Deep Object Detection Summary

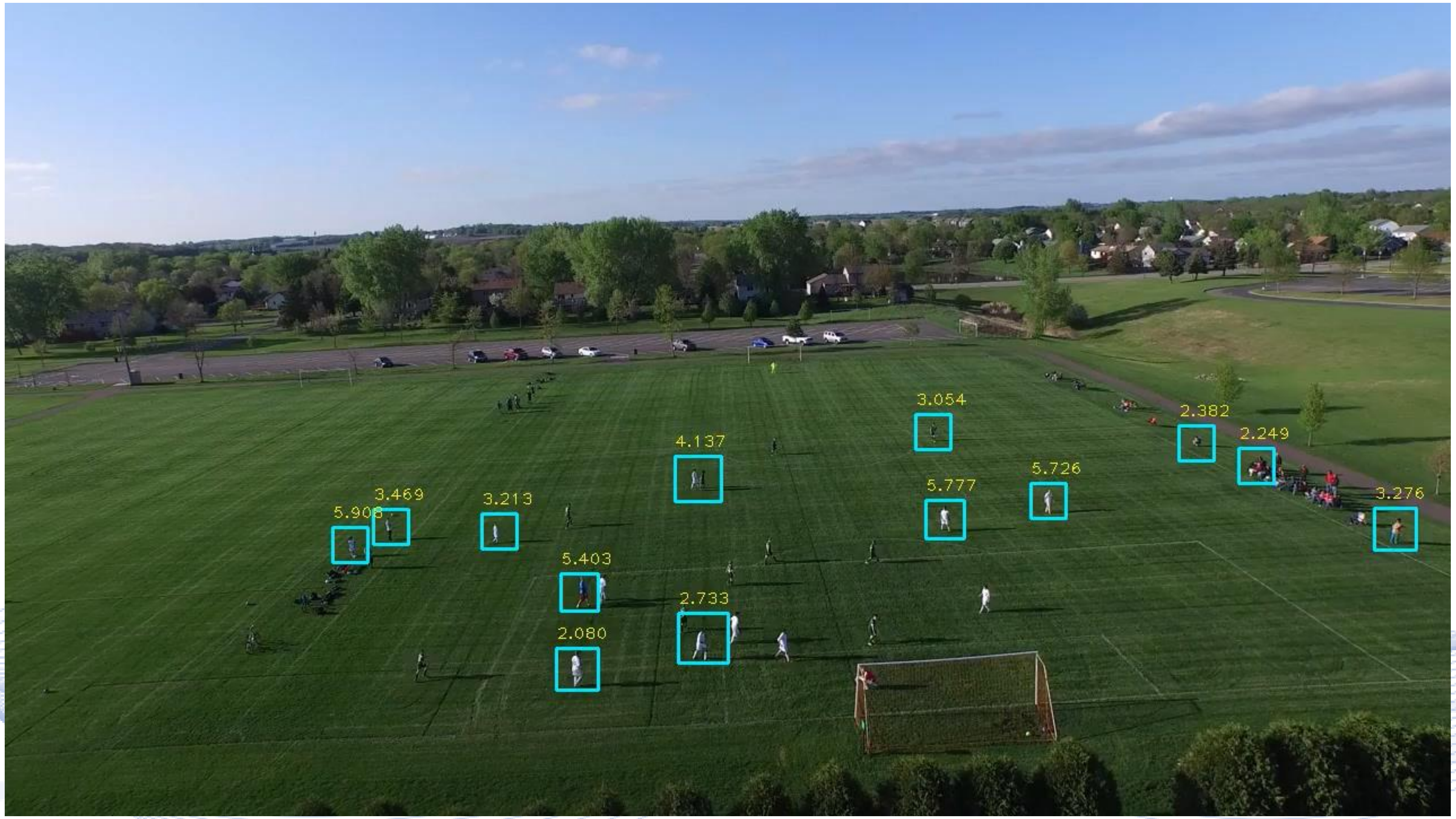
V. Nousi, D. Triantafyllidou, A. Tefas, I Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 3.2

Artificial Intelligence &
Information Analysis Lab

Object Detection

Artificial Intelligence & Information Analysis Lab

Object Detection



Object Detection

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

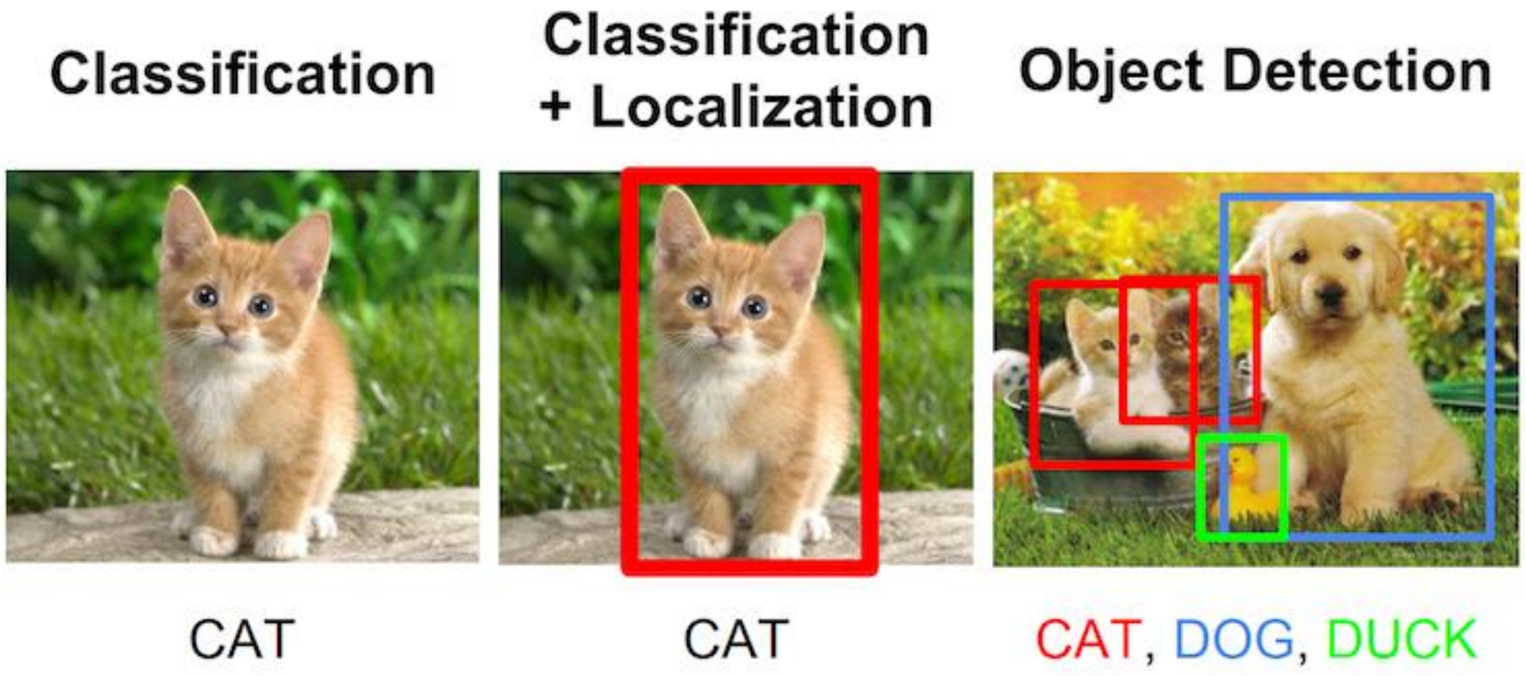


Figure: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Artificial Intelligence & Information Analysis Lab

Object Detection

- **Input**: an image.
- **Output**: **bounding boxes** containing depicted objects.
  - Each image contains a **different number of objects (outputs).**
- Typical 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, multiclass object detection.**

Object Localization Performance Metrics

•***Intersection over Union (IoU)***:

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

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

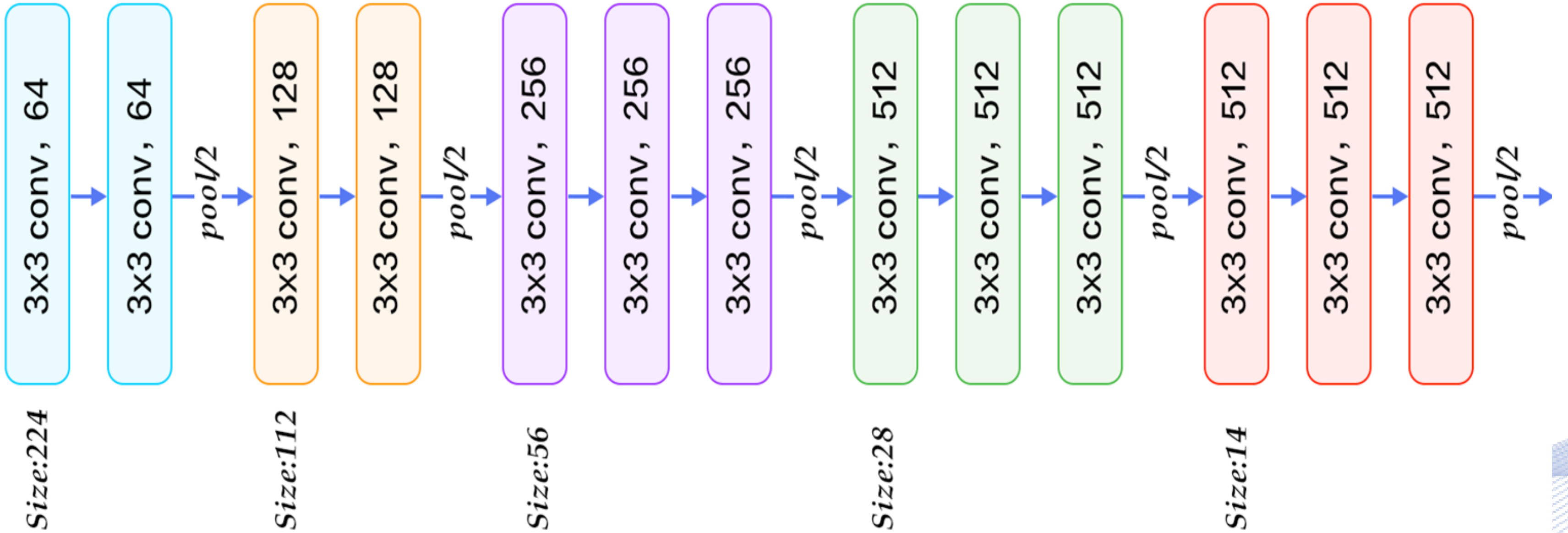
Training

- Recent approaches treat **localization as a regression problem** to find $[H, W, X, Y]$ using a CNN.
- Mixed classification + localization loss of the form:

$$\mathcal{L}(a, \mathcal{I}; \theta) = \beta_1 * \mathbf{1}[a \text{ is positive}] * \ell_{loc}(\phi(b_a; a), f_{loc}(\mathcal{I}; a, \theta)) + \\ + \beta_2 * \ell_{cls}(y_a, f_{cls}(\mathcal{I}; a, \theta))$$

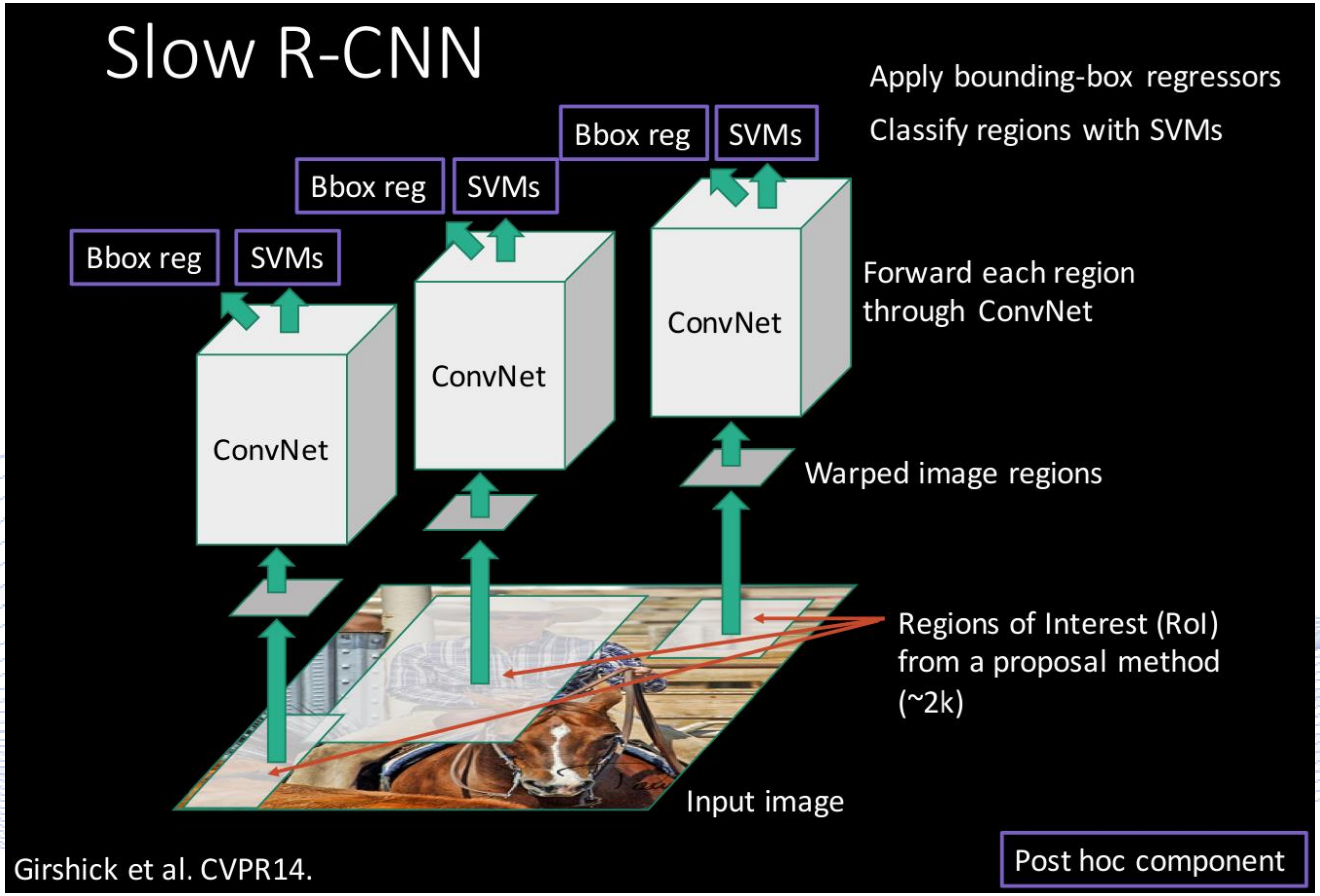
  - β_1 and β_2 balance the localization and classification losses.
    - a is the best matching ground truth ROI (anchor box) for the detected ROI (box) b_a.
    - $\mathbf{1}[\alpha \text{ is positive}]$: indicator vector (vector of ones, if α matches b_a with good IoU).
    - f_{loc} is the localization CNN function, f_{cls} is the classification CNN function.
  - ℓ_{loc}, ℓ_{cls} are loss functions, e.g., MSE, cross-entropy.

Object Detection with CNNs



Object detection: CNN pipeline for bounding box regression.

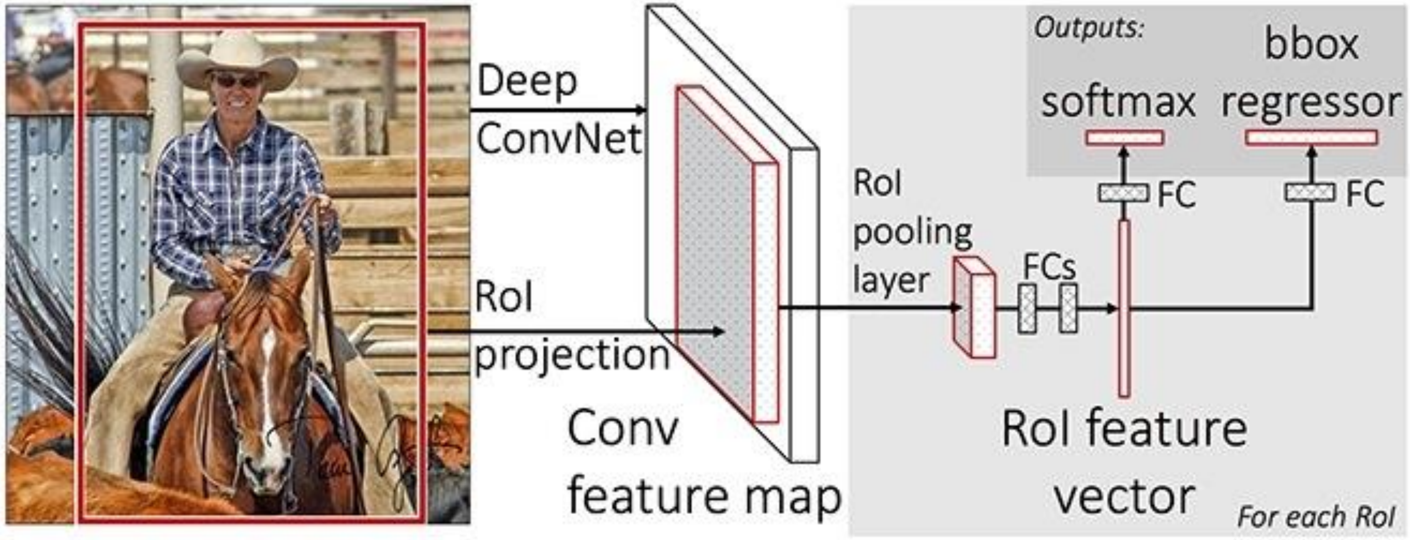
R-CNN



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

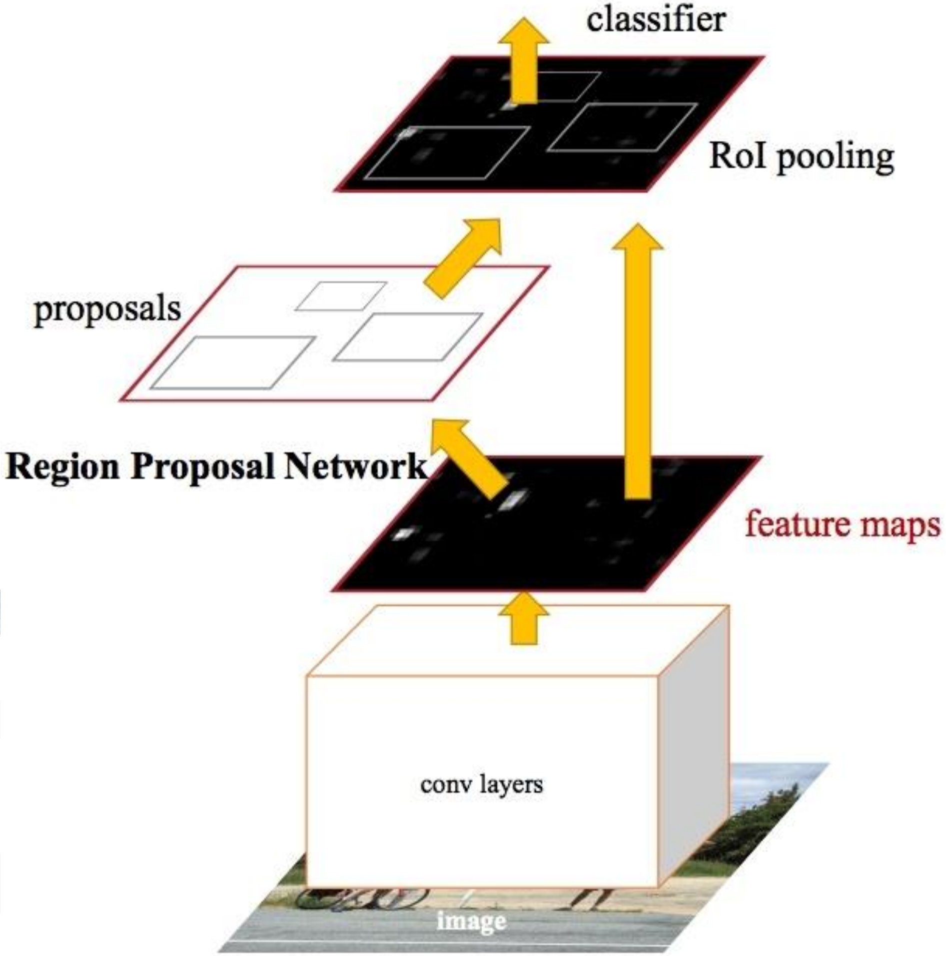
Fast R-CNN

- Fast R-CNN **weaknesses**:
  - Multiple **overlapping RoIs**
    - duplicate computations.
  - **Externally** computed region proposals (selective search).



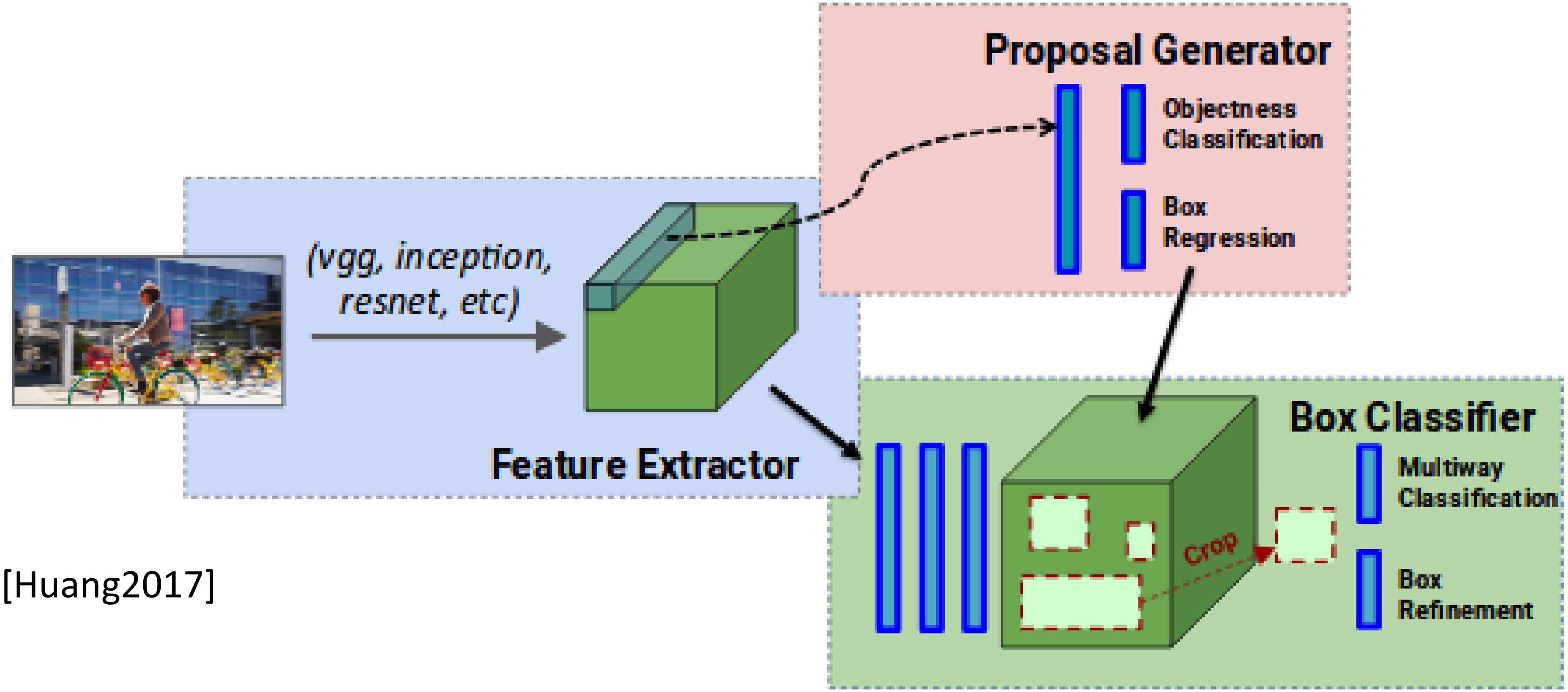
Girshick, Ross. "Fast R-CNN." *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015.

Faster R-CNN



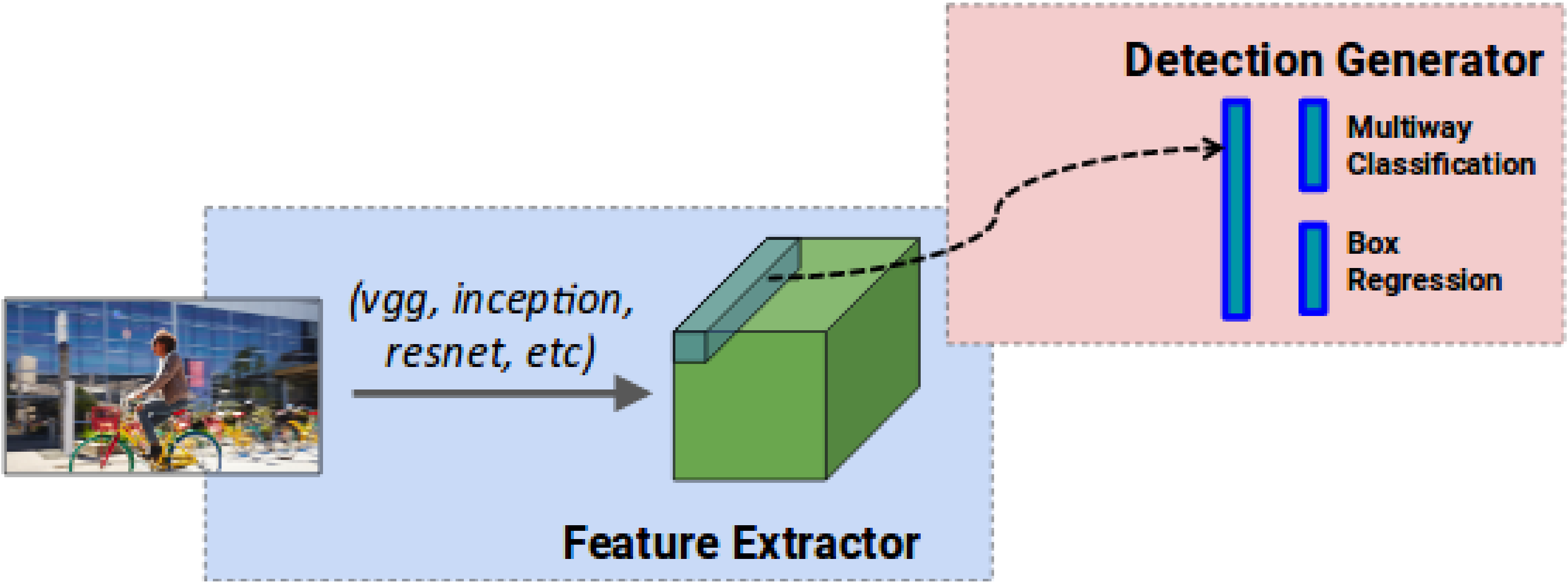
- Faster R-CNN [Ren2015]: The **Region Proposal Network** shares layers with the feature extracting network and **internally produces region proposals** (no selective search).

R-FCN



[Huang2017]

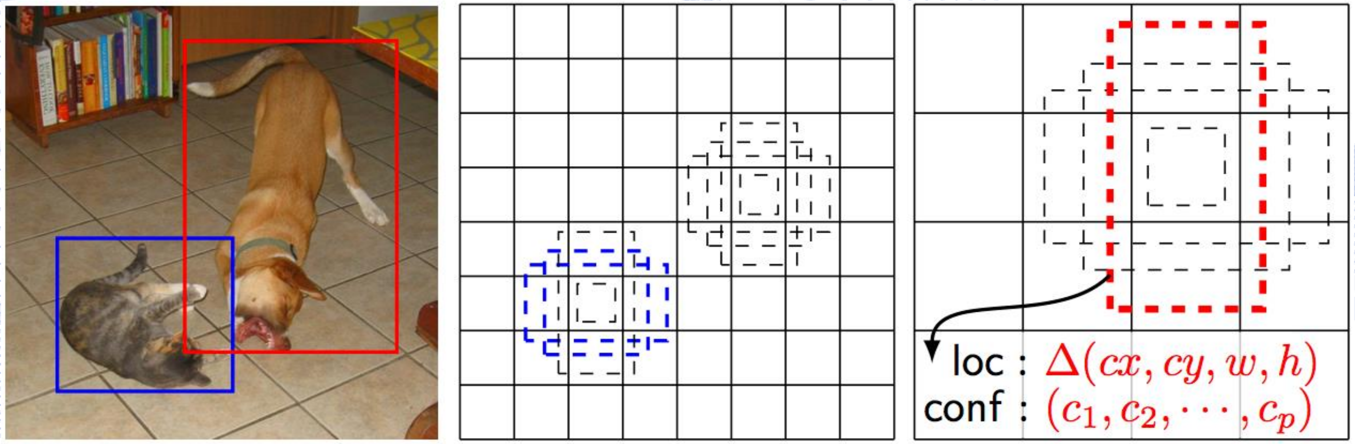
SSD



[Huang2017]

SSD

- Example: The cat has 2 anchors (ROIs) that match on the 8×8 feature map, but none match the dog. We choose the one having biggest IoU and refine it.
- On the 4×4 feature map there is one anchor that matches the dog and is refined.

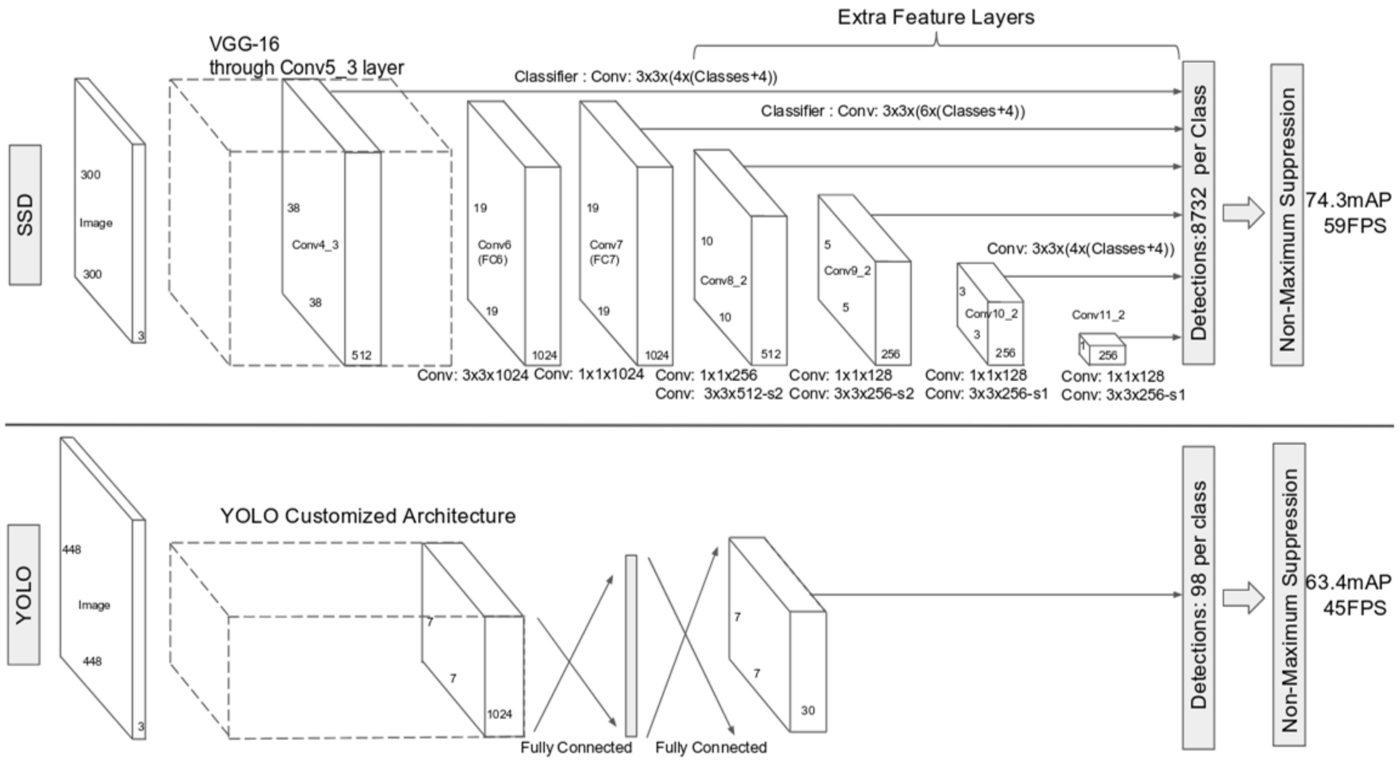


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

[Liu2016]

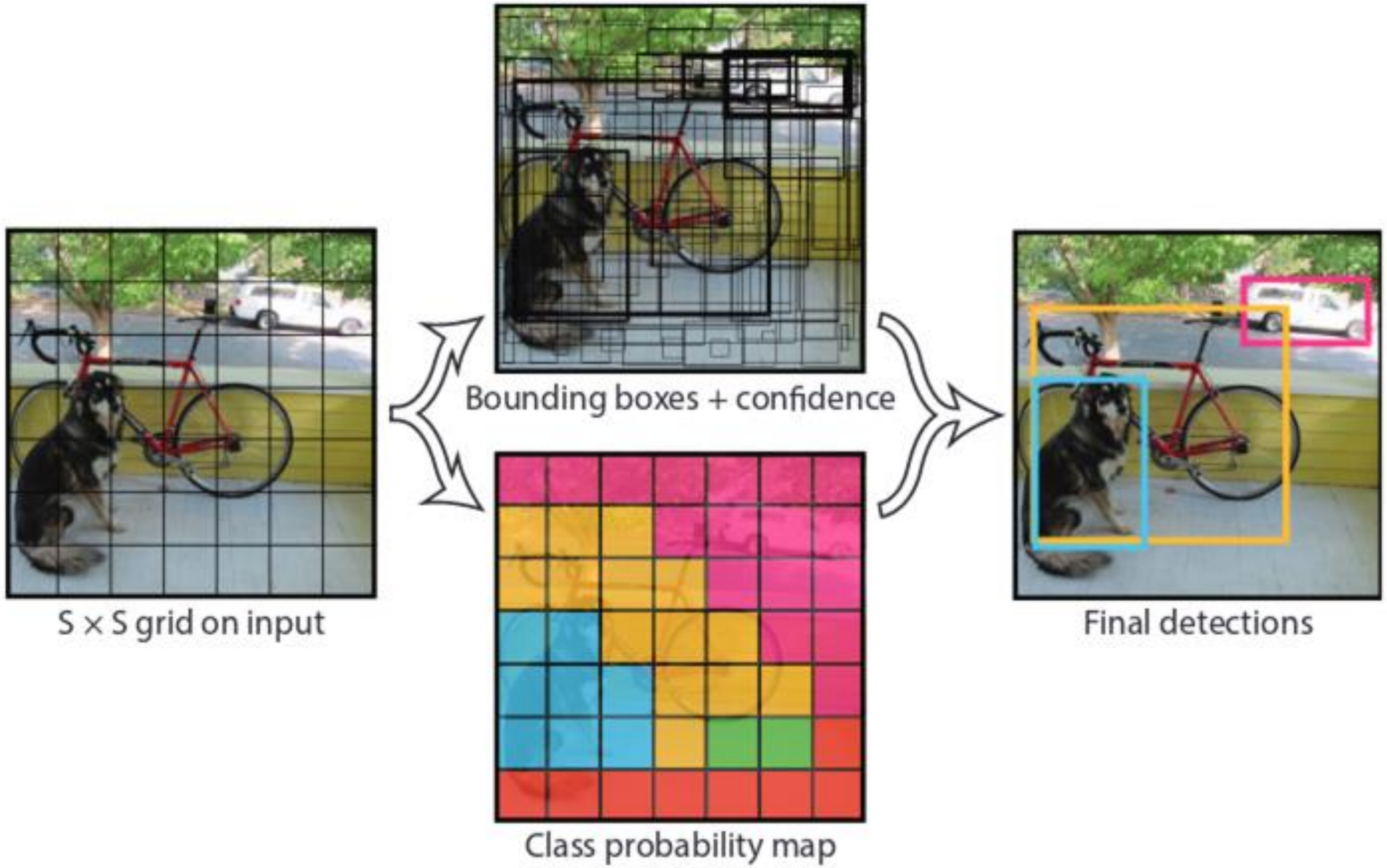
YOLO

- **Simpler YOLO architecture:** Darkenet19 convolutional network plus FC layer.
- Prediction only at the final convolutional feature map.



[Liu2016]

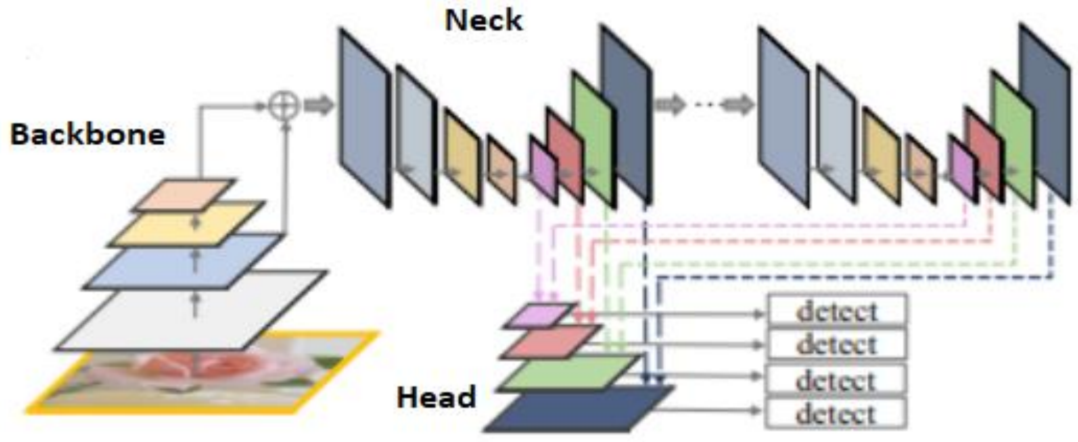
YOLO



[Redmon2016]

YOLO v4

YOLO v4 design:

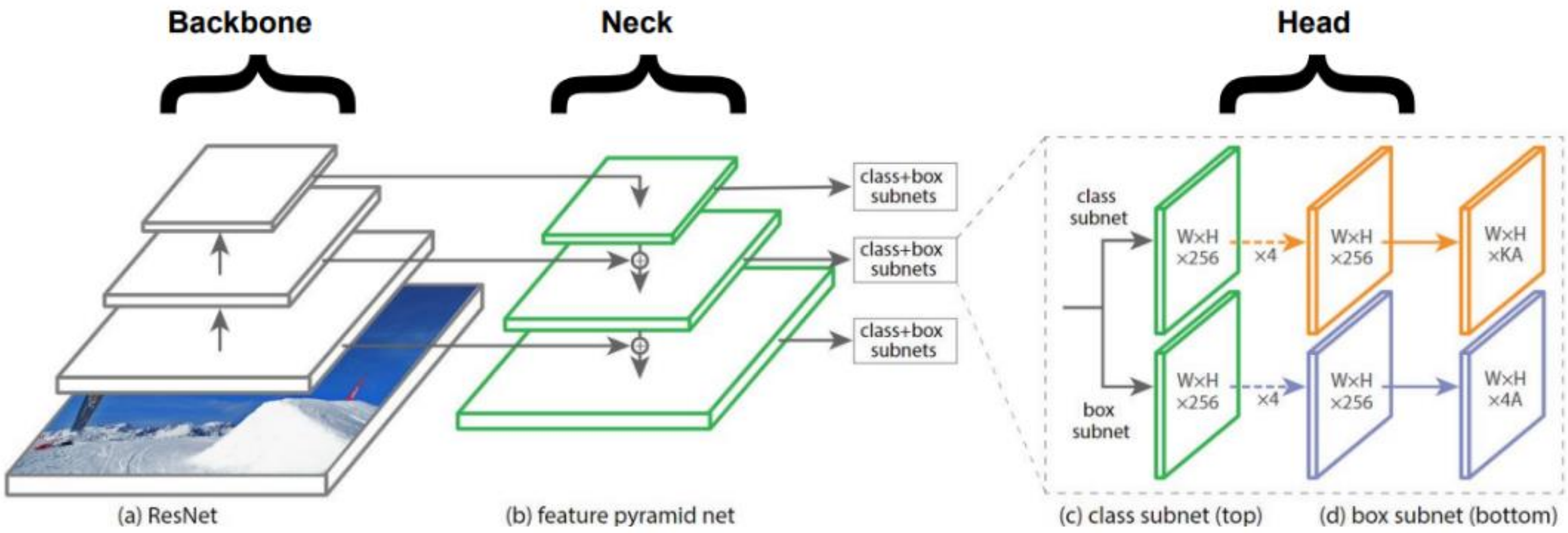


[Bochkovskiy2020]

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

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.



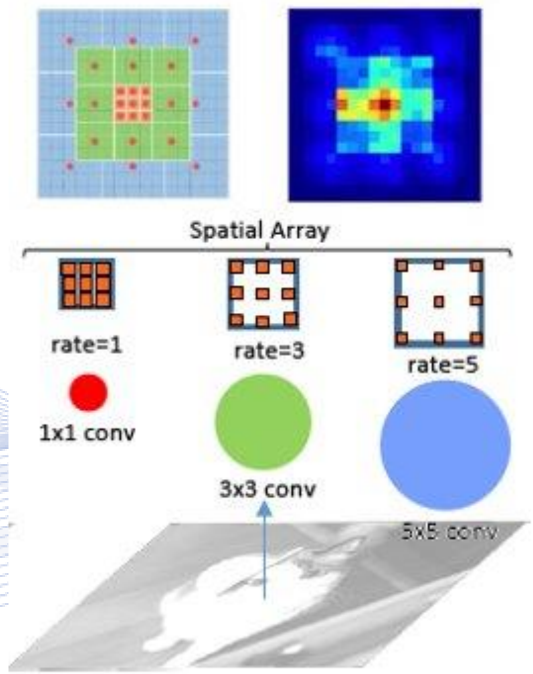
[Lin2017]

RFBNet

- Architecture inspired by the structure of ***Receptive Fields in human visual systems*** [Liu2018].

- Use of multiple dilated convolutions with different kernel sizes in each convolutional layer.

- State-of-the-art results and fast inference time.



Using object detectors for drone-based shooting

- **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 Detection Performance Metrics

Top-5 Classification Error:

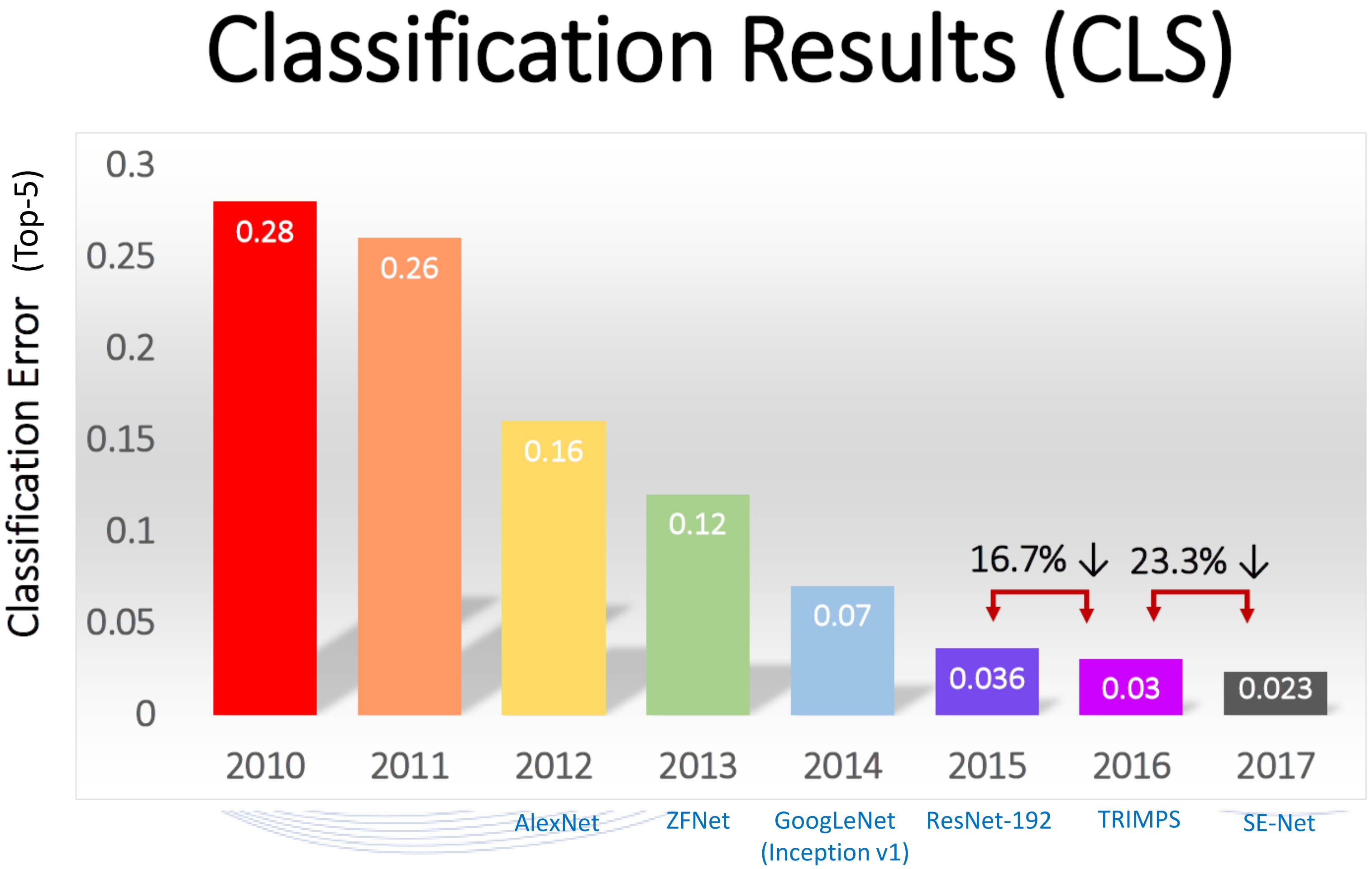
- Given the ground truth object class label $\mathcal{C}_i$ and top 5 predicted class labels $\mathcal{C}_{i1}, \dots, \mathcal{C}_{i5}$ the prediction is correct, if $\mathcal{C}_{ij} = \mathcal{C}_i, j = 1, \dots, 5$. The error of a single prediction is:

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

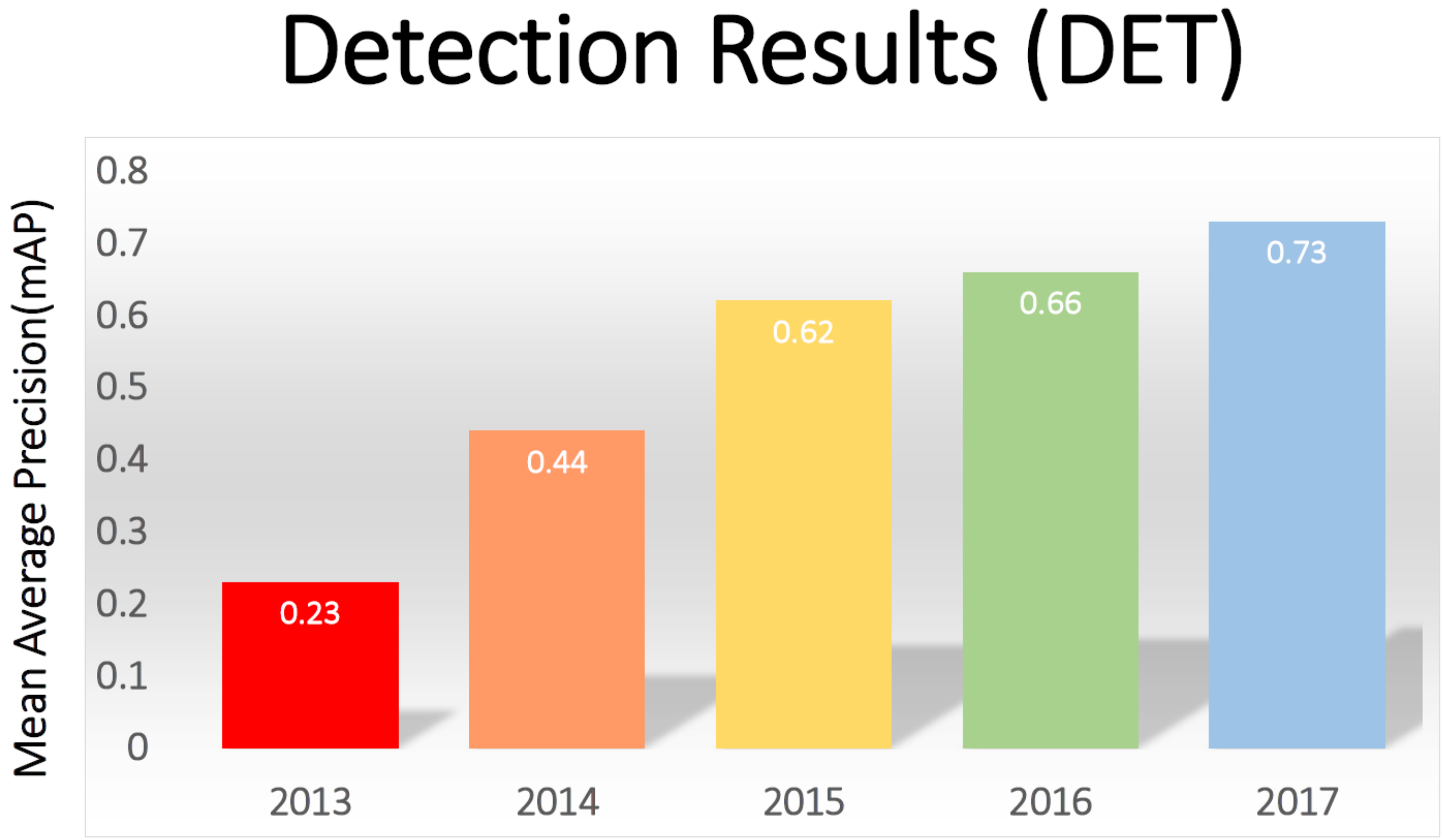
- The top-5 error is the fraction of N_t test images on which the prediction is wrong:

$$top5error_{CLS} = \frac{1}{N_t} \sum_{i=1}^{N_t} \min_j \{e_{CLS}(c_{ij}, C_i)\}, \qquad j = 1, \dots, 5.$$

Object Detection Performance Metrics

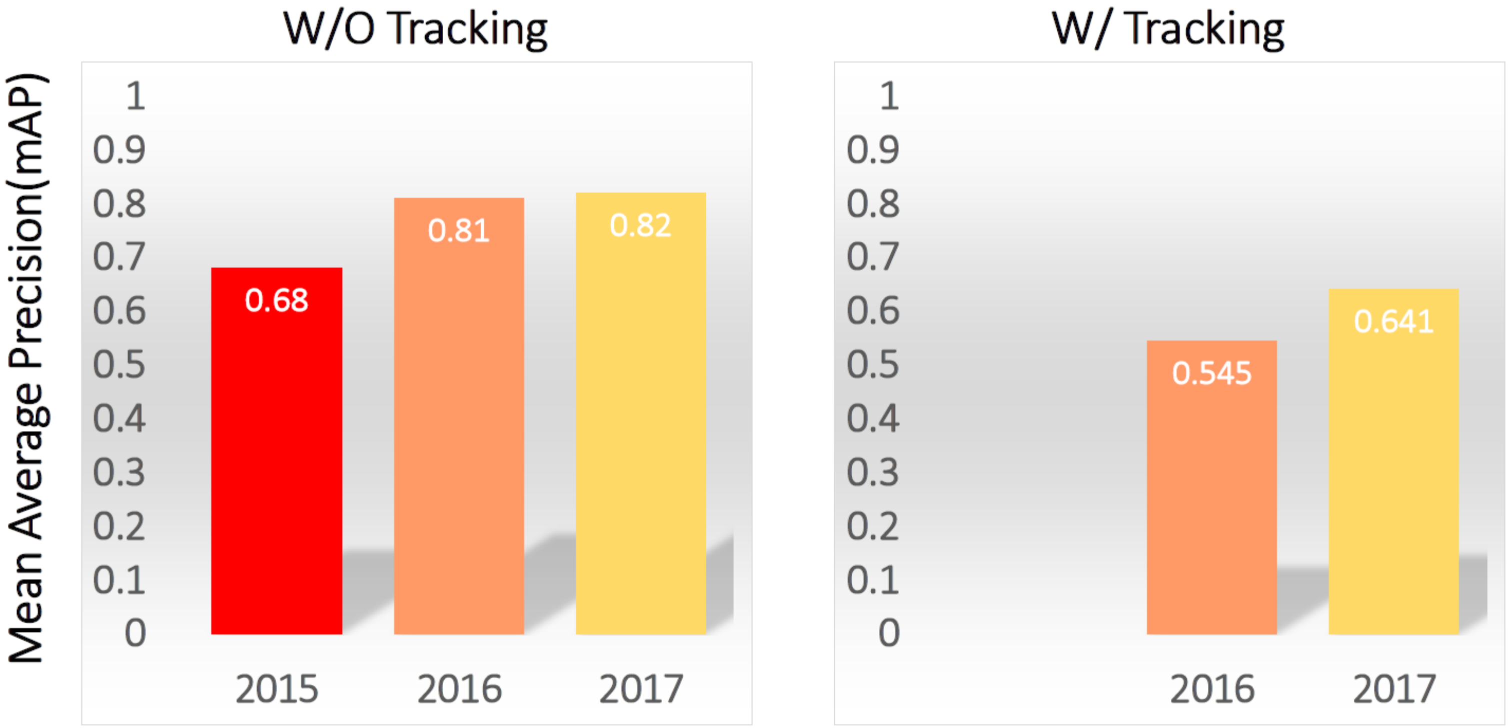


Object Detection Performance Metrics



Object Detection Performance Metrics

Video Detection Results (VID)



Object Detection Performance Metrics

Top-5 Localization Error:

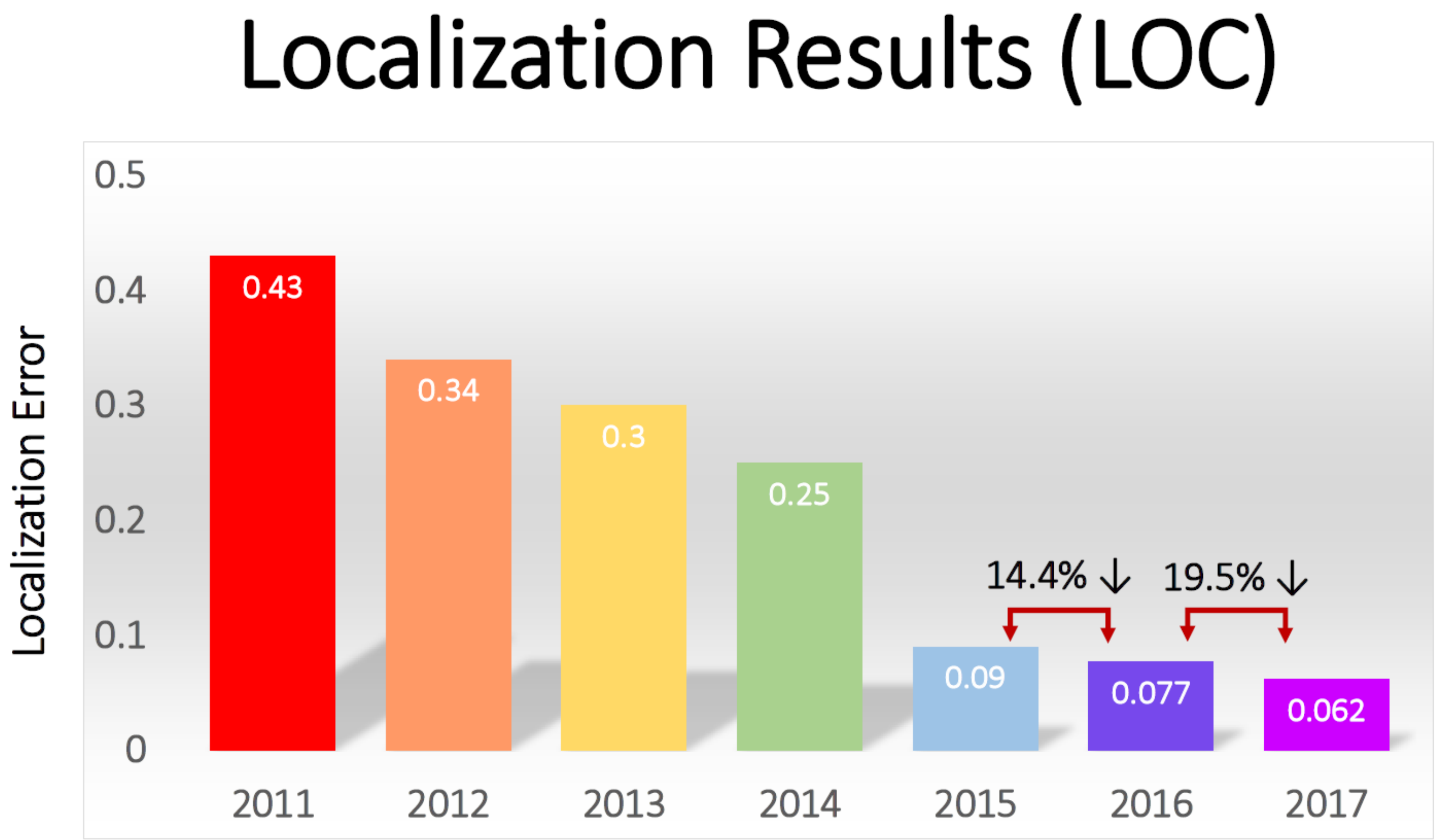
For each test image $i = 1, \dots, N_t$, let us have:

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

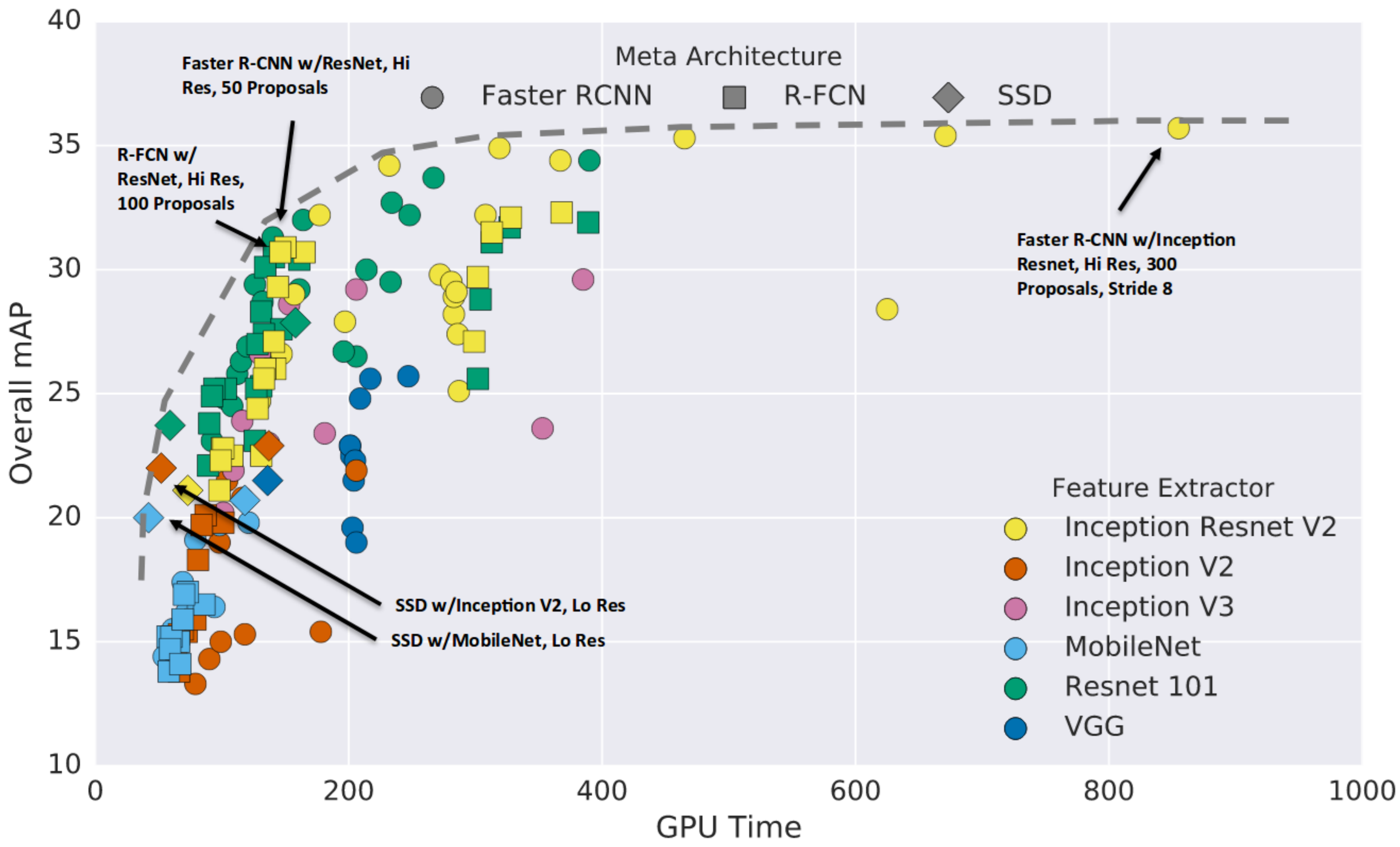
$$e_{LOC}(\mathcal{A}_{ij}, \mathcal{B}_{ik}) = \begin{cases} 1, & J(\mathcal{A}_{ij}, \mathcal{B}_{ik}) \leq 0.5 \\ 0, & J(\mathcal{A}_{ij}, \mathcal{B}_{ik}) > 0.5 \end{cases},$$

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

Object Detection Performance Metrics



CNN comparison



[Huang2017]

CNN comparison

- **Faster R-CNN is more accurate but slower.**
- **YOLO, SSD are much faster** but not as accurate.
- YOLO, SSD make **more mistakes when objects are small** and have trouble correctly predicting the exact location of such objects.

Object detection acceleration

- Examples of acceleration techniques:
  - Input size reduction.
  - Specific object detection instead of multi-object detection.
  - Parameter reduction.
  - Post-training optimizations with TensorRT (NVIDIA), including FP16 (floating point 16 bit) computations.

Object Detection on NVIDIA Jetson TX2

- YOLO: good precision in general, but too heavyweight:
  - small objects are more challenging.
- Evaluation on VOC:

Input Image Size	FPS	mAP	Forward time (ms) No TensorRT	Forward time (ms) TensorRT	Forward time (ms) FP16
608x608	2.9	71.26	241.5	128.8	69.3
544x544	3.2	73.64	214.4	121.2	64.3
480x480	5.4	74.50	155.4	62.3	35.7
416x416	6.4	73.38	155.3	56.5	32.5
352x352	7.8	71.33	111.0	45.0	24.3
320x320	8.5	70.02	103.0	40.4	22.8

Object Detection on NVIDIA Jetson TX2

- SSD: generally good precision, not as fast, still prone to mistakes when objects are small.

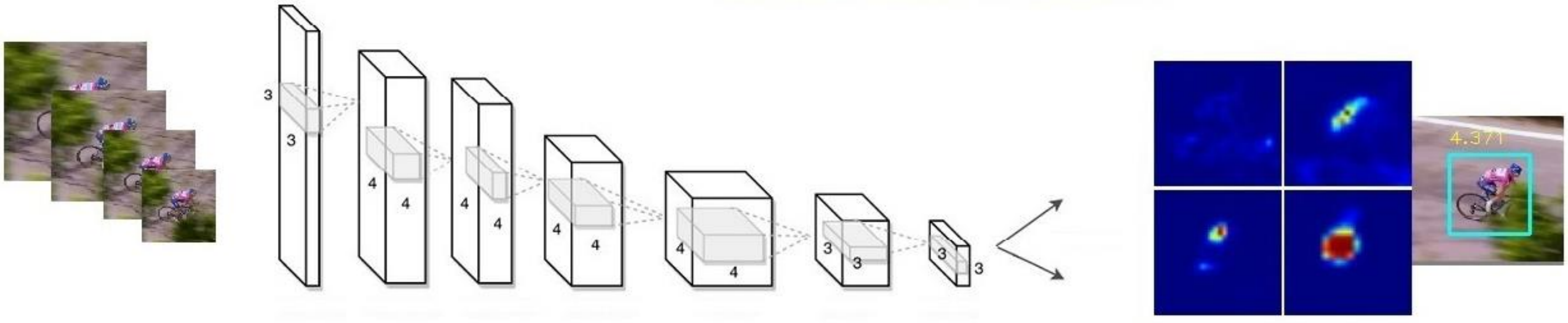
- MobileNets and Inception V2 can be used as feature extractors to provide speed ups.

Object detection

- **State-of-the-art** object detectors (YOLO, SSD, etc) are based on **very Deep** and **multiple-channel CNNs.**
- **Lightweight** architectures can provide equally satisfactory results.
- Such architectures are trained with incremental positive and negative example mining methods.

Object detection

- **Detection with Light weight deep CNNs.**
- The method develops an image pyramid representation of varying resolutions and performs detection at multiple scales.

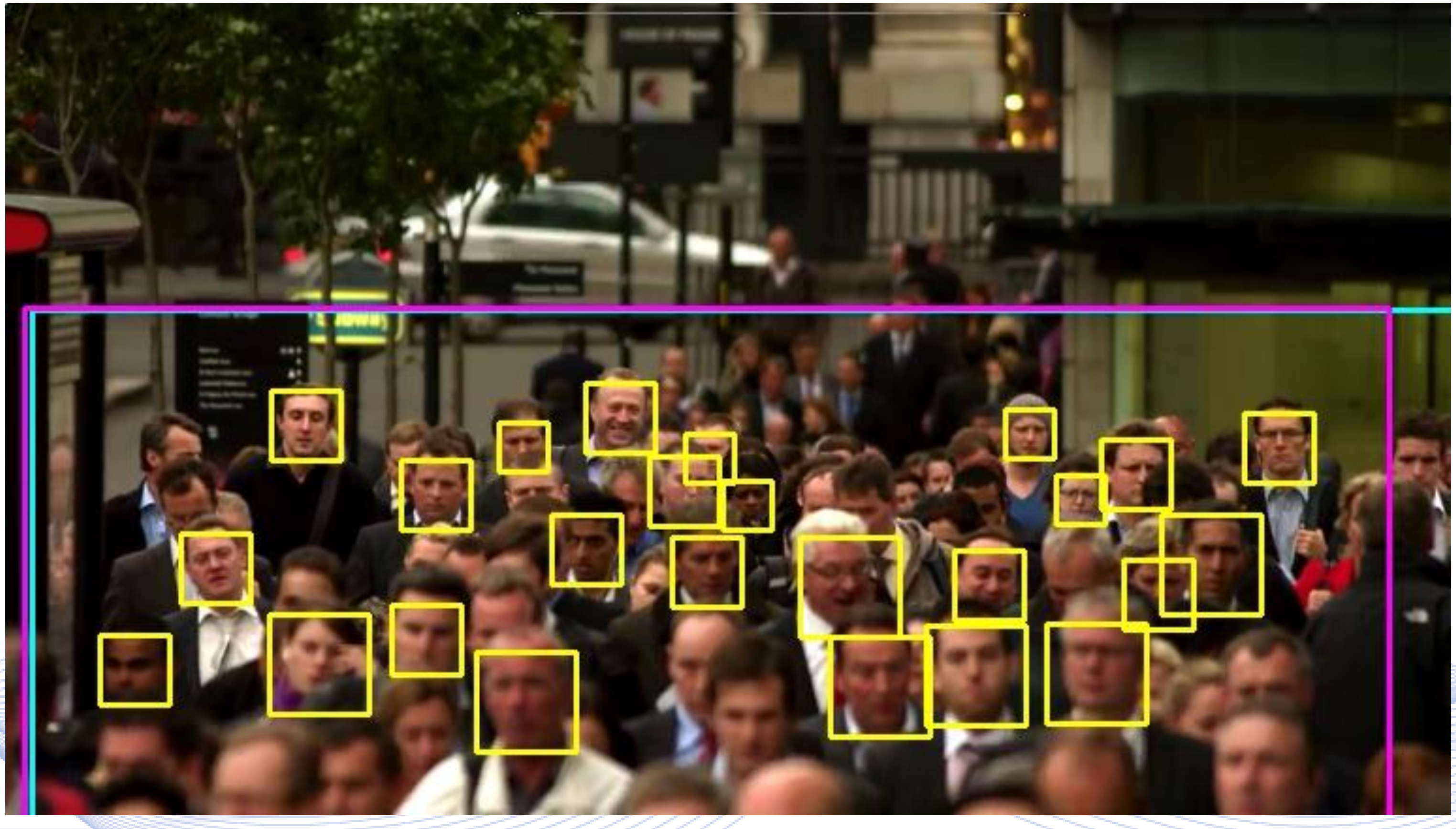


Object detection

Test execution time for a 32×32 pixel object ROI of the proposed CNN architecture using NVIDIA's tensorRT library (in msec):

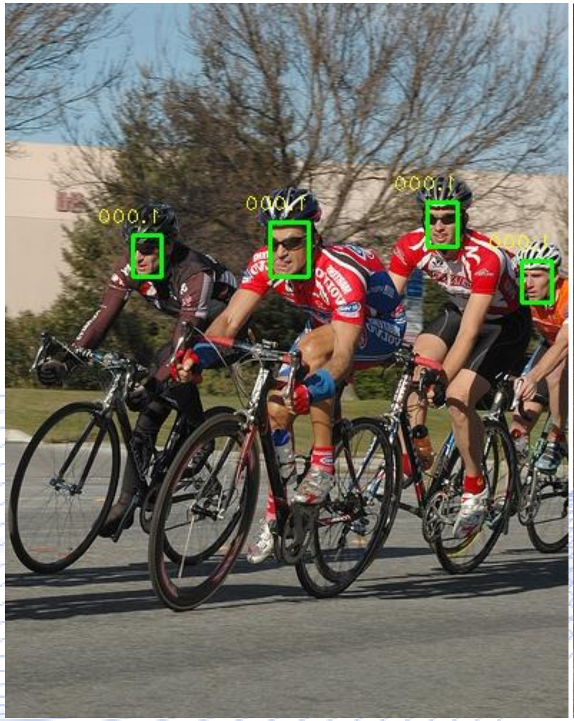
NVIDIA TX2		geForce GTX 1080
tensorRT	no tensorRT	
0.491933	2.84615	0.652406

Face detection examples

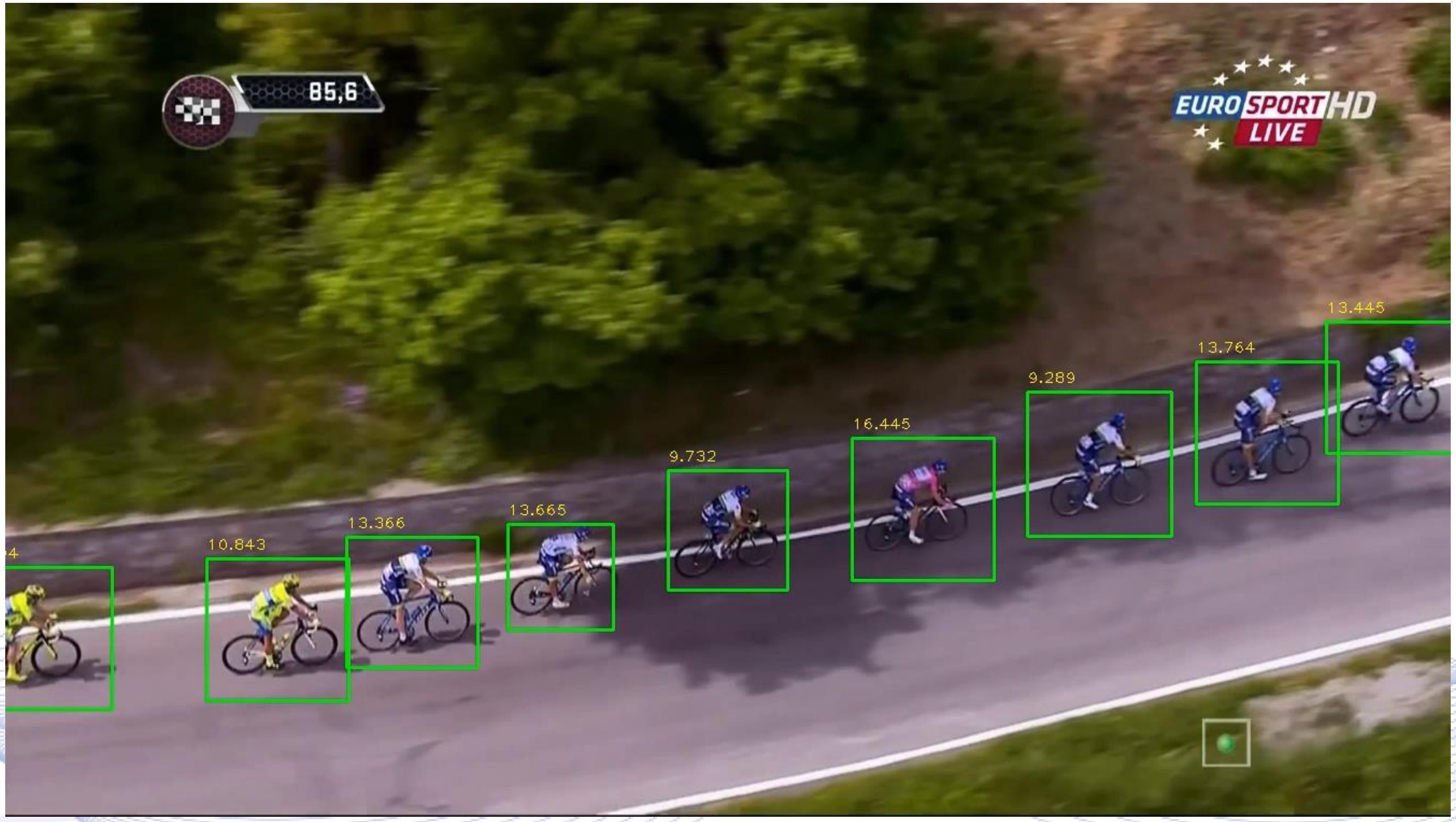


Face detection examples

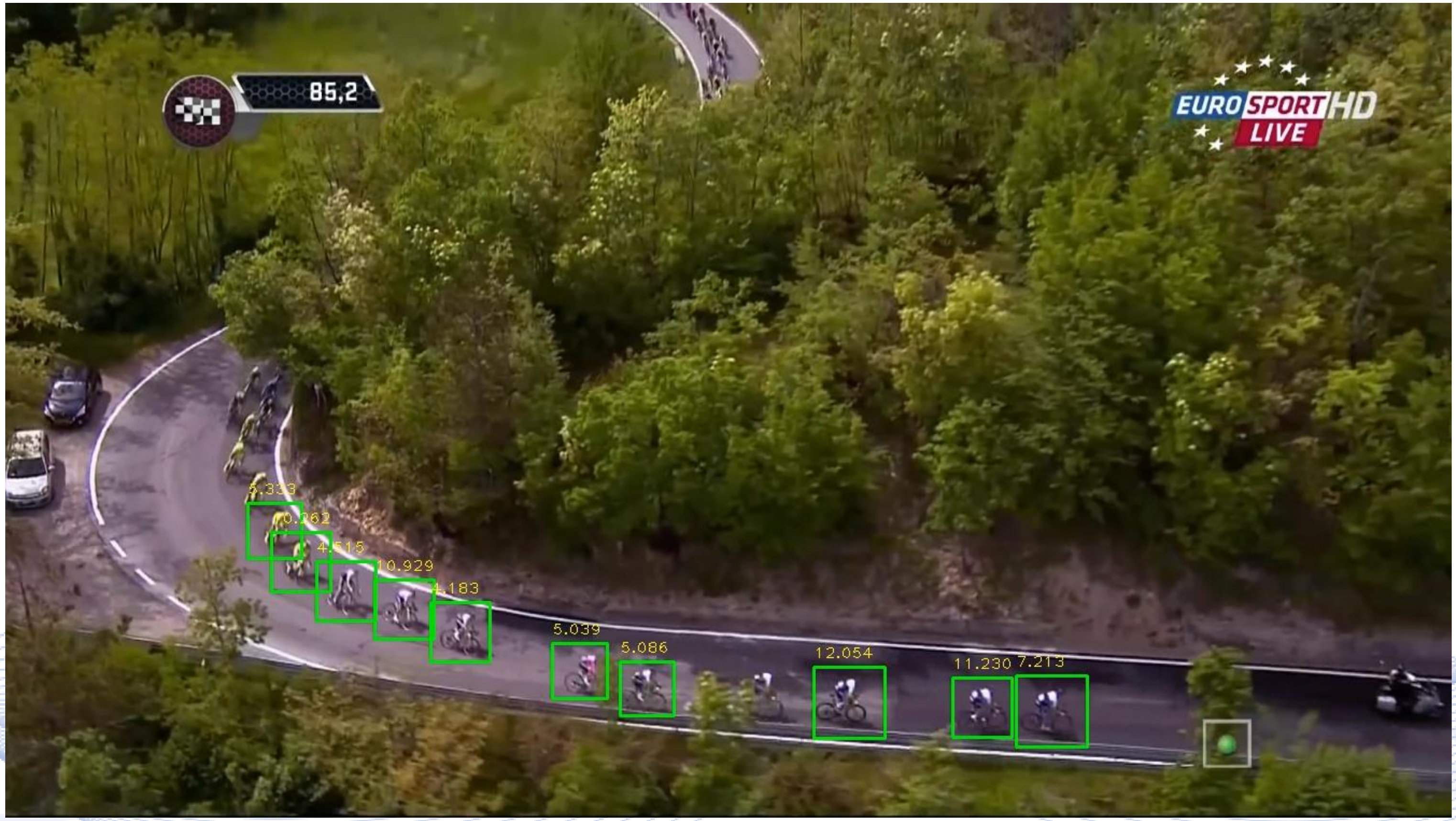
Face detection examples


Bicycle detection

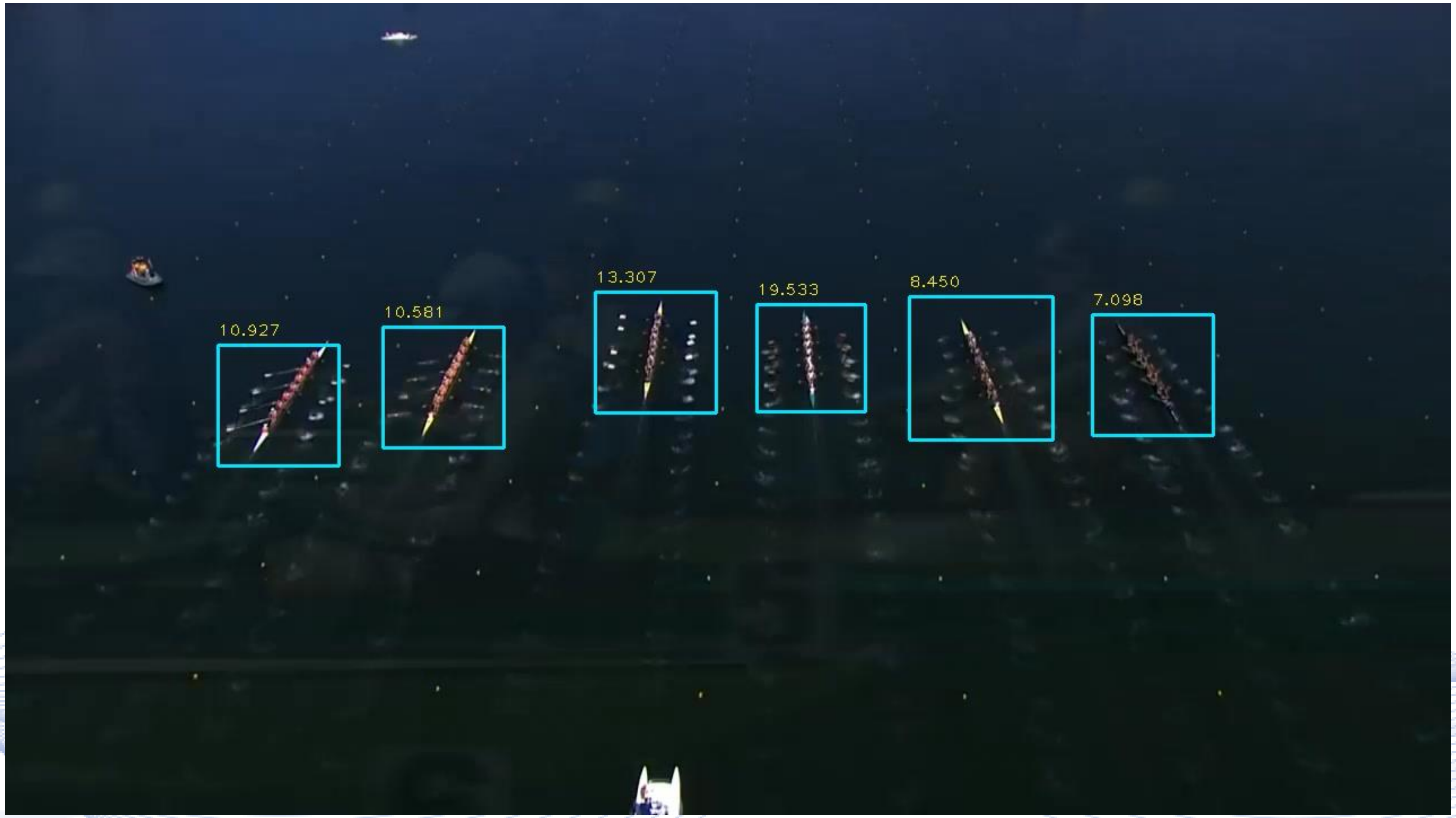


Bicycle detection



Football player detection

Boat detection



Q & A

Thank you very much for your attention!

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