is passed once from a CNN to generate a CNN feature map (big speedup).



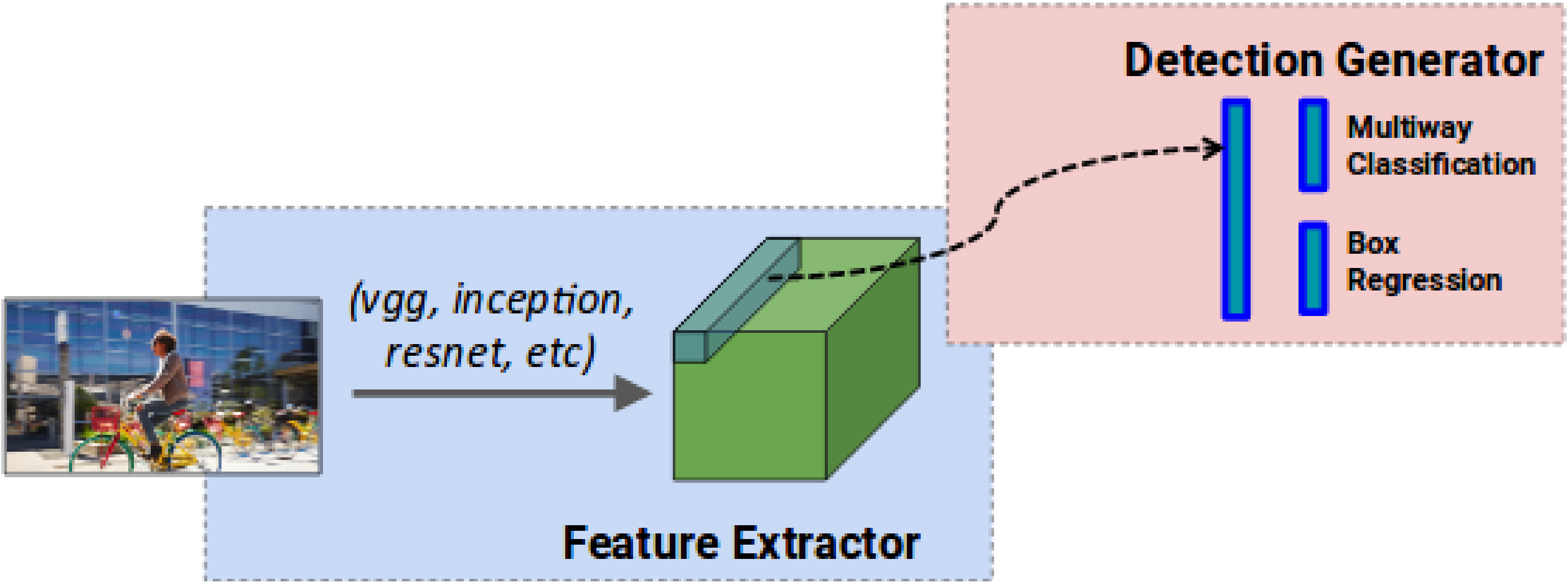
ROI pooling.

SSD

Single-Shot Detector (***SSD***).

- Region-based object detection (R-CNN, Fast R-CNN, Faster R-CNN, R-FCN): accurate, but too slow for real-time applications.
- SSD approach: ***Combine a classification network and bounding box regression into single architecture***, without any external steps or duplicated computations.
- It uses ***anchors*** (***ROIs of precomputed size and aspect ratio***). No region proposals are used.

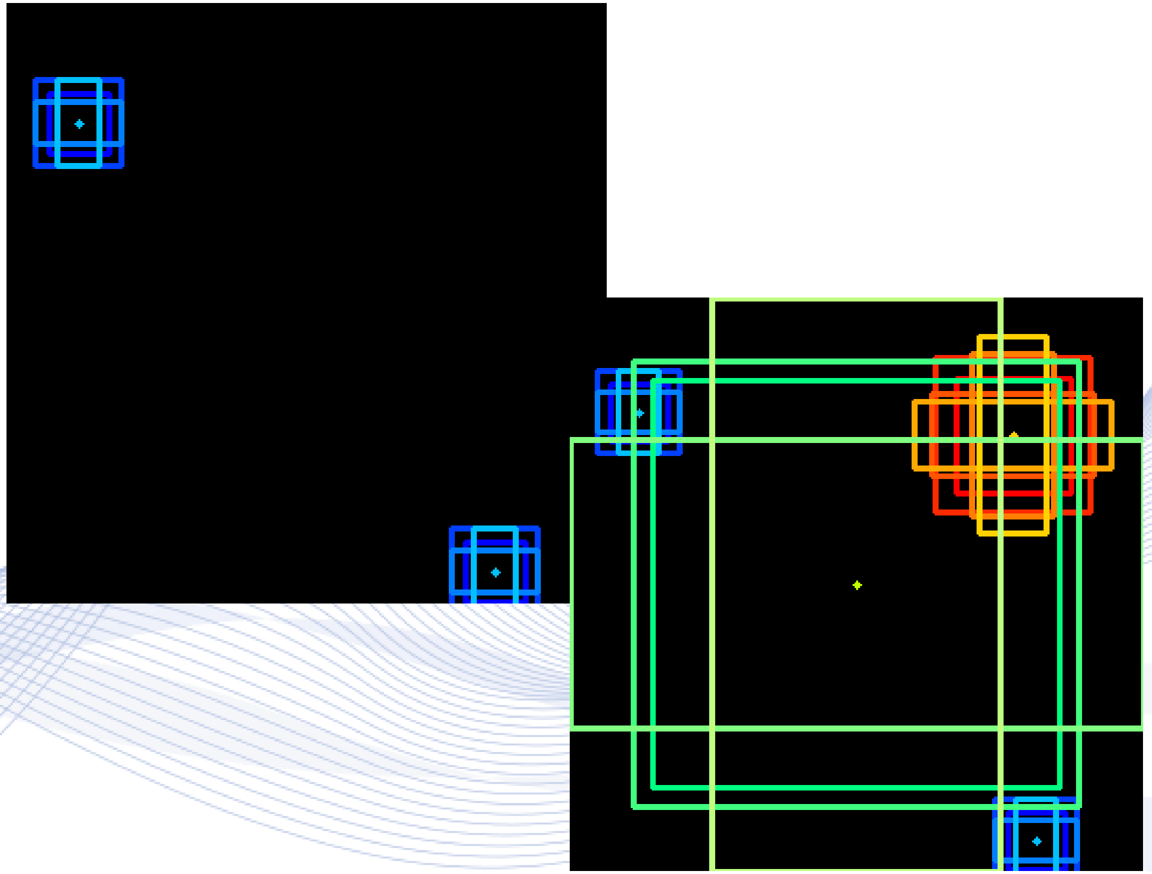
SSD



SSD architecture [HUA2017].

Single Shot Detector

- Anchors ***overlap at various spatial locations***, aspect ratios and scales of the feature maps on various CNN layers.
- During training, anchor location and size are ***refined via regression*** to better fit objects.



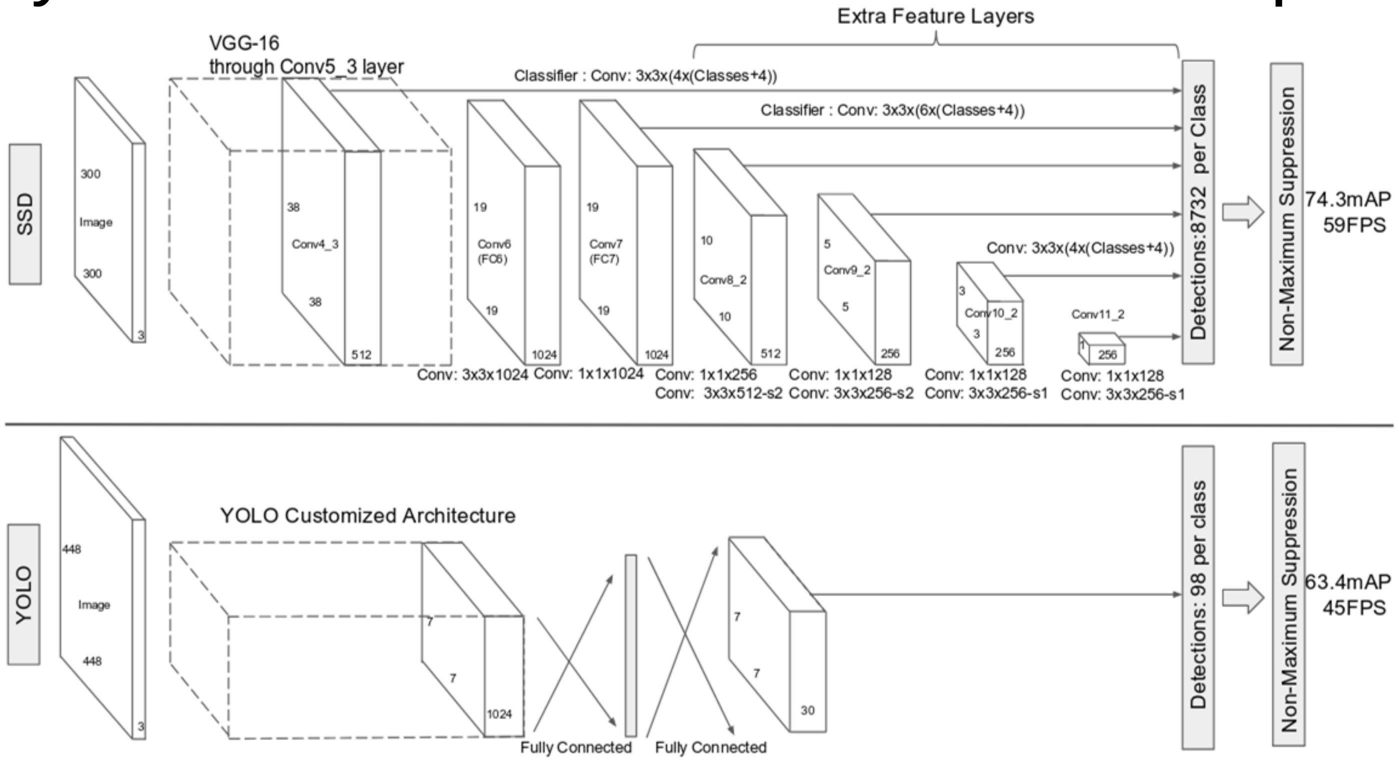
CNN Object Detection architectures

- **Backbone** refers to a CNN used for image feature extraction:
  - ResNet, MobileNet, VGG etc.
- **Neck** is extra object detector layers that go on top of the backbone. They extract different feature maps from different stages of the backbone.
  - FPN, PANet, Bi-FPN etc.
- ***Head*** network performs actual object detection: classification (probability of $m+1$ classes) and regression of RoI parameters (x, y, h, w).

YOLO

YOLO (***You Only Look Once***) architecture:

- Darkenet19 convolutional network plus FC layer.
- Prediction only at the final convolutional feature map.



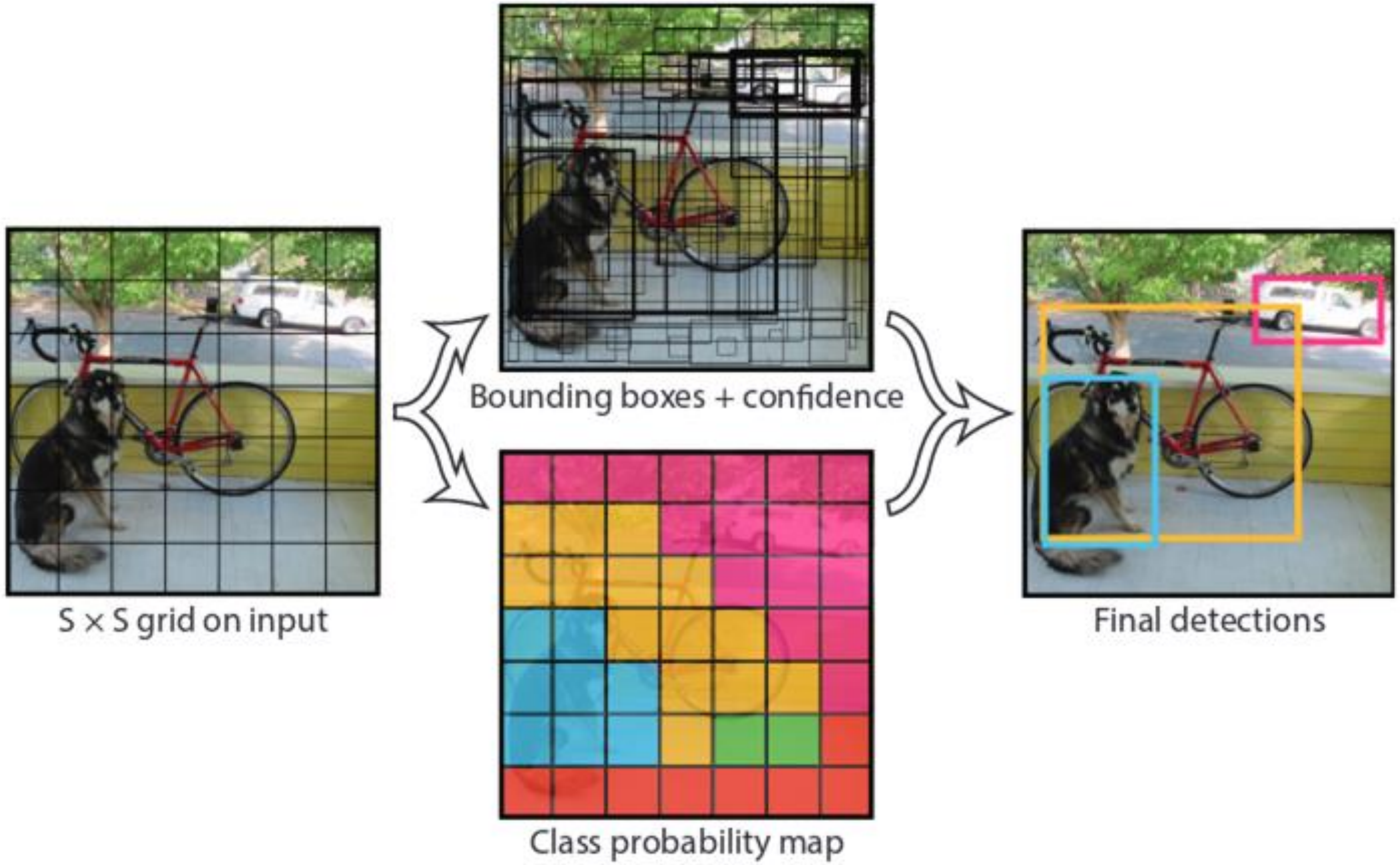
[LIU2016]

Artificial Intelligence & Information Analysis Lab

YOLO

- YOLO divides the input image into an $S \times S$ ***grid***.
- If the ***center*** of an object falls within a cell of the grid, that cell is responsible for detecting that object.
- N is the maximal number of bounding boxes that each grid cell can detect.
- Each cell predicts N ***bounding boxes*** and confidence and classification scores for those boxes.
- The maximal number of detected objects is $N \times S \times S$.

YOLO



Yolo object detection [RED2016].

YOLO

Each ROI is assigned five object predicted values: x, y, w, h and ***confidence***:

$$Pr(Object) \times J(\mathcal{A}, \mathcal{B}) = Pr(Object) \times \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}.$$

- $\mathcal{A}, \mathcal{B}$: estimated, ground truth bounding boxes.

It takes into account:

- a) confidence on object box existence/classification.
- b) how accurate the predicted box is.

YOLO v2

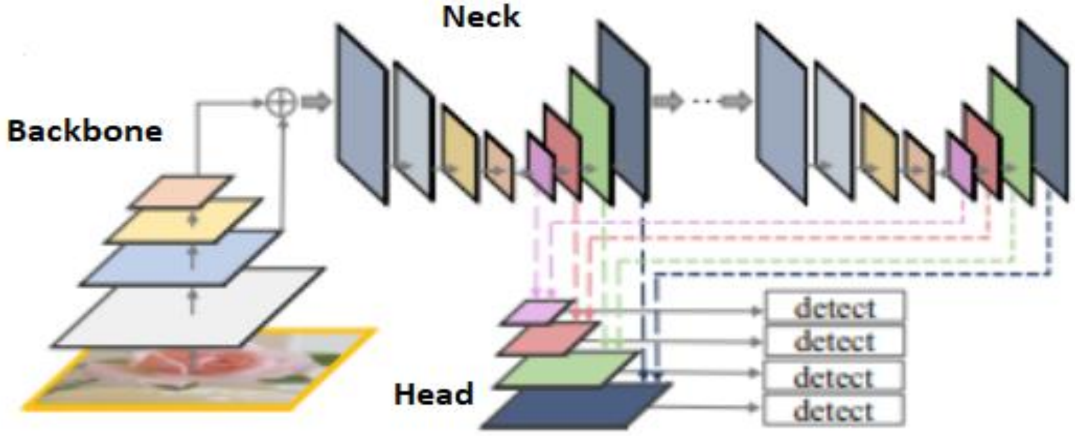
- ***Fully convolutional***, no densely-connected layers:
  - It may be run at varying input sizes.
- It can utilize ***multi-scale capabilities*** during training as well.
- ***Very fast*** architecture and implementation.
- Uses ***precomputed anchors***.

YOLO v3

- ***Deeper ResNet-based backbone architecture***: 53 convolutional layers with skip connections.
- ***Multiscale Detection***: detection occurs at multiple layers at different points in the architecture, to detect objects of different scales.
- Much ***better mAP***, but ***significantly slower.***
- Much ***better at detecting small objects*** [RED2018].

YOLO v4

YOLO v4 design:



[BOC2020]

- ***Backbone***: CSPDarknet53.
- ***Neck***: Spatial pyramid pooling (SPP) and Path Aggregation Network (PAN).
- ***Head***: Same as YOLO v3.

YOLO v4

- It achieves state-of-the-art results on both accuracy and inference time, surpassing all previous object detectors [BOC2020].

- It can be trained on a single conventional GPU with 8/16 GB of VRAM, such as an Nvidia 1080Ti or a 2080Ti GPU.

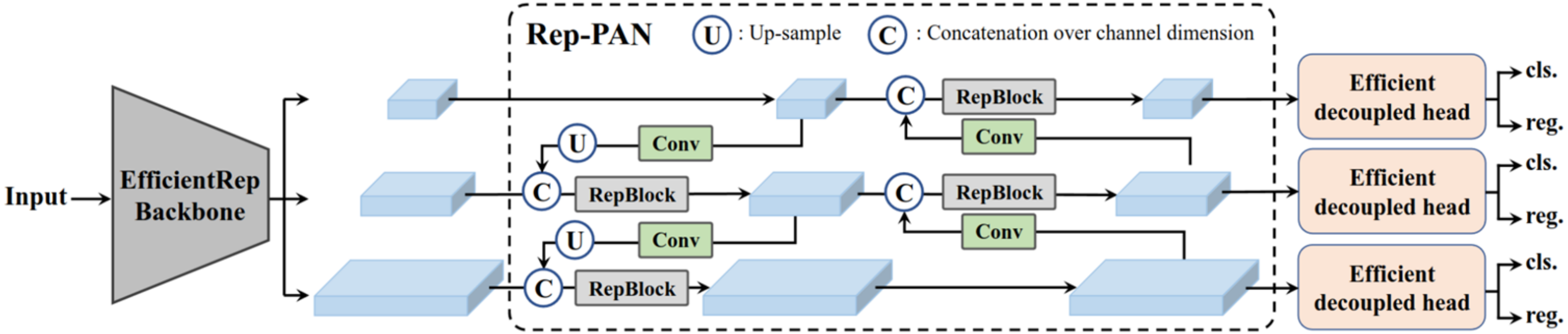
YOLO v5

- ***Detection at Multiple Layers***: Object detection at three scales, improving identification of large and small objects.
- ***Auto-Learning Bounding Box Anchors***: Automatic learning of optimal anchors during training, enhancing performance.
- ***Data Augmentation Techniques***: Use of Mosaic and MixUp methods for improved robustness and generalization.

YOLO v6

- ***EfficientRep Backbone***: RepVGG and CSPStackRep blocks are combined to optimize speed and accuracy across model sizes.
- ***Rep-PAN Neck***: RepVGG and CSPStackRep blocks are used to enhance feature integration at multiple scales.
- ***Efficient Decoupled Head***: a hybrid-channel strategy is used to reduce the computation costs.

YOLO v6



YOLOv6 Architecture [LI2022].

YOLO models

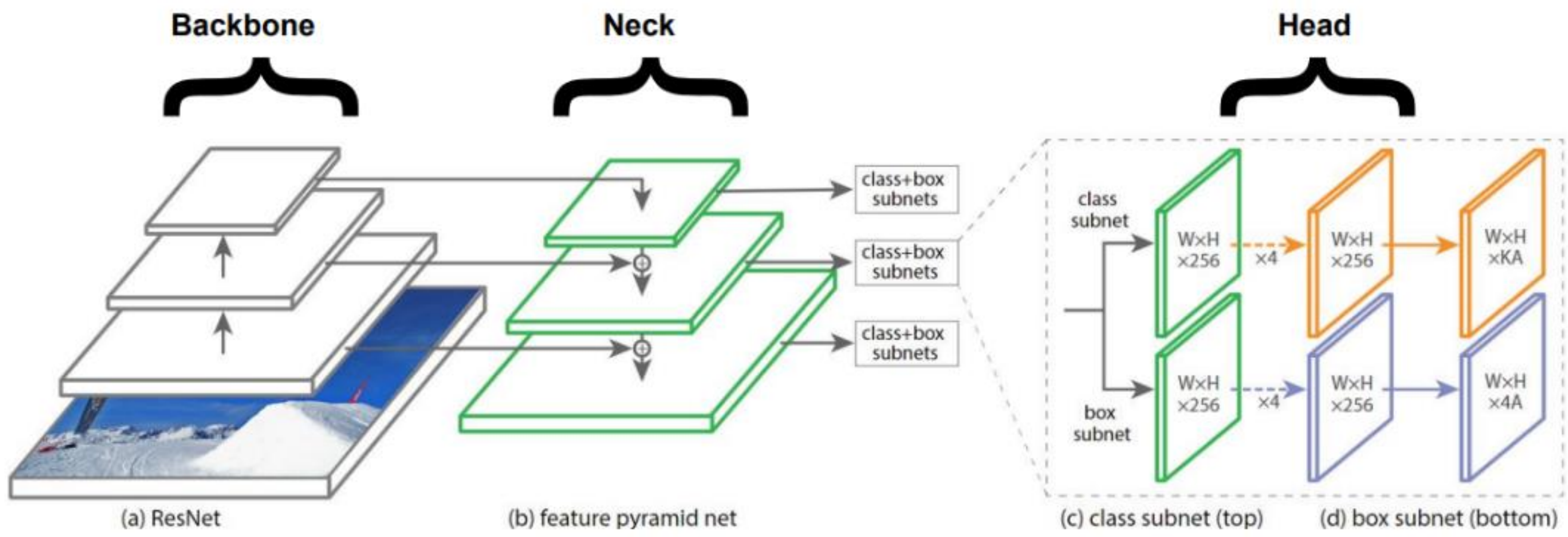
YOLO versions have evolved significantly, each uniquely enhancements to the single-stage detector architecture:

- ***YOLOX***
- ***YOLOv7***
- ***YOLOv8***
- ***YOLOv9***
- ***YOLOv10.***

Each model improves detection accuracy, speed, and robustness for various applications.

RetinaNet

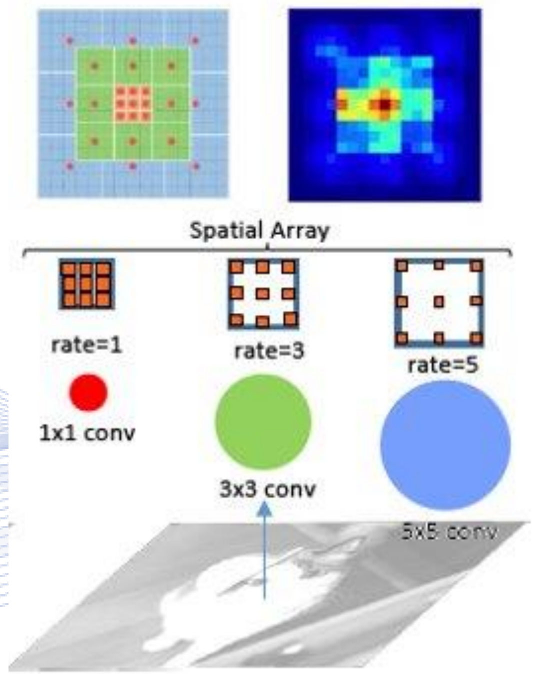
- ResNet is used as a backbone for feature extraction.
- ***Feature Pyramid Network (FPN)*** is used as a neck on top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution image.



RetinaNet architecture [LIN2017].

RFBNet

- It is inspired by the structure of receptive fields in human visual system [LIU2018].
- Use of multiple dilated convolutions with different kernel sizes in each convolutional layer.
- State-of-the-art results and fast inference time.

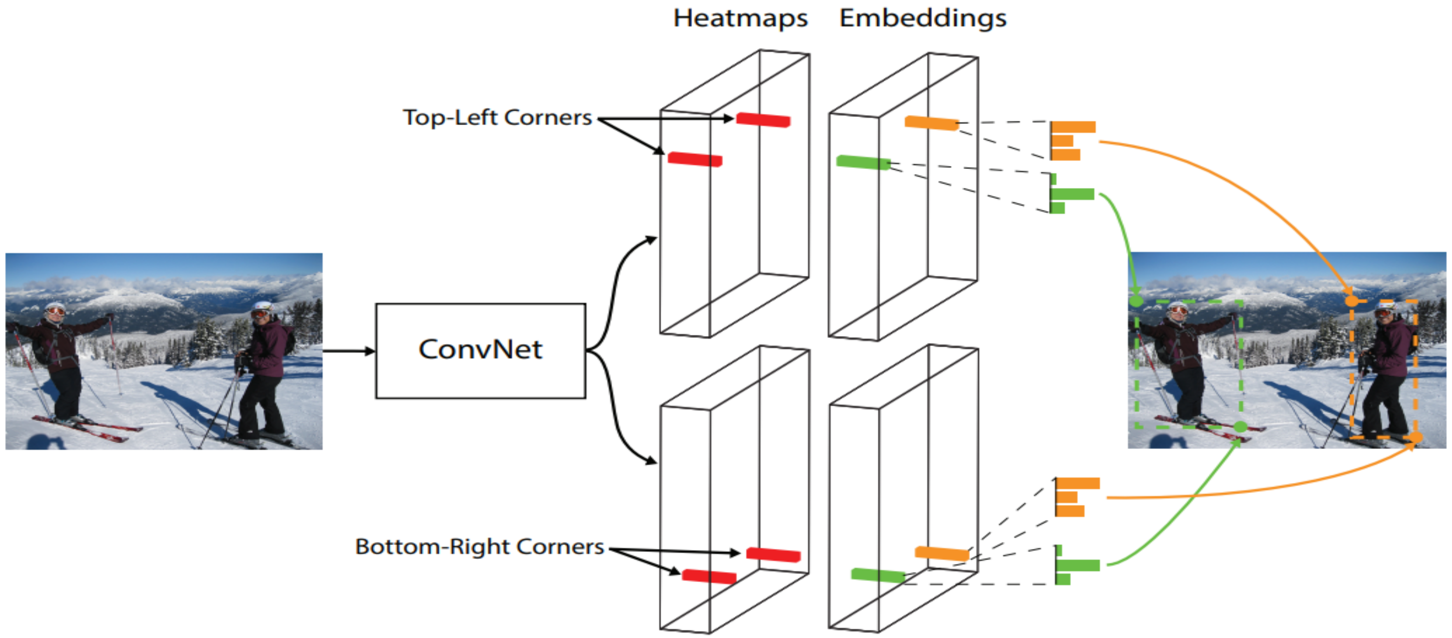


CornerNet

- CornetNet [LAW2018] eliminates the use of anchor boxes and directly predicts a bounding box (RoI) by a pair of keypoints, i.e., its top-left and the bottom-right corner.
- A CNN outputs a heatmap for all top-left and bottom-right corners, as well as an embedding vector for each detected corner, that describes a local neighborhood region.
- CornetNet is trained to predict similar embeddings for corners that belong to same object, by utilizing these heatmaps.

CornerNet

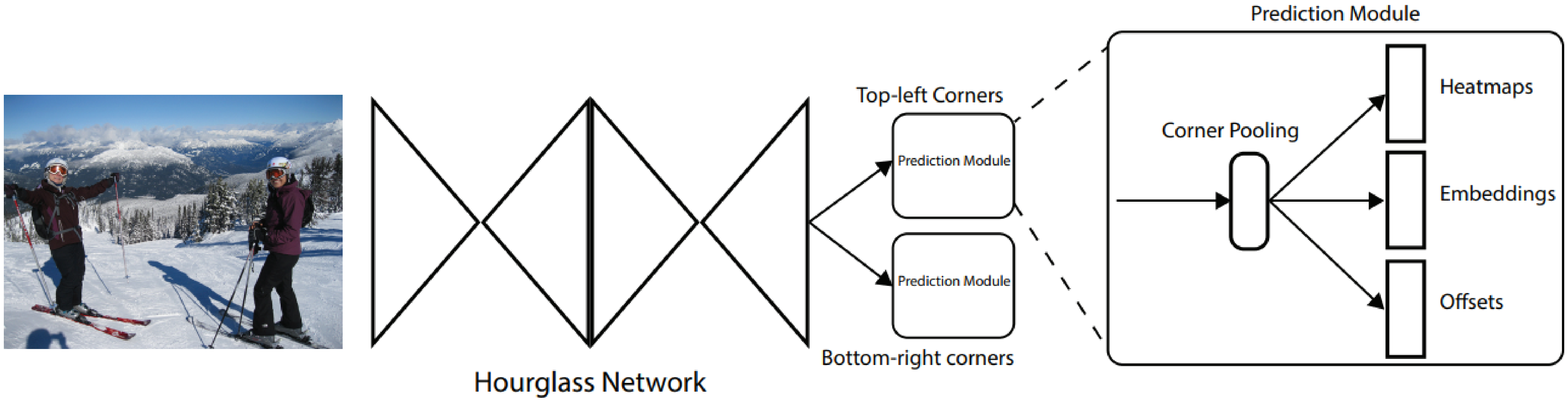
- Each set of heatmaps has C channels and is of size $h \times w$ pixels (C is the number of categories to detect).



CornerNet pipeline [LAW2018].

CornerNet

- CornetNet uses an ***hourglass network*** as a backbone, which is followed by two prediction modules, one for the top-left and one for the bottom-right RoI corners.



CornerNet overview [LAW2018].

DETR

- ***Detection Transformer*** (***DETR***) views object detection as a direct set prediction problem, while removing many hand-designed components like Non-Maximum Suppression (NMS) or anchor generation.

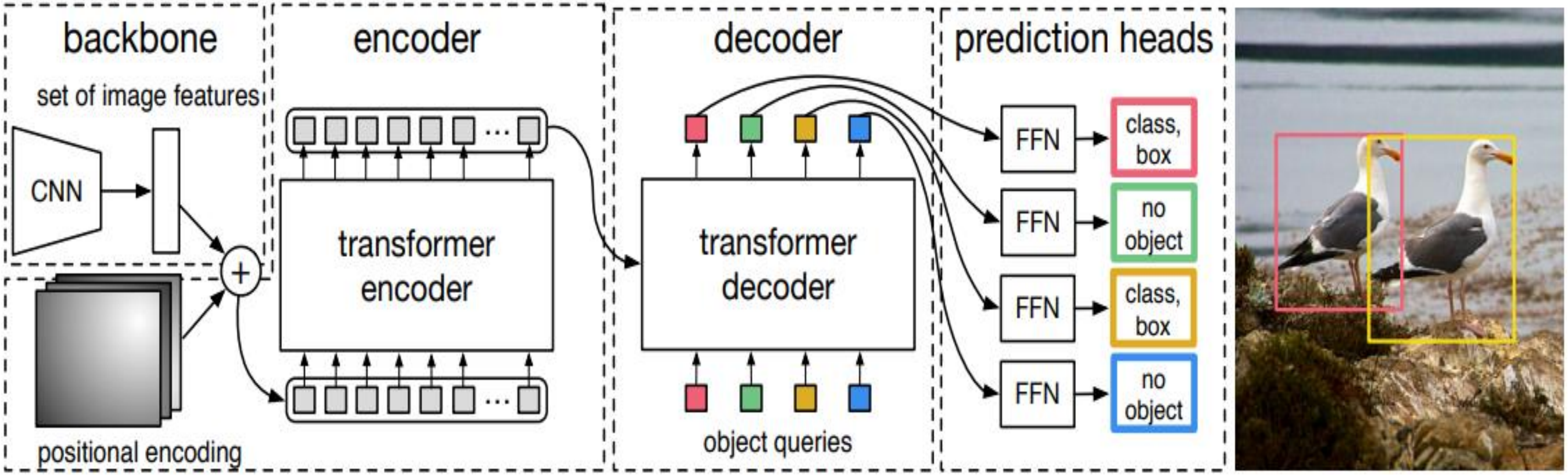
- DETR utilizes an encoder-decoder sequence processing model called ***Transformer*** [VAS2017] and a bipartite matching loss.

DETR

DETR architecture has three main components:

- A RestNet50/101 CNN backbone for feature extraction.
- An encoder-decoder transformer model.
- A feed-forward head network makes the final detection predictions.

DETR



DETR architecture [CAR2020].

DETR

Given a set of ground truth $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, N\}$ of N images and a set of predictions $\hat{\mathbf{y}}_i, i = 1, \ldots, N$ we search for a permutation of N elements $\sigma \in \mathcal{G}_N$ with the lowest cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathcal{G}_N} \sum_{i}^{N} J(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)}).$$

DETR

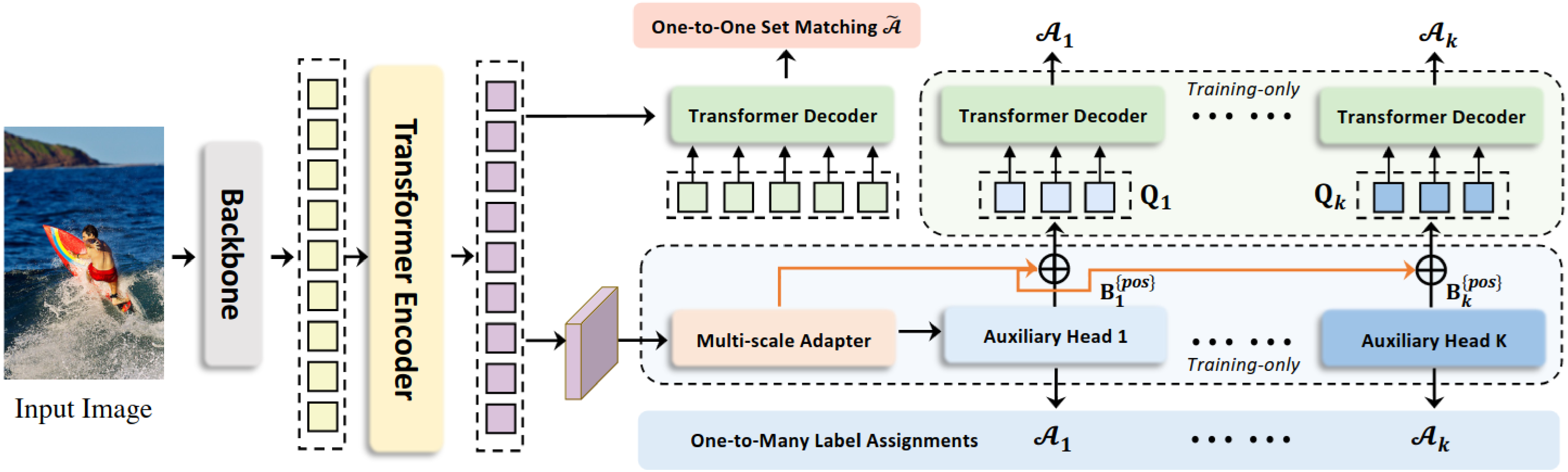
Bipartite matching loss J is a pair-wise matching cost between ground truth $\mathbf{y}_i$ and a prediction $\hat{\mathbf{y}}_{\sigma(i)}$.

- $\mathbf{y}_i = \{\mathbf{y}_{1i}, \boldsymbol{y}_{2i}\}$ where $\mathbf{y}_{1i}$ is the target class label and $\mathbf{y}_{2i}$ defines ground truth box coordinates.
- J significantly reduces low-quality predictions and eliminates the need for output reductions, e.g., by using NMS.

Co-DETR

- ***Vision Transformer*** (***ViT***) as backbone.
- Novel collaborative hybrid assignments training scheme to learn more efficient and effective DETR-based detectors from versatile label assignment manners.

Co-DETR

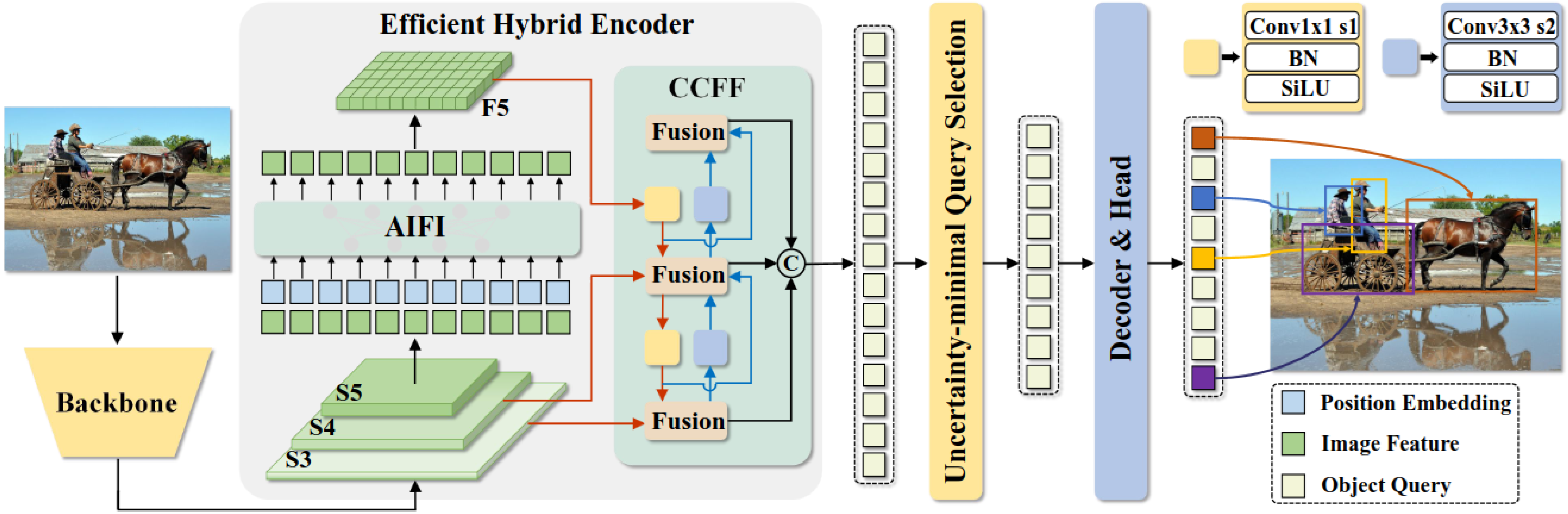


Co-DETR architecture [ZON2023].

RT-DETR

- ***Real-Time DEtection TRansformer*** (***RT-DETR***) extends DETR to the real-time detection scenario and achieves state-of-the-art performance.
- Efficient hybrid encoder that expeditiously processes multiscale features.
- Uncertainty-minimal query selection that improves the quality of initial object queries.

RT-DETR



RT-DETR architecture [ZHA2024].

Object detector speedup

- ***Fine-tuning a pretrained model*** on a new domain (e.g., boat/bicycle detection), instead of training from scratch usually yields better results
- ***Tiny versions of the proposed detectors*** (e.g., Tiny YOLO) can ***increase the detection speed*** (but at the cost of accuracy).

Training datasets created by AUTH

Dataset	Train	Positive	Negative	Test
Crowd	40000	20000	20000	11550
Football	80000	40000	40000	10000
Bicycles	51200	25600	25600	7000
Face	140000	70000	70000	7468

Object detector speedup

- ***Reducing the input image size can also increase the detection speed.***
- However, this can ***significantly impact the accuracy*** when detecting very small objects (which is the case for drone shooting).

Model	Input Size	Pascal 2007 test mAP*
YOLO v.2	544x544	77.44
YOLO v.2	416x416	74.60
YOLO v.2	288x288	67.12
YOLO v.2	160x160	48.72
YOLO v.2	128x128	40.68

Object detector speedup

- **We evaluated the faster detector (YOLO) on an GPU accelerated embedded system** (NVIDIA TX2).
- Adjusting the input image size allows throughput increase.
- Detection with satisfactory accuracy is not real-time.

Model	Input Size	FPS
YOLO v.2	604x604	3
YOLO v.2	544x544	4
YOLO v.2	416x416	7
YOLO v.2	308x308	10
Tiny YOLO	604x604	9
Tiny YOLO	416x416	15

Object Localization Performance Metrics

- ***Intersection over Union (IoU)***:

$$J(\mathcal{A},\mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}.$$

- $\mathcal{A}, \mathcal{B}$: estimated, ground truth ROIs (sets, bounding boxes).
- $|\mathcal{A}|$: set cardinality (area counted in pixels).
- Also called ***Jaccard Similarity Coefficient*** or ***Overlap Score.***

Object Localization Performance Metrics

Object localization performance: a) $J(\mathcal{A}, \mathcal{B}) = 0.67$; b) $J(\mathcal{A}, \mathcal{B}) = 0.27$.

Artificial Intelligence & Information Analysis Lab

Object Detection Performance Metrics

- Let us have N_t test images. Object detection consists of:
- Object classification
  - Performance measured by, e.g., top5error.
- Object localization
  - find object ROI (bounding box) parameters $[x, y, h, w]$ through (CNN) regression.
  - Performance measured by the Jaccard similarity coefficient.

Object Detection Performance Metrics

Top-5 Classification Error:

- Given the ground truth object class label $\mathcal{C}_i$ and top 5 predicted class labels $\mathcal{C}_{i1}, \dots, \mathcal{C}_{i5}$ the prediction is correct, if $\mathcal{C}_{ij} = \mathcal{C}_i, j = 1, \dots, 5$. The error of a single prediction is:

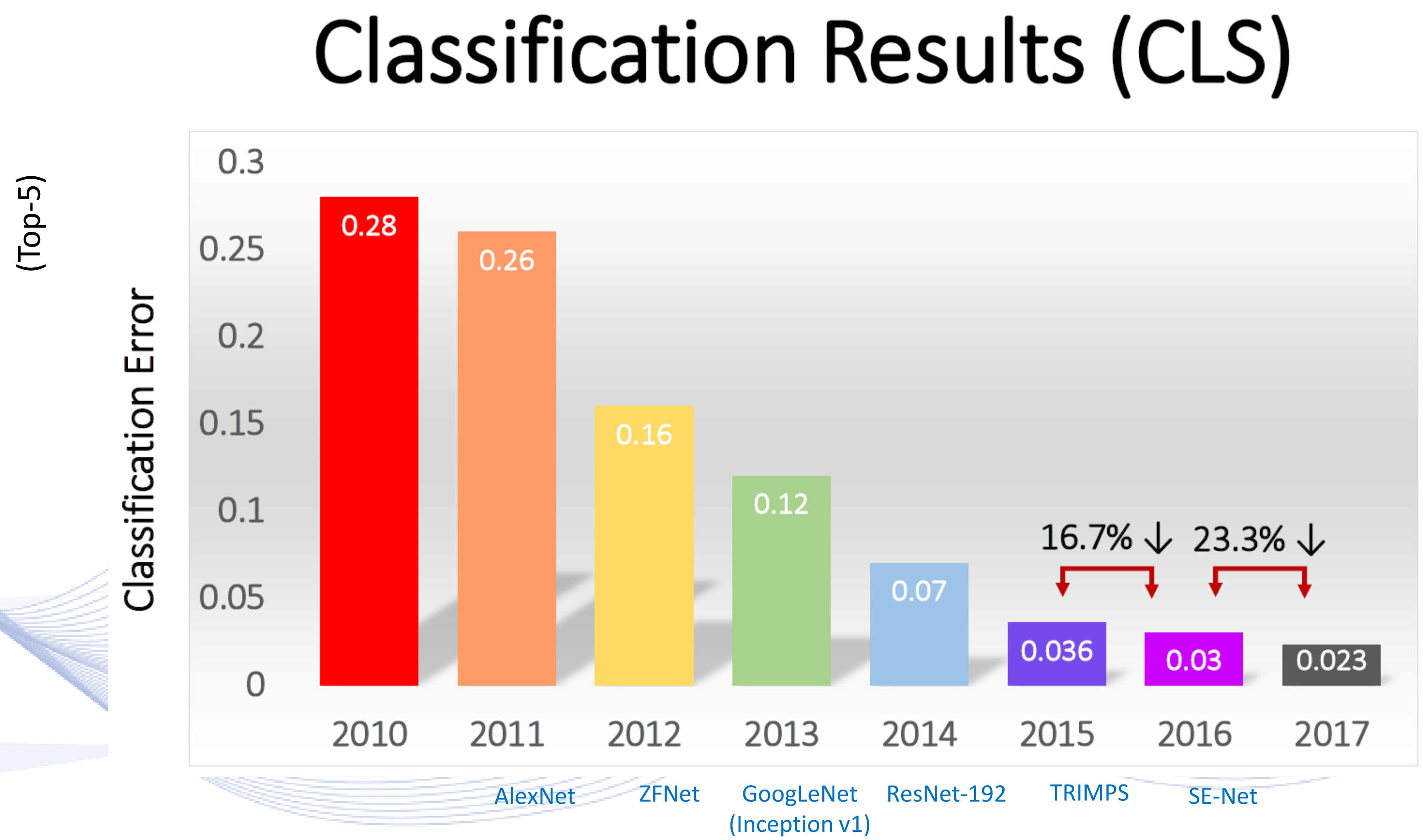
$$e_{CLS}(\mathcal{C}_{ij}, \mathcal{C}_i) = \begin{cases} 1, & \mathcal{C}_{ij} \neq \mathcal{C}_i, \quad \forall j \in [1,5] \\ 0, & \text{otherwise.} \end{cases}$$

Object Detection Performance Metrics

- The top-5 error is the fraction of N_t test images on which the prediction is wrong:

$$top5error_{CLS} = \frac{1}{N_t}\sum_{i=1}^{N_t} \min_j\{e_{CLS}(\mathcal{C}_{ij}, \mathcal{C}_i)\}, \qquad j = 1, \ldots, 5.$$

Object Detection Performance Metrics



Object detection performance history.

Object Detection Performance Metrics

- Object detection on images $i = 1, \ldots, N_t$:

bounding boxes $\mathcal{A}_{ij}$ and confidence scores s_{ij}.

- If $\mathcal{A}_{ij}$ is matched to a ground truth box $\mathcal{B}_{ik}$:

$$J(\mathcal{A}_{ij}, \mathcal{B}_{ik}) > T(\mathcal{B}_{ik}), \quad \text{then } z_{ij} = 1.$$

- The threshold $T(\mathcal{B}_{ik})$ depends on the box size:

$$T(\mathcal{B}_{ik}) = \min(0.5, hw/(h+1)(w+1)).$$

Object Detection Performance Metrics

For M ground truth object ROIs on all N_t images:

- Let $n_{ij} = 1$ for a successful classification at ***confidence threshold*** t $(s_{ij} \geq t)$:
- ***Recall, Precision*** definitions (modified):

$$r(t) = \frac{\sum_{ij} n_{ij} z_{ij}}{M},$$

$$p(t) = \frac{\sum_{ij} n_{ij} z_{ij}}{\sum_{ij} n_{ij}}.$$

Object Detection Performance Metrics

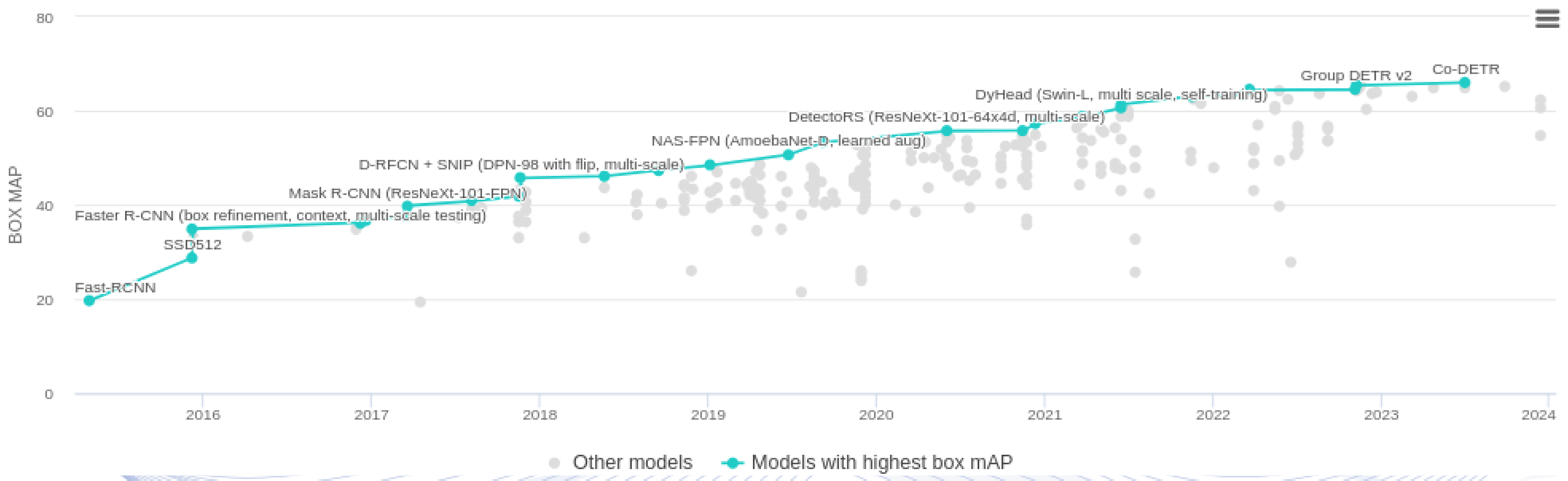
Mean Average Precision (***mAP***):

- It is calculated over N levels of confidence threshold $t_n, n = 1, \ldots, N$:

$$mAP = \frac{1}{N} \Sigma_n \, p(t_n).$$

Artificial Intelligence & Information Analysis Lab

Object Detection Performance Metrics



Box mAP on COCO test-dev [PWC].

Object Detection Performance Metrics

Top-5 Localization Error:

For each test image $i = 1, \ldots, N_t$, let us have:

- a pair of ground truth a) label $\mathcal{C}_i$ and b) bounding box B_{ik},
- a set of classification/localization predictions $\left\{\left(\mathcal{C}_{ij}, \mathcal{A}_{ij}\right)\right\}_{j=1}^{5}$ of class labels $\mathcal{C}_{ij}$ with corresponding bounding boxes $\mathcal{A}_{ij}$.

Object Detection Performance Metrics

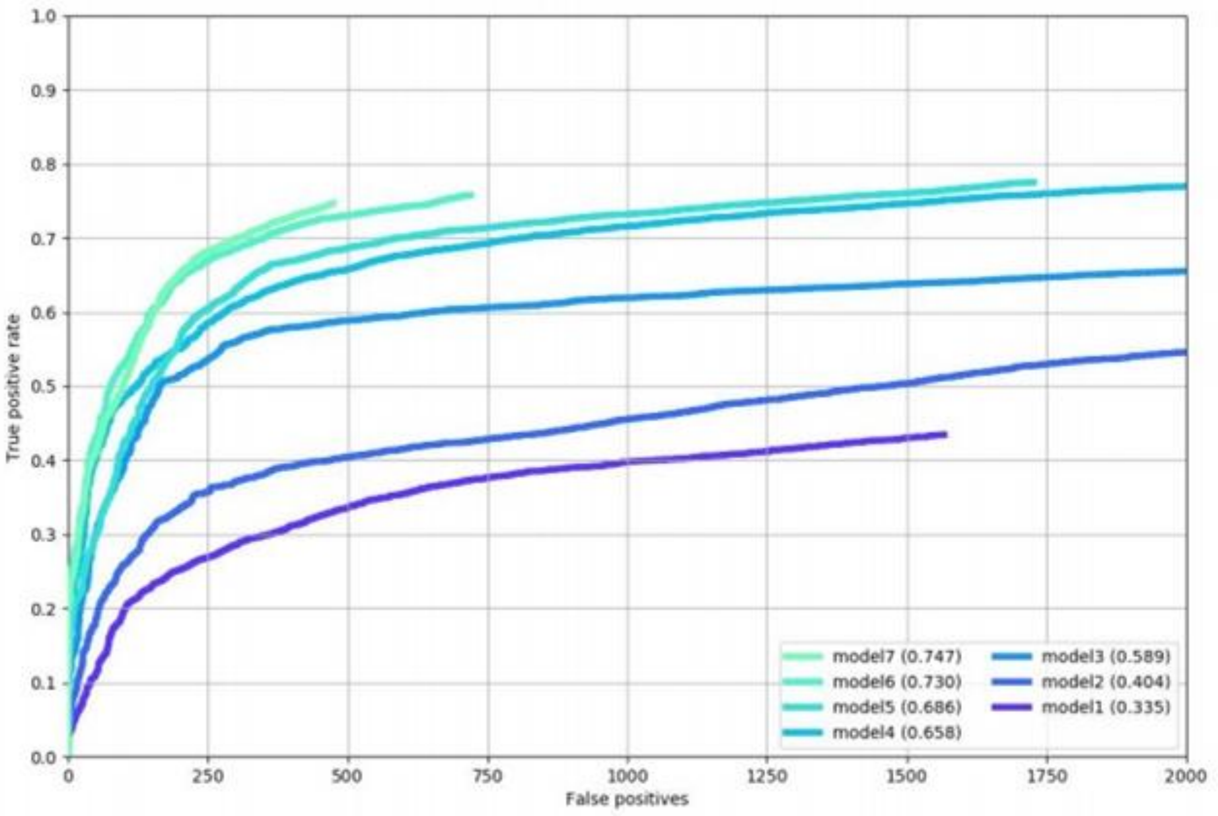
Top-5 Localization Error definition:

$$e_{LOC}(\mathcal{A}_{ij}, \mathcal{B}_{ik}) = \begin{cases} 1, & J(\mathcal{A}_{ij}, \mathcal{B}_{ik}) \leq 0.5 \\ 0, & J(\mathcal{A}_{ij}, \mathcal{B}_{ik}) > 0.5 \end{cases},$$

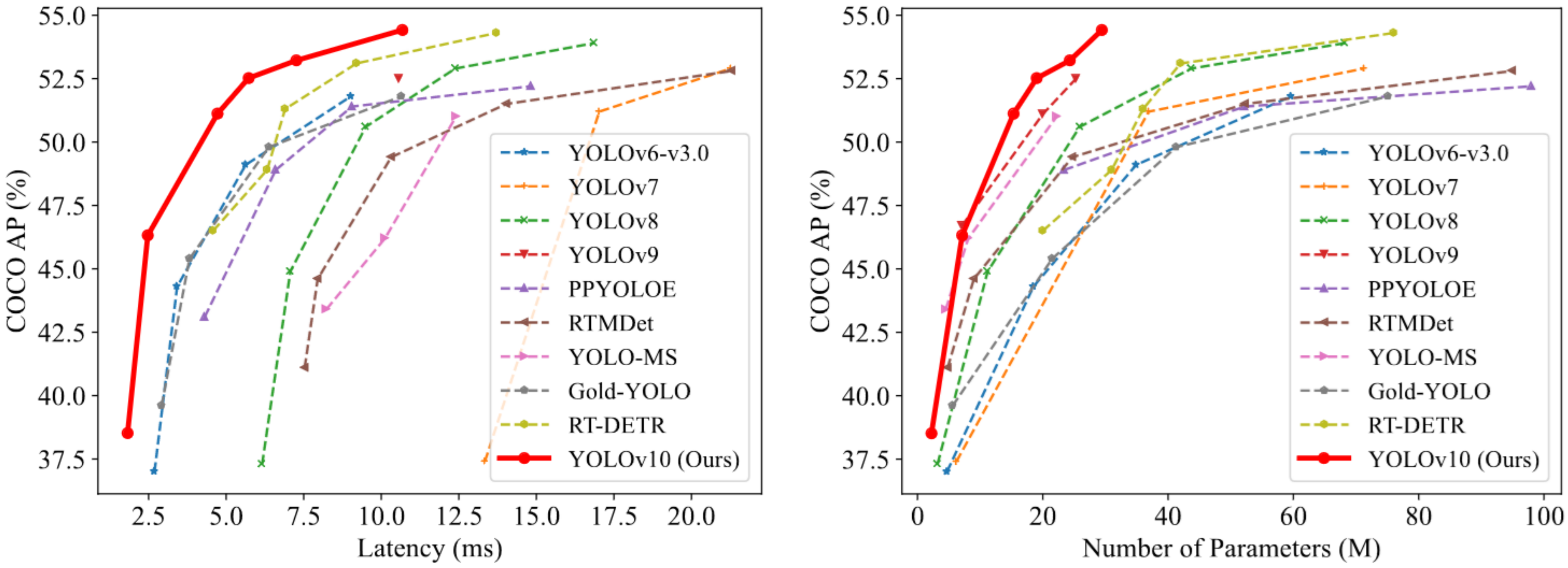
$$top5error_{LOC} = \frac{1}{N_t} \sum_{i=1}^{N_t} \min_j \{e_{LOC}(\mathcal{A}_{ij}, \mathcal{B}_{ik})\}, \qquad j = 1, \ldots, 5.$$

Object Detection Performance Metrics

- ***False Positive*** (***FP***) vs ***True positive*** (***TP***) plots, as a function of detection threshold e.g., for various training stages.
- The closer the curve is to the upper left corner, the better.

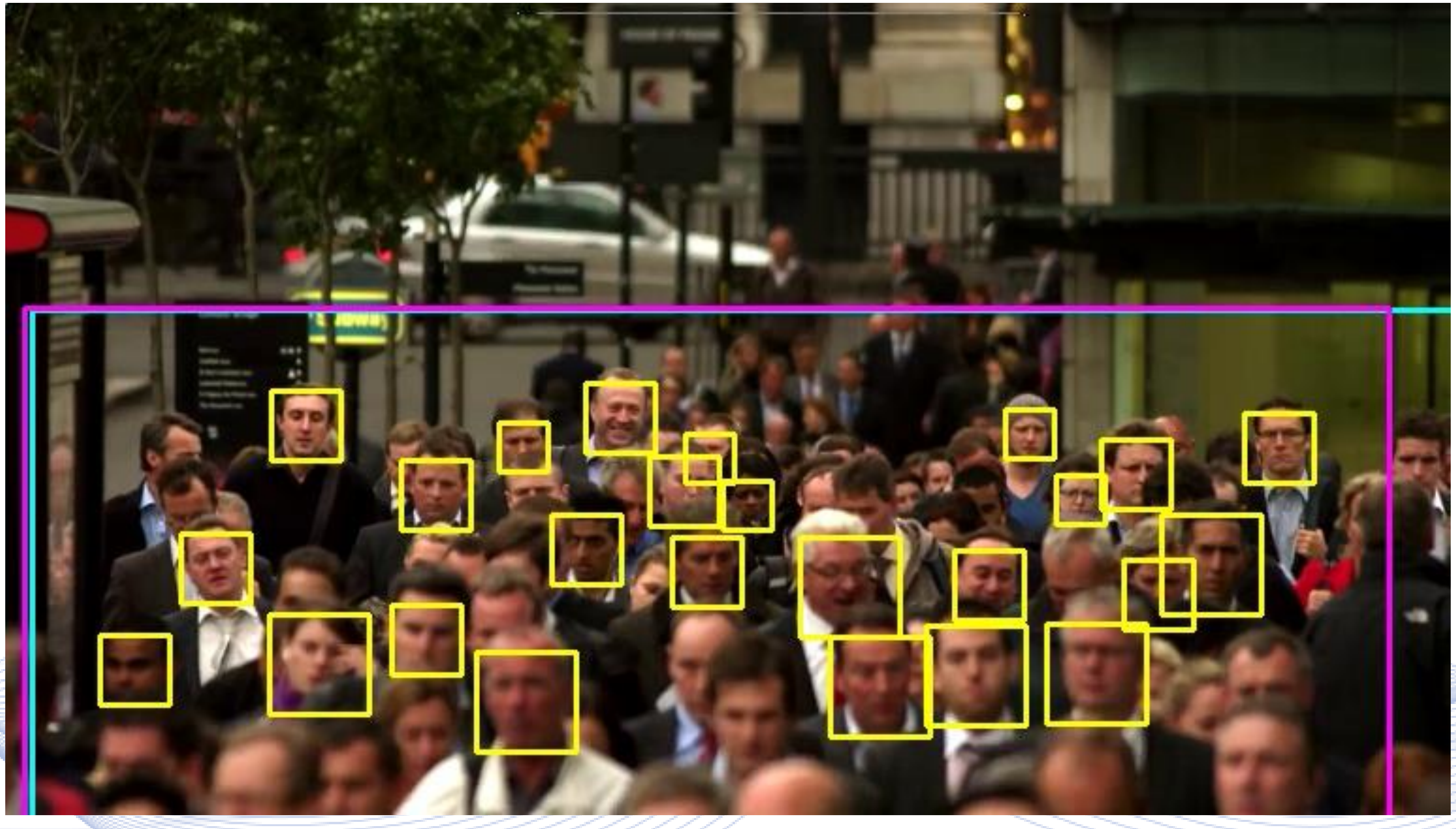


Real-Time Object Detectors



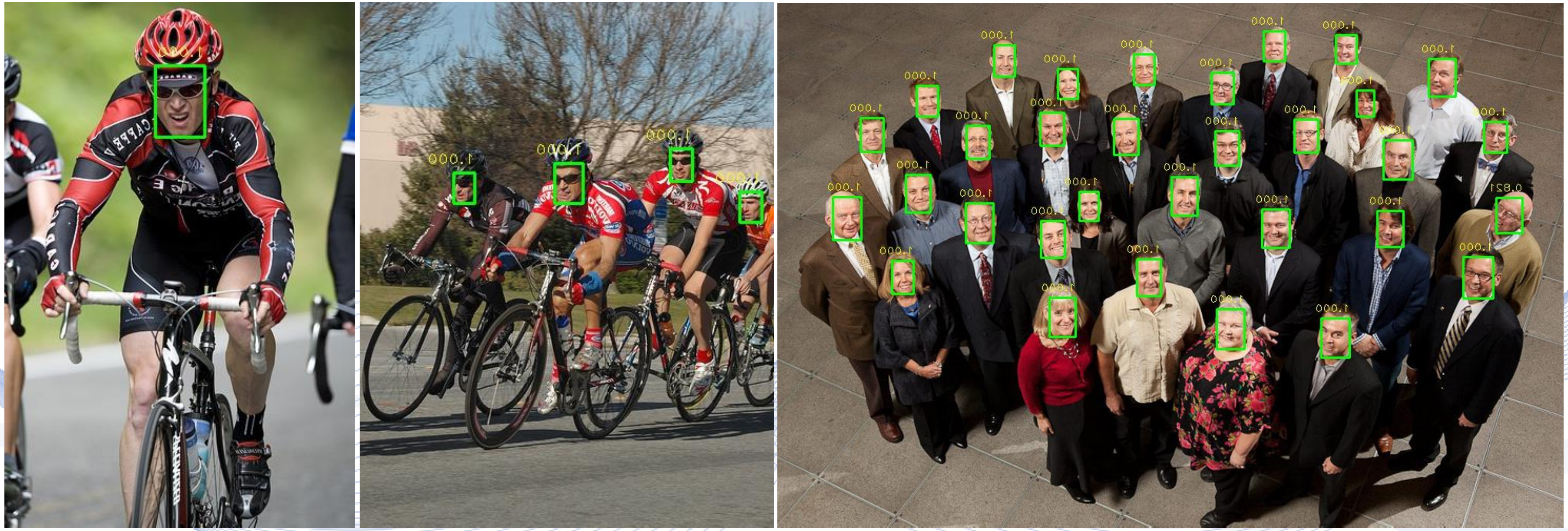
Real time object detectors comparison on COCO dataset [WAN2024].

Face detection examples

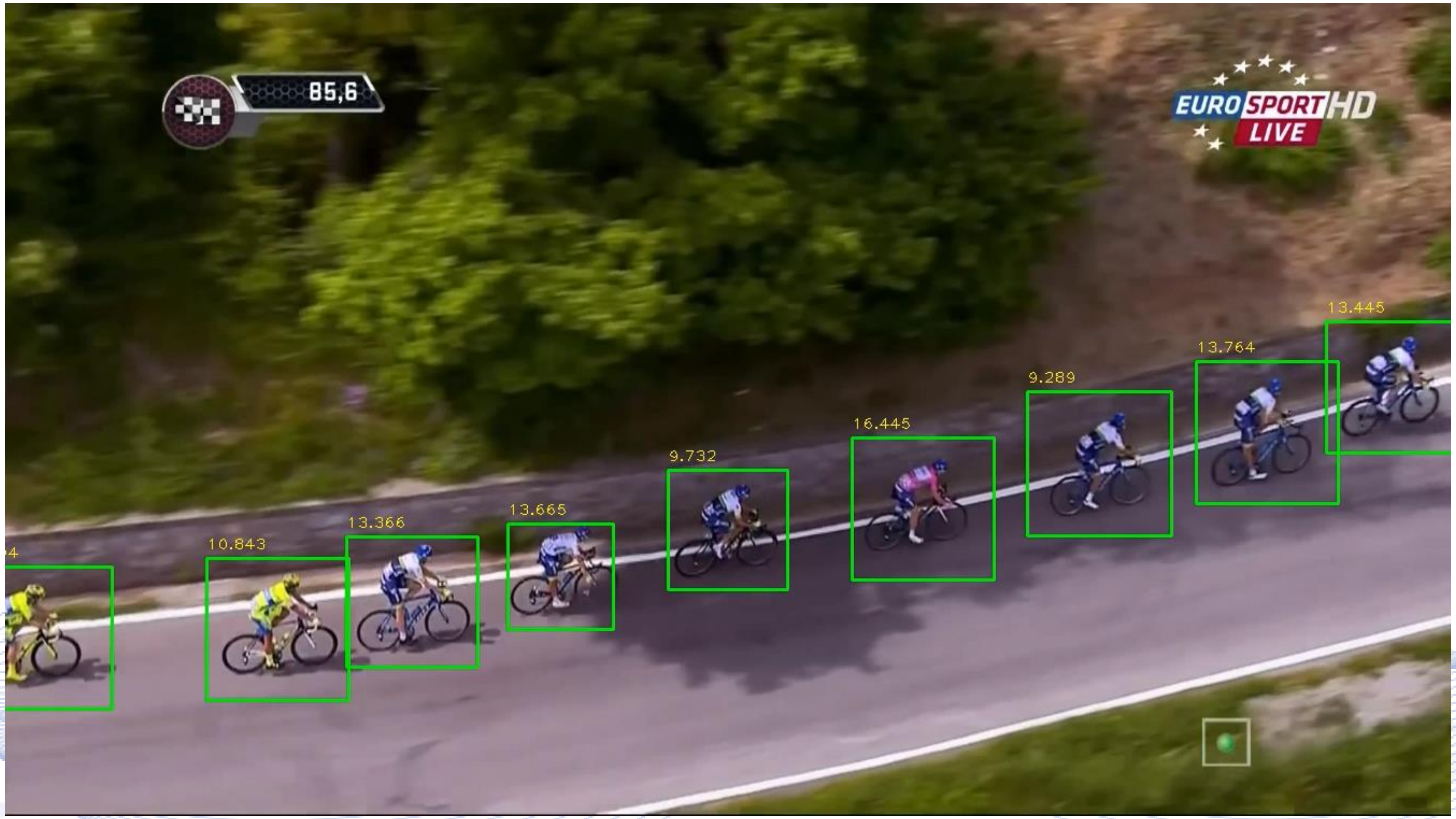


Face detection examples

Face detection examples



Object Detection for UAV sports cinematography



Bicycle detection.

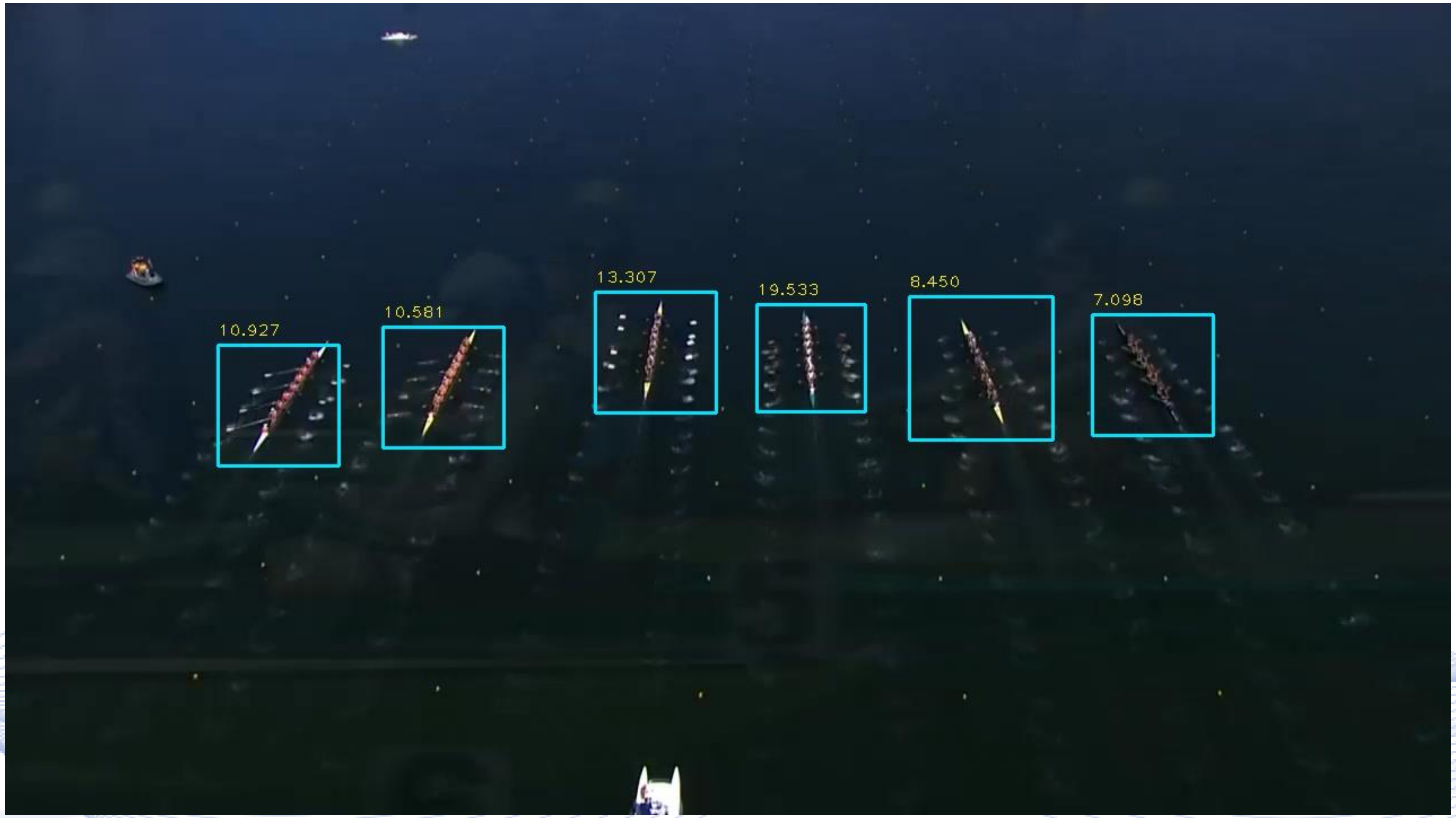
Object Detection for UAV sports cinematography

Bicycle detection.

Object Detection for UAV sports cinematography

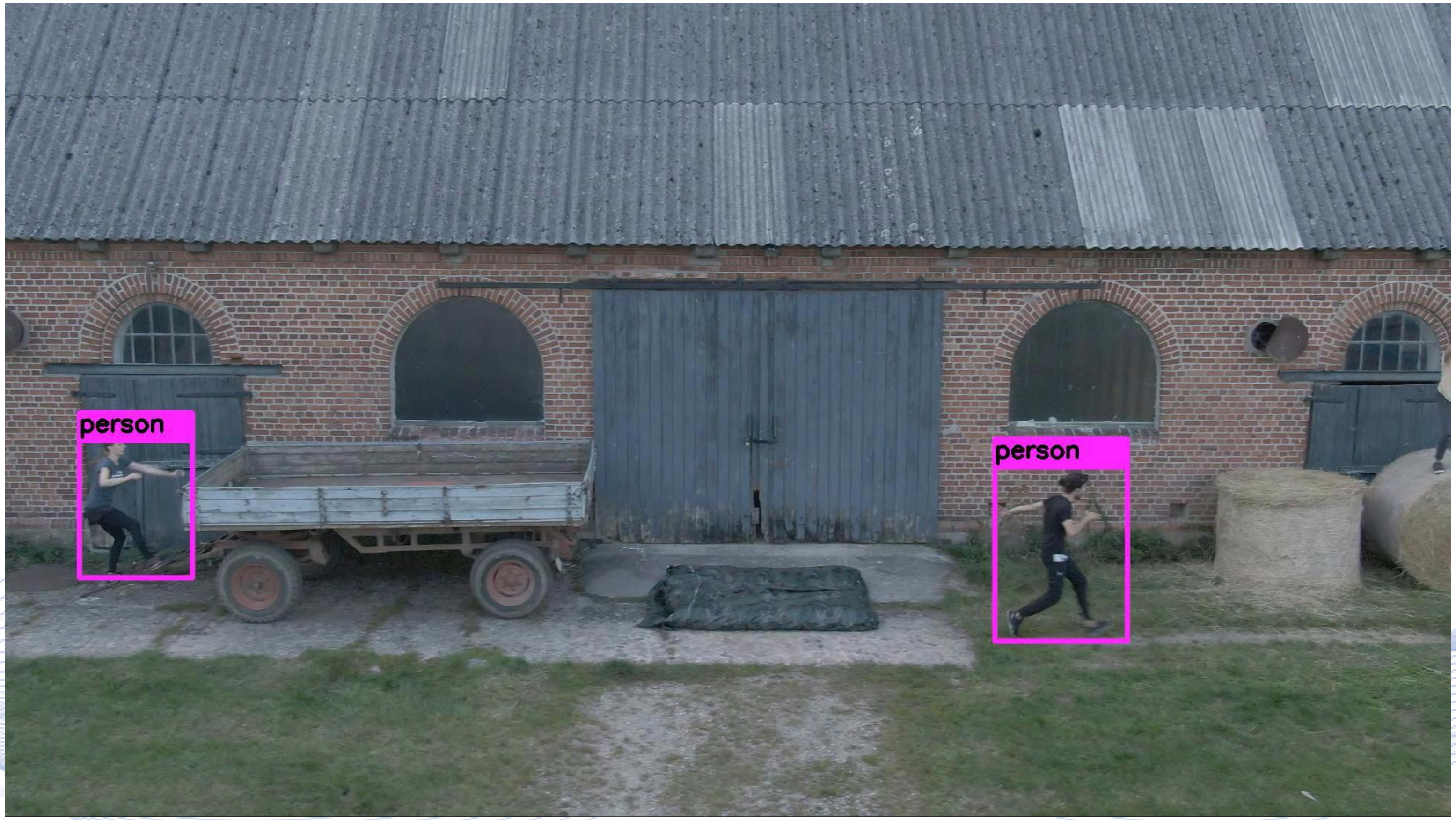
Football player detection.

Object Detection for UAV sports cinematography



Boat detection.

Object Detection for UAV sports cinematography



Parkour athlete detection.

Object Detection for UAV powerline inspection

Insulator, dumper & tower detection.

Forest Fire Detection

Real Time Detection Transformer (RT-DETR) for forest fire detection.

Pipe Defect Detection

DETR pipe defect detection.

References

[NOU2018] P. Nousi, E. Patsiouras, A. Tefas, I. Pitas, Convolutional Neural Networks for Visual Information Analysis with Limited Computing Resources, 2018 IEEE International Conference on Image Processing (ICIP), Athens, Greece, October 7-10, 2018.

[REN2015] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in Neural Information Processing Systems. 2015.

[LIU2016] Liu, Wei, et al. "SSD: Single Shot Multibox Detector." European Conference on Computer Vision. 2016.

[RED2016] J.Redmon, et al. "You only look once: Unified, real-time object detection." IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[RED2017] J.Redmon, A. Farhadi. "YOLO9000: Better, Faster, Stronger." IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[RED2018] J.Redmon, A. Farhadi, " YOLOv3: An Incremental Improvement " arxiv, 2018

[BOC2020] A. Bochkovskiy, CY Wang and HY M. Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. arXiv, 2020.

References

[SER2014] P. Sermanet et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*. 2014.

[GIR2014] R. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[GIR2015] R. Girshick, "Fast R-CNN." Computer Vision (ICCV), 2015 IEEE International Conference on. IEEE, 2015.

[HUA2017] J. Huang, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *IEEE CVPR*. 2017.

[LIN2017] T.Y. Lin, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

[LIU2018] S. Liu, D. Huang. "Receptive field block net for accurate and fast object detection." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

References

[HOW2017] A. Howard, Andrew G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.“2017.

[LAW2018] H. Law, J. Deng. “CornerNet: detecting objects as paired keypoints”. arXiv, 2019.

[DUA2019] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, Q. Tian. “CenterNet: keypoints triplets for object detection”. arXiv, 2019.

[VAS2017] A. Vaswani, Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. “Attention is all you need”. NeurIPS (2017).

[CAR2020] N. Carion, F. Massa, G. Synnaeve, N. Usanier, A. Kirillov, S. Zagoruyko. “End-to-end object detection with transformers”. arXiv, 2020.

[PYT] https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection

[PWC] https://paperswithcode.com/sota/object-detection-on-coco

[ZON2023] Z.Zong, G.Song, Y.Liu “DETRs with Collaborative Hybrid Assignments Training”. ICCV (2023).

References

[LI2022] C.Li, L.Li, H.Jiang, et al. “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications”. arXiv, 2022.

[ZHA2024] Y.Zhao, W.Lv, S.Xu et al. “DETRs Beat YOLOs on Real-time Object Detection”. arXiv, 2024.

[WAN2024] A.Wang, H.Chen, et al. “YOLOv10: Real-Time End-to-End Object Detection”. arXiv, 2024.

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr