Object Detection

for Autonomous Driving

Arjun Jain

4th Summer School on Computer Vision, IIIT Hyderabad

2nd July 2019

Location: Hiranandani, Powai

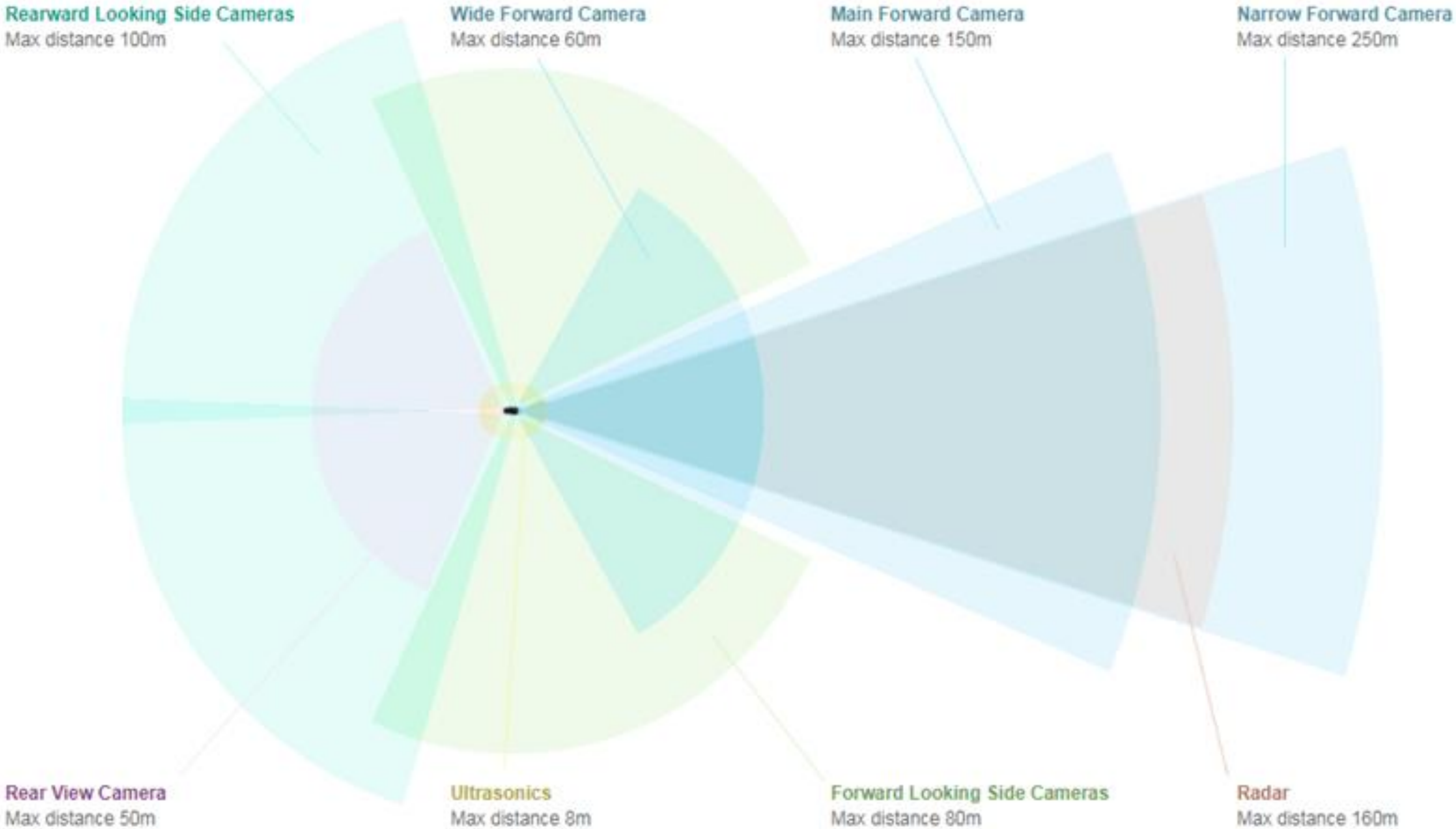
Computer Vision for Autonomous Cars

Existing roads are made for agents that can *see in the visible spectrum*

- *Lane markings*
- *Traffic lights*
- *Traffic signs*
- *Intention and Gestures of traffic policemen, construction workers*
- *Path prediction of pedestrians*
- *Non verbal clues of other drivers*

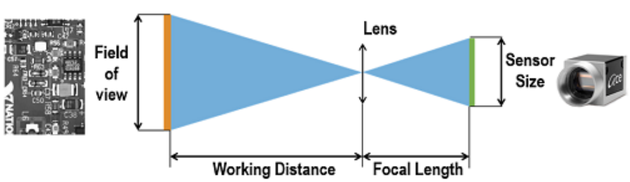
Computer vision is thus the most important modality for perception

Tesla Camera Stack



System Design Specs

- Urban speed limit: 50 kmph ~ 14m/s
- Highway speed limit: 100 kmph ~ 28m/s
- Comfortable deceleration < 2.5m/s^2
- Max emergency deceleration = 5m/s^2
- Time to complete stop after hitting breaks = 28/5 = 5.6s for highway
- Brake system latency 300ms
- Vision + Fusion + Tracking time = 300ms
- $S = 0.6 * 28 + (v^2 - u^2)/2a = 95.2m$



For a typical camera (2MP, focal length=2000px), the image pedestrian 1.75 meters tall and is 95.2m away is 35px tall in the image

System Design Specs

- Vision + Fusion + Tracking time = 300ms
- Tracking needs 3 cycles of Vision + Fusion for high certainty
- Fusion needs confirmation from at least 2 sensors
- 100 ms for Vision + Fusion → < 50ms for vision
- For 8 cameras in parallel, 1 system

This is hard engineering

Problem

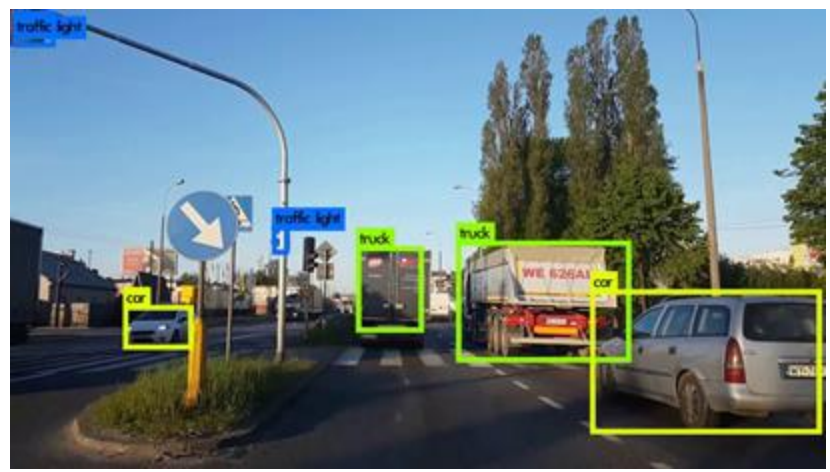
From **Wikipedia** The Free Encyclopedia

Detecting instances of semantic **objects** of a certain class (such as humans, buildings, or cars) in digital images and videos

Practically:

The task of assigning a label and a bounding box to all objects in an image

Inside a refrigerator



On the street

Tesla Autopilot Overlay

Data: The Oil Driving Object Detection Research

COCO
Common Objects in Context

200,000 images and 80 object categories

IMAGENET

Detection: 500,000 images and 200 object categories

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

11,530 and 20 categories

Google Open Images v4

15,440,132 boxes on 600 categories

Computer Vision Tasks

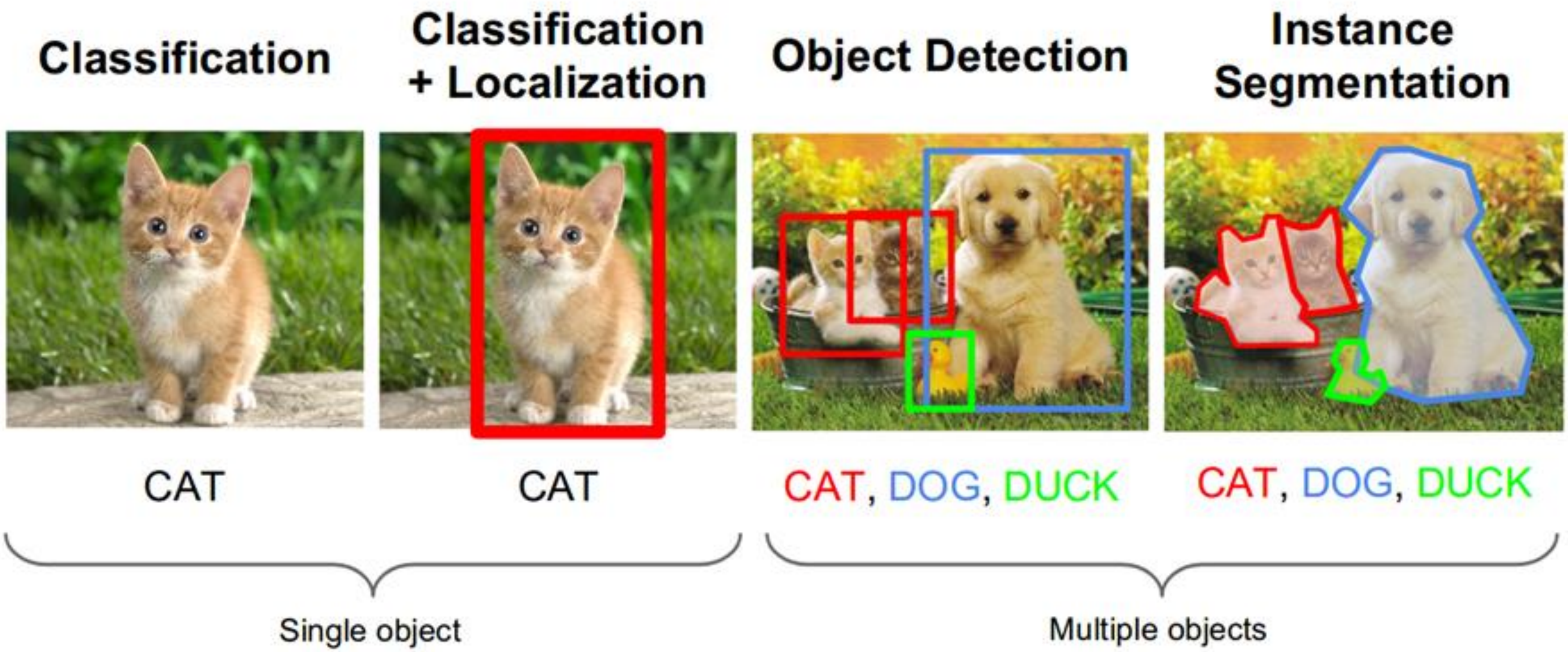
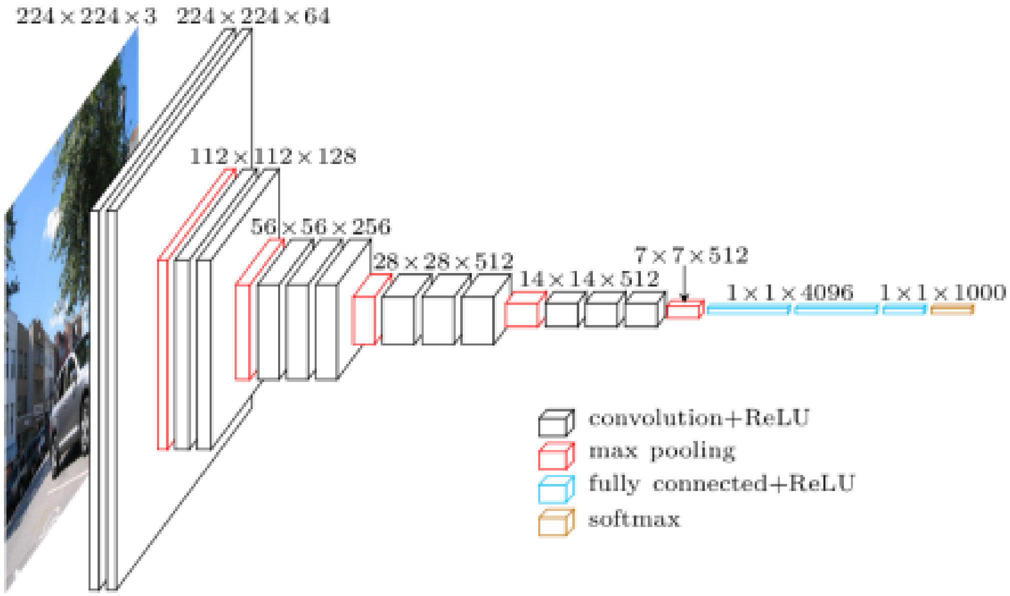


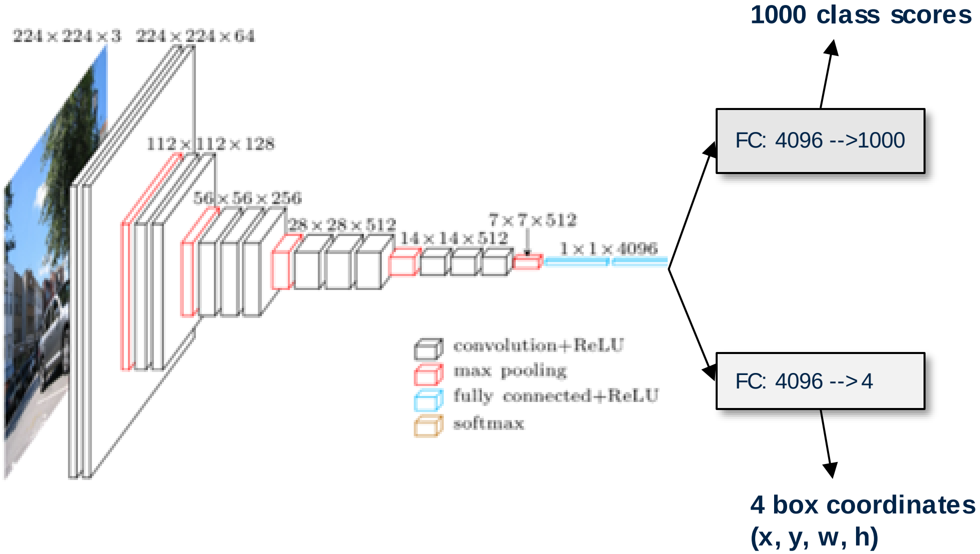
Image: CS231 Lecture Notes, Stanford University

Classification

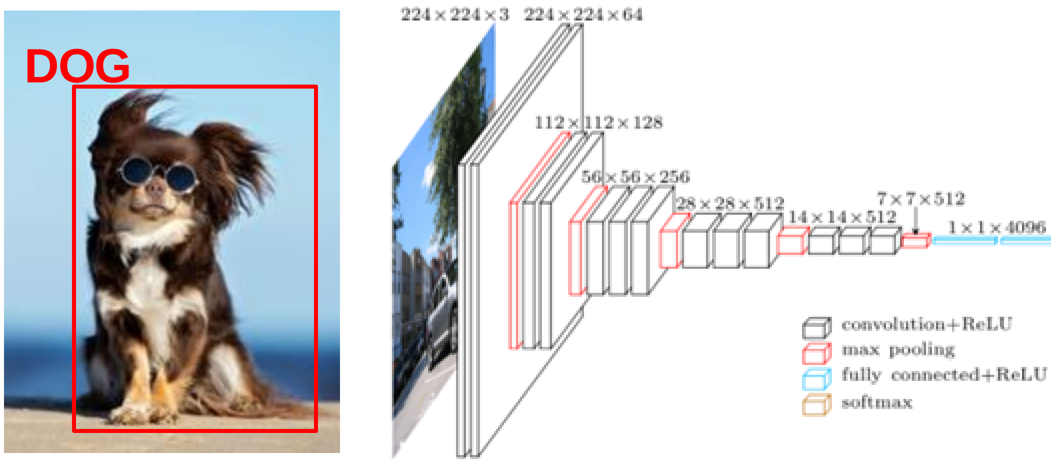


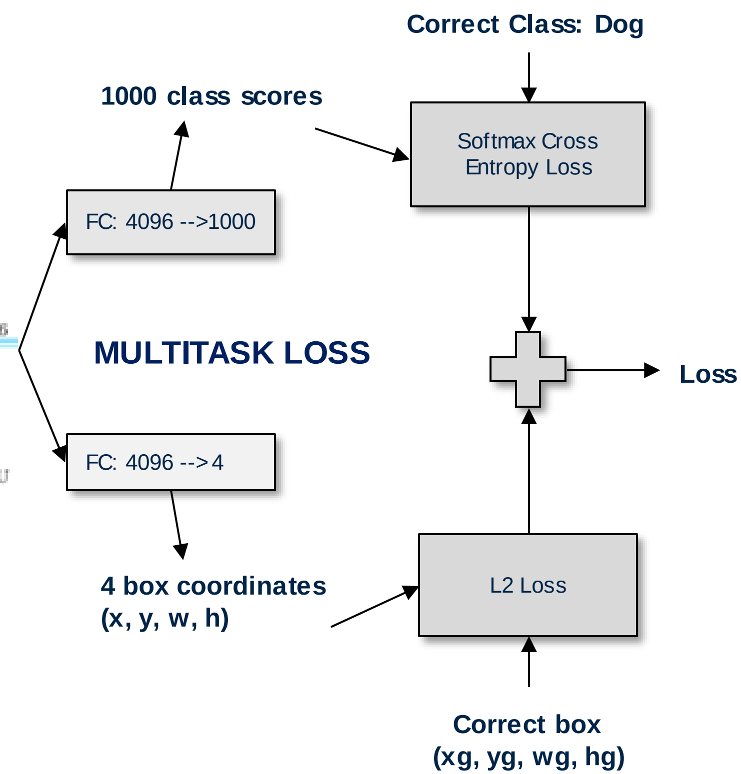
1000 class scores

Classification + Localization



Classification + Localization





How would you do it for multiple objects?

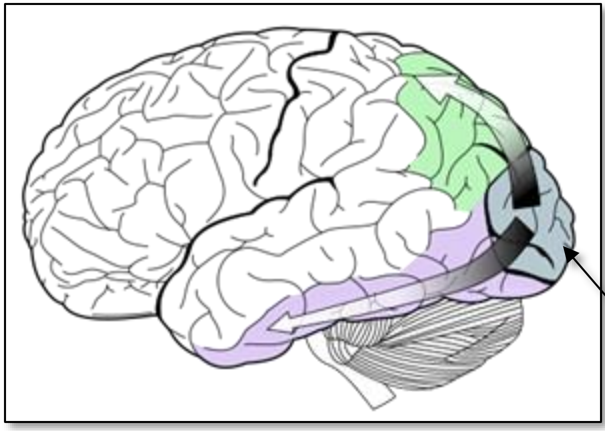
max planck institut informatik

NYU

Can We Learn from Biology?

The Two-Stream Hypothesis:

Dorsal stream | WHERE pathway – object's spatial location relative to the viewer and relation within visual objects



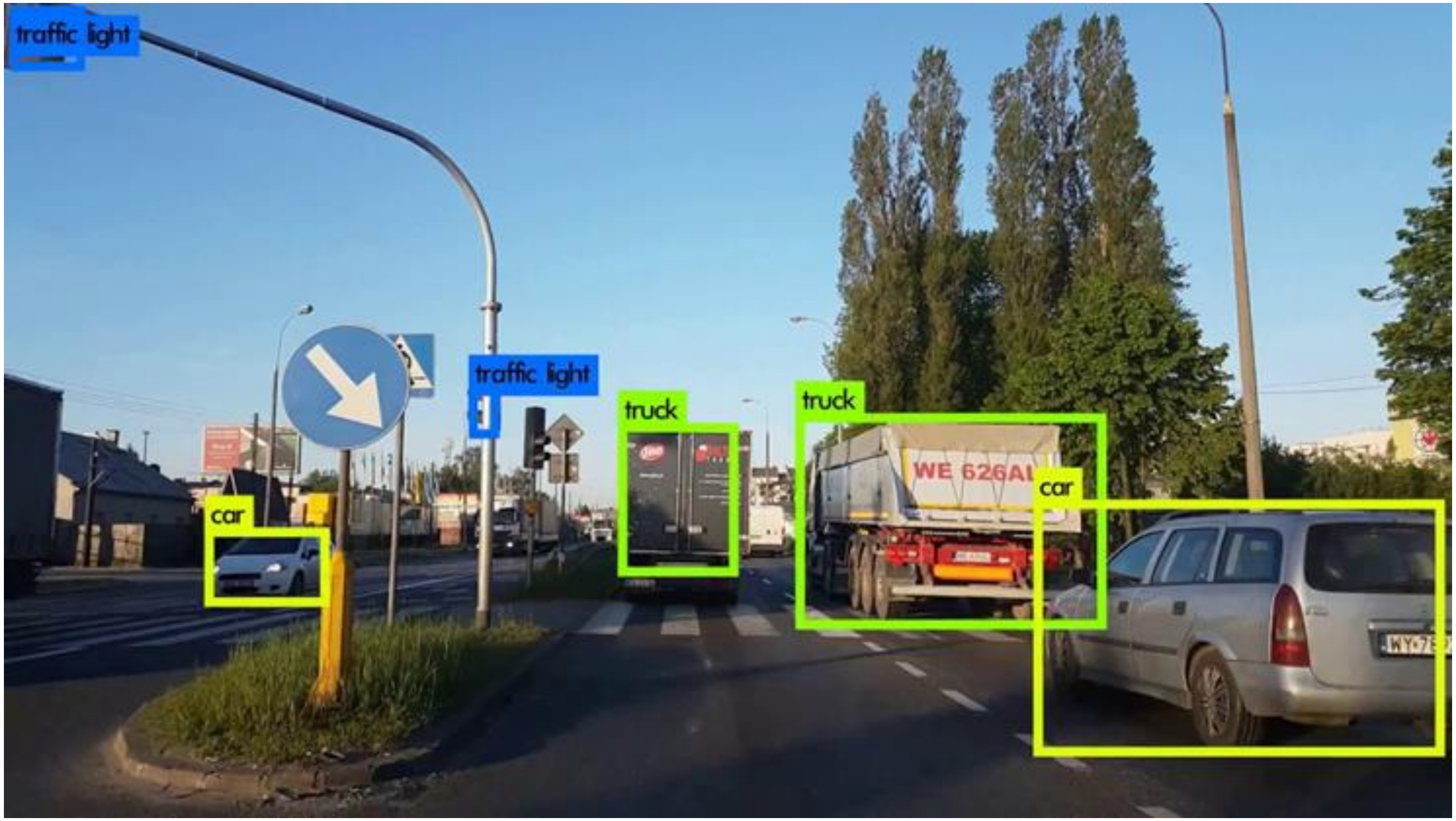
Ventral stream | WHAT pathway –
object identification and recognition

Streams share initial processing (Primary Visual Cortex, V1)

From Wikipidea: https://en.wikipedia.org/wiki/Two-streams_hypothesis

Repeat: Object Detection

Estimating bounding box (regression) + their labels (classification) for **multiple boxes**

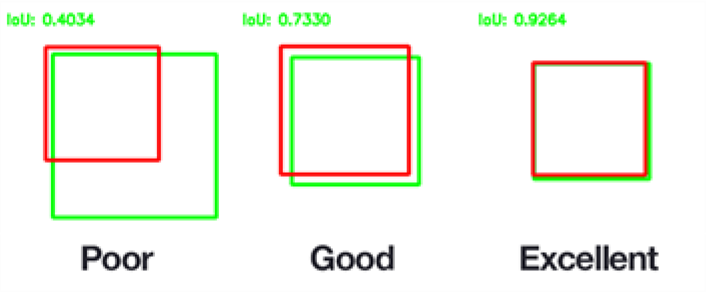


Performance Metrics for Object Detection

- IoU
- Precision and Recall
- Mean Average Precision (mAP)

Performance Metrics for Object Detection

IoU (intersection over union): Measure of box similarity



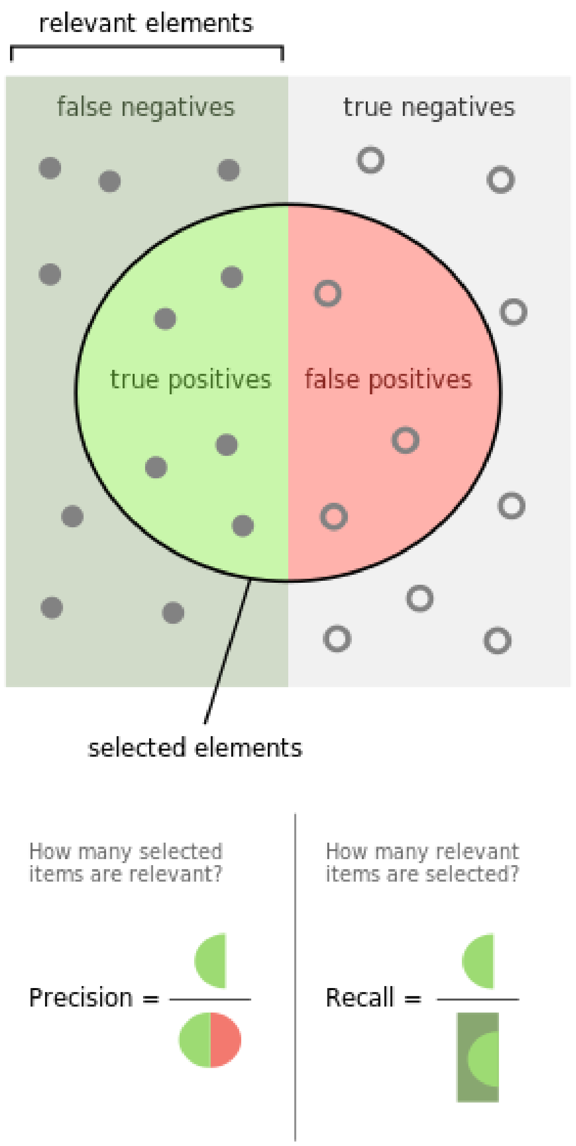
Area of overlap / area of union

Performance Metrics for Object Detection

Precision: what percentage of your positive predictions are correct

Recall: what percentage of ground truth objects were found

Why not be satisfied with Precision 100% or Recall 100%?



Mean Average Precision (mAP) for object detection

Total number of ground truth boxes = 6

Step 1: Sort predictions according to confidence (usually classifier's output after softmax)

Step 2: Calculate IoU of every predicted box with every ground truth box (10 x 6 matrix)

Step 3: Match predictions to ground truth using IoU, correct predictions are those with IoU > threshold (typically 0.5), without replacement.

Confidence	Rank	Correct
0.91	1	TRUE
0.87	2	TRUE
0.83	3	FALSE
0.81	4	TRUE
0.77	5	FALSE
0.65	6	TRUE
0.56	7	TRUE
0.40	8	FALSE
0.32	9	FALSE
0.31	10	TRUE

Mean Average Precision (mAP) for object detection

Step 4: Calculate precision and recall at every row

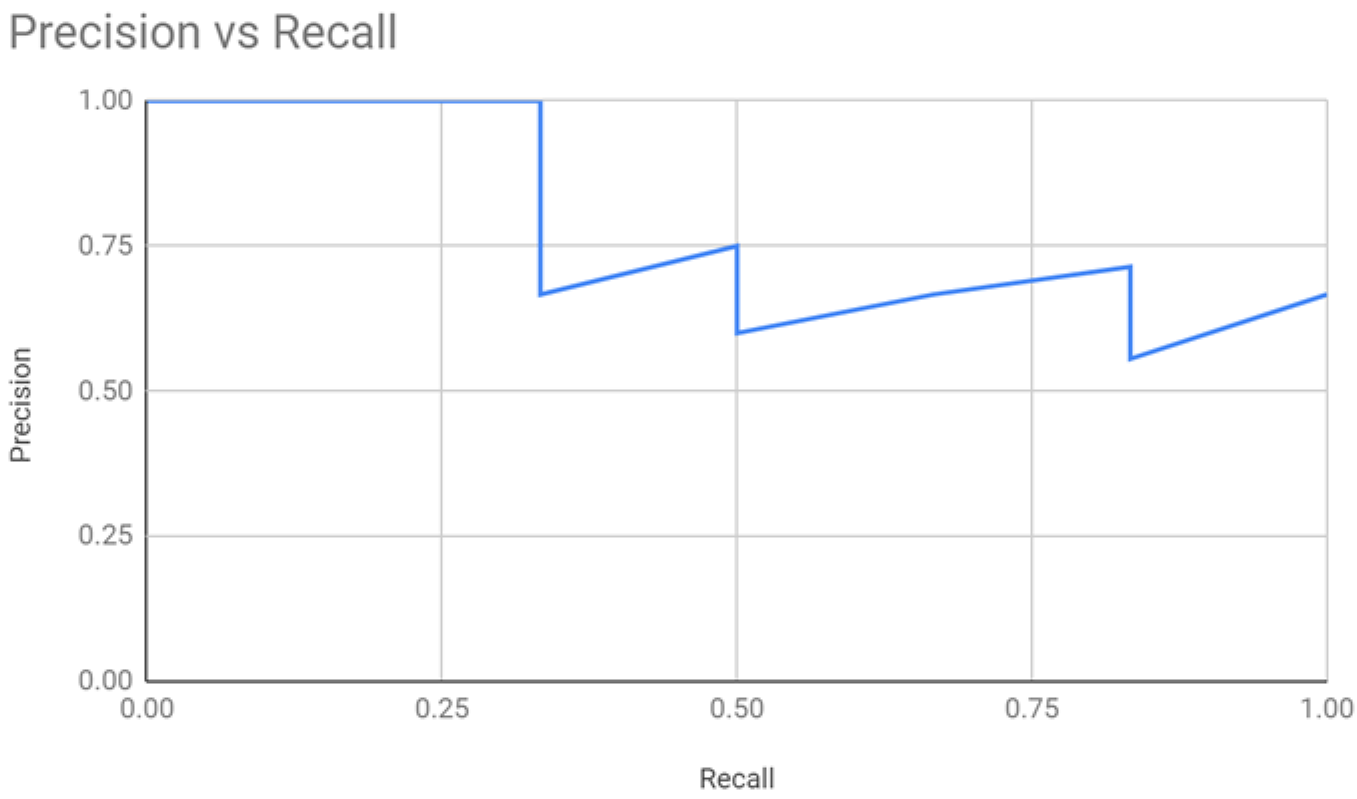
Step 5: Take the mean of **maximum precision at 11 recall values** (0.0, 0.1, … 1.0) to get AP

Step 6: Average across all classes to get the mAP score

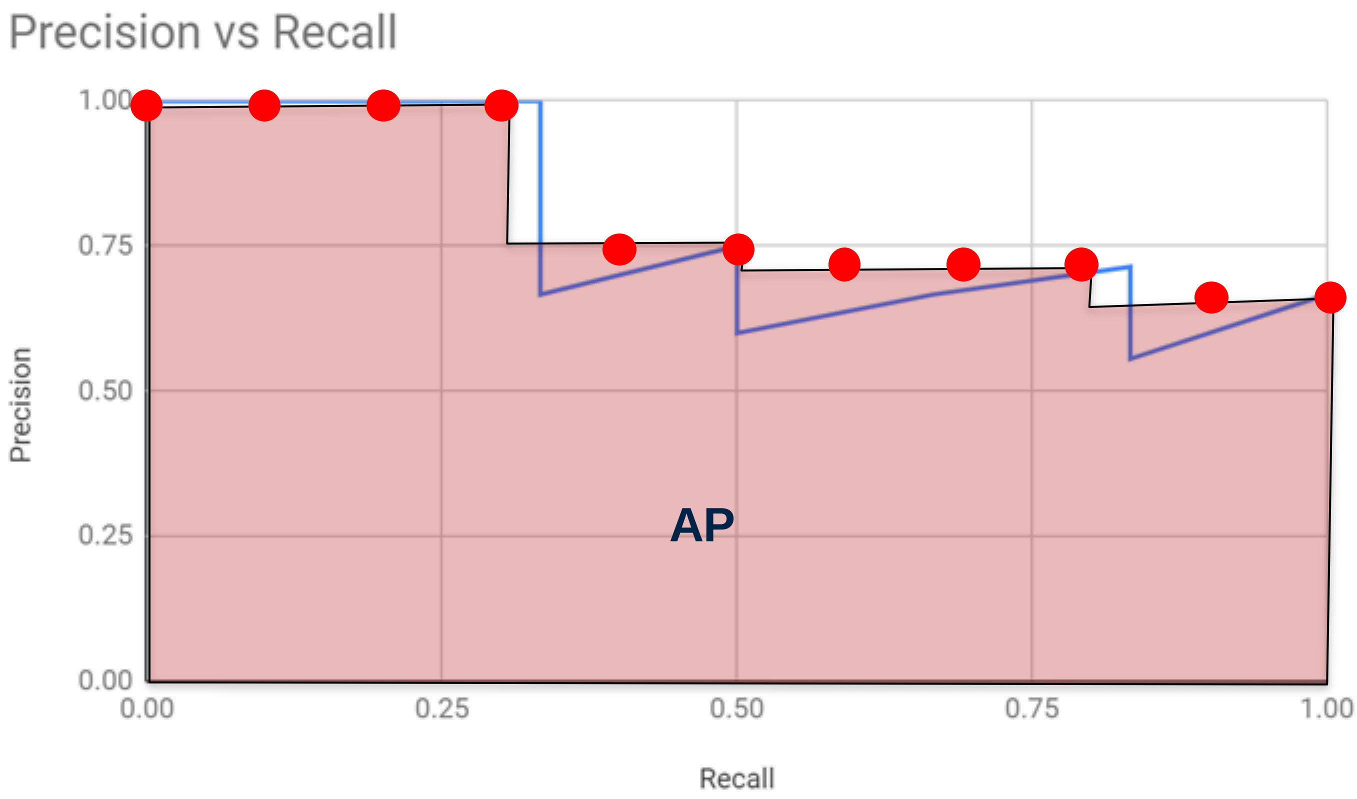
Total number of ground truth boxes = 6

Confidence	Rank	Correct	Precision	Recall
0.91	1	TRUE	1.00	0.17
0.87	2	TRUE	1.00	0.33
0.83	3	FALSE	0.67	0.33
0.81	4	TRUE	0.75	0.50
0.77	5	FALSE	0.60	0.50
0.65	6	TRUE	0.67	0.67
0.56	7	TRUE	0.71	0.83
0.40	8	FALSE	0.63	0.83
0.32	9	FALSE	0.56	0.83
0.31	10	TRUE	0.67	1.00

Precision and Recall Curve

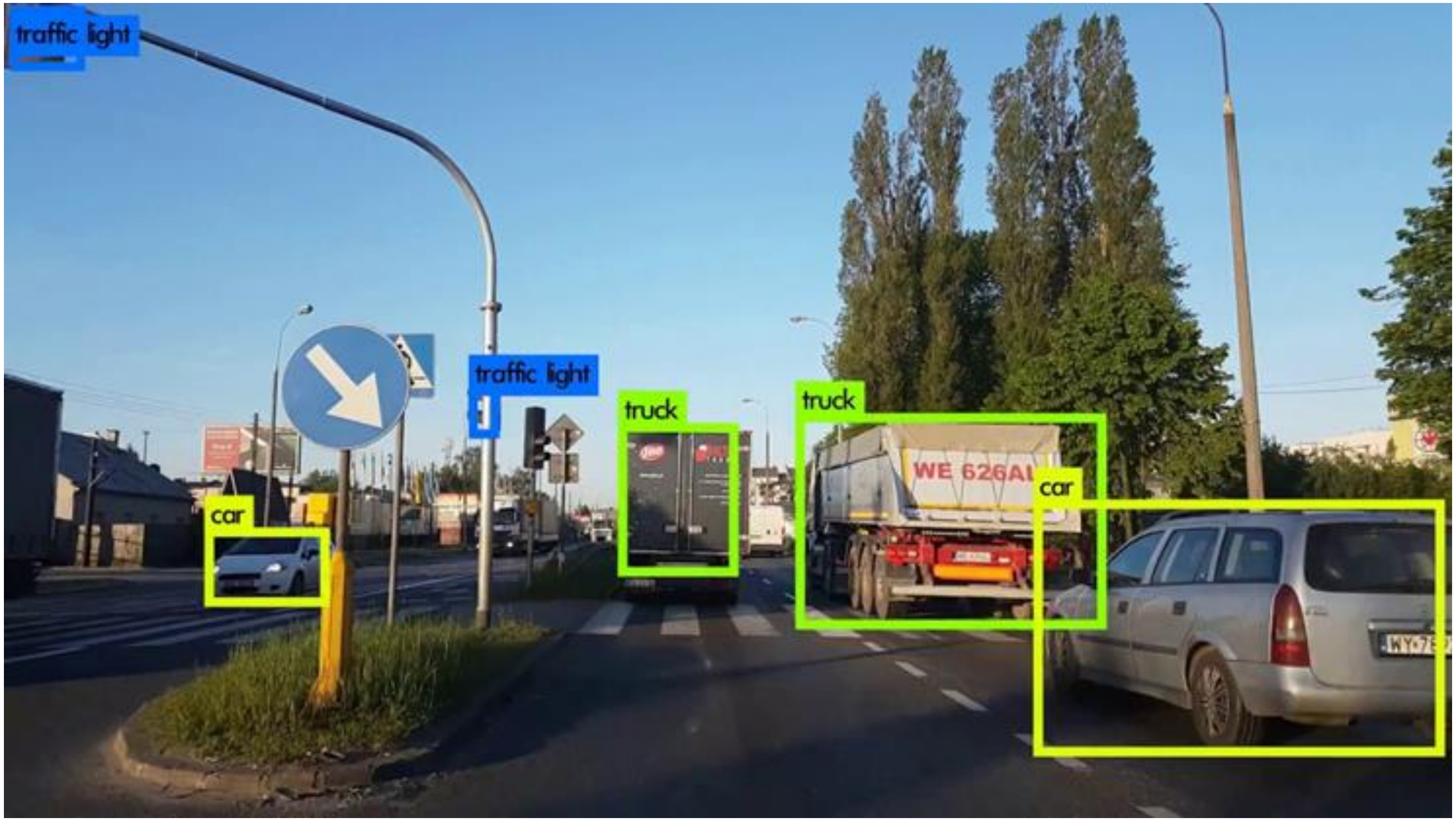


Precision and Recall Curve – Area under the Curve



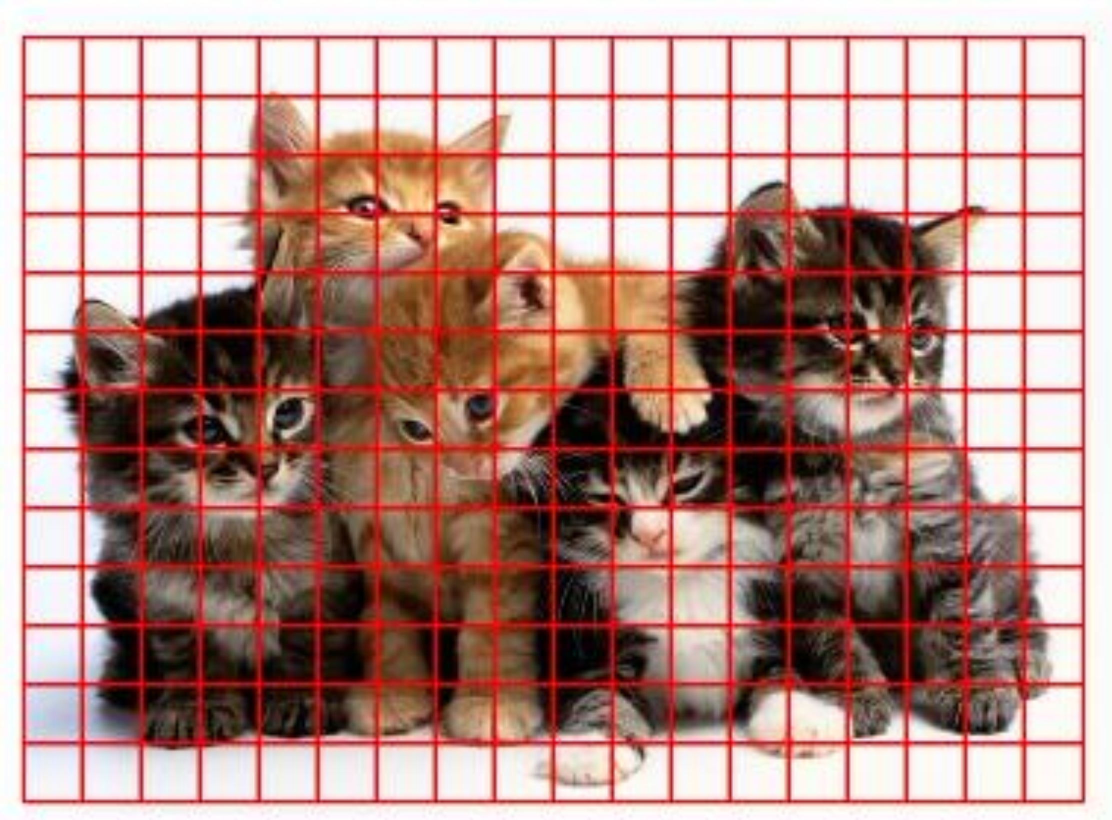
Object Detection

Estimating bounding box (regression) + their labels (classification) for **multiple boxes**



Object Detection: Brute Force Approach

- Run a classifier for **every possible box**.
- So for this 13x18 grid, any ideas how many possible boxes?

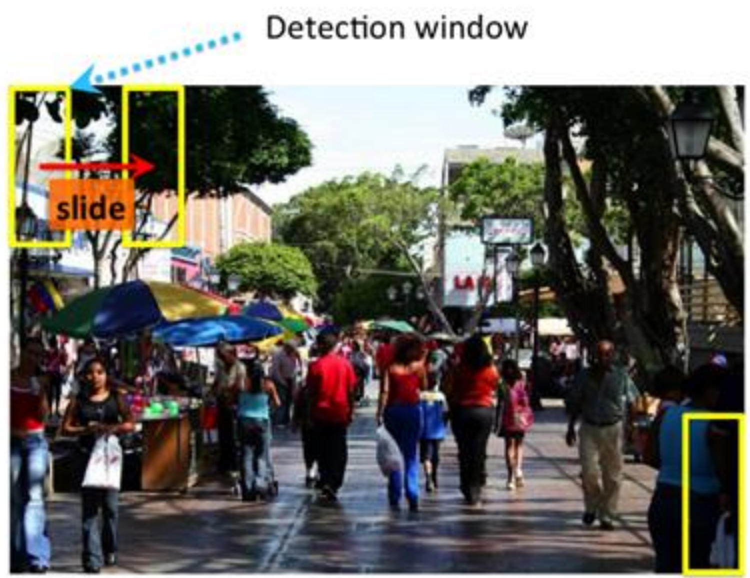


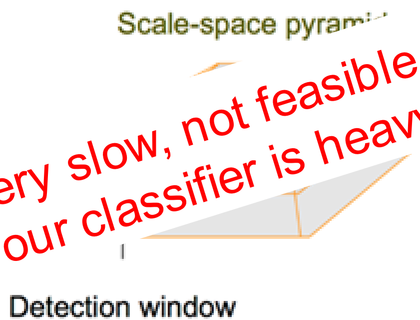
Object Detection: Sliding Window Approach

- Run a classifier in a **sliding window** fashion

Object Detection: Sliding Window Approach

- Run a classifier in a **sliding window** fashion **at every scale on an image pyramid**





Very slow, not feasible if your classifier is heavy

Image is resized several times to capture all possible scales

- Note that we are still not able to deal with different aspect ratios. Lets see what can be done next

Object Detection: Smarter Approach

- Ideas how to reduce number of boxes?
  - Find 'blobby' image regions which are likely to contain objects
  - Run classifier for **region proposals or boxes** *likely* **to contain objects**
- Later: Class-agnostic object detector - "Region Proposals"

Object Detection: Region Proposals

- Greedily combine sub-segmentation to produce larger candidate object locations

Candidate object locations

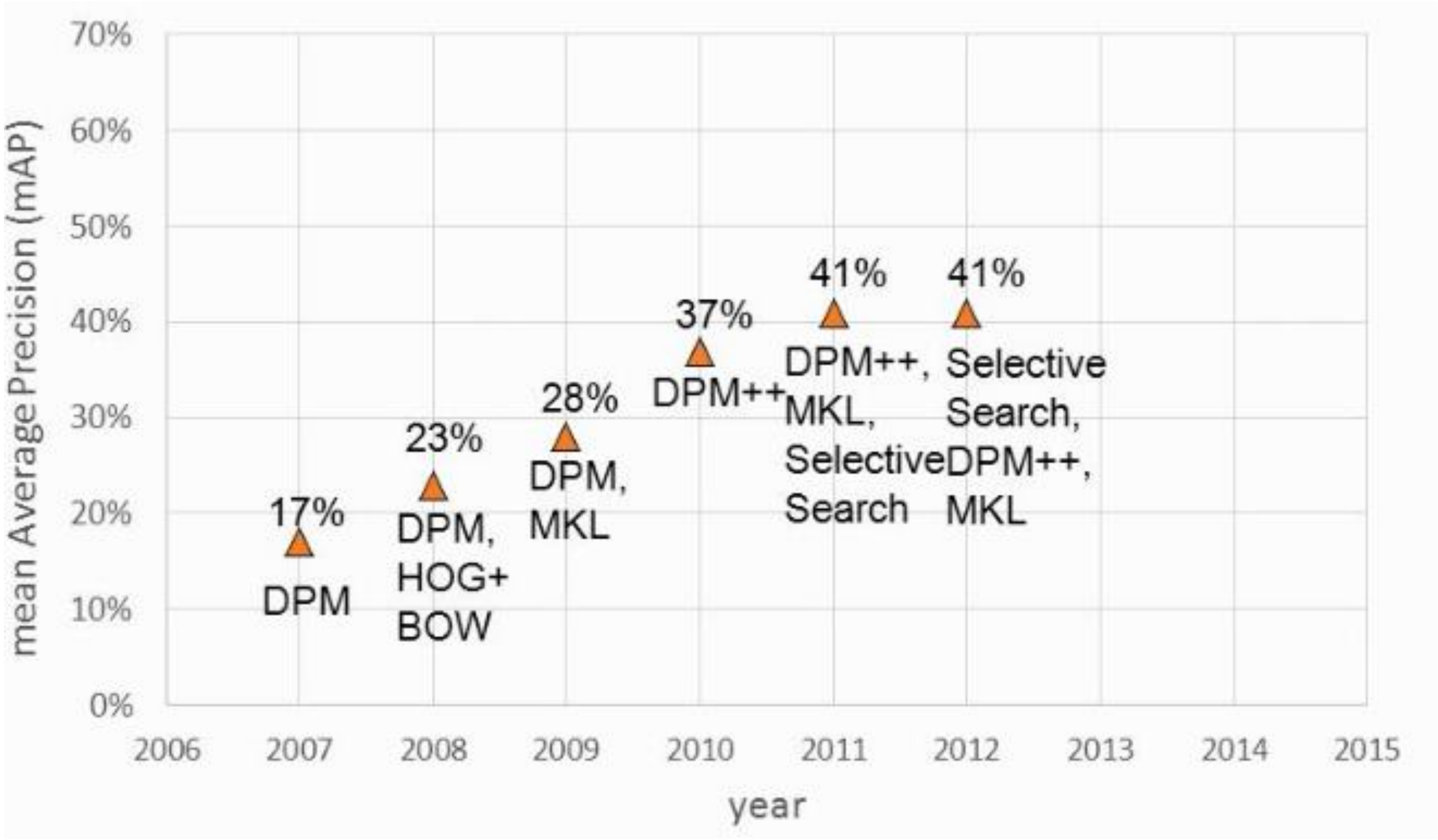
Extract HOG features over windows

Run Linear SVM classifier

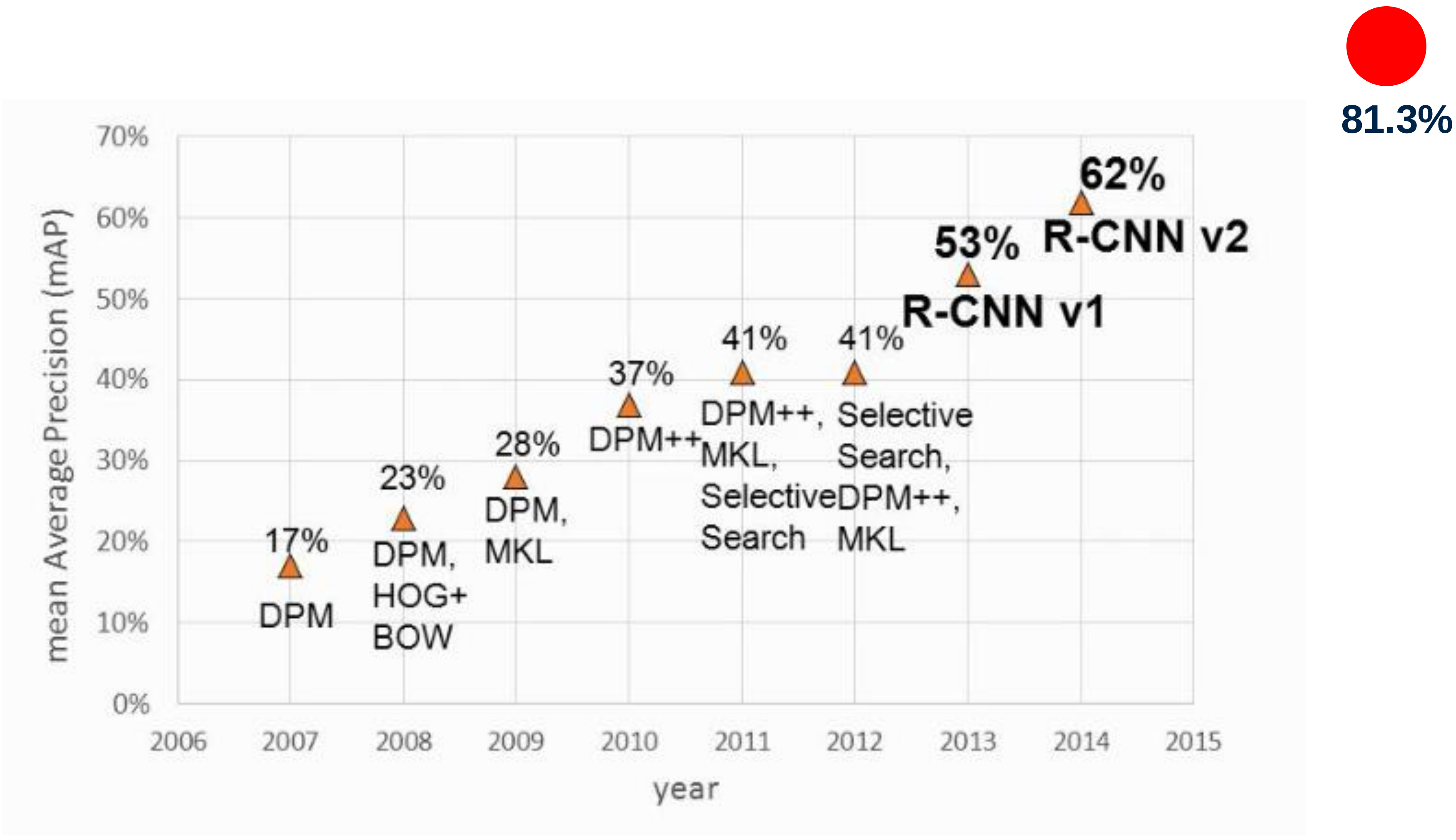
Run post-processing (Non-Maximum Suppression)

Selective Search for Object Recognition, Uijlings et al. (2013)

Pre Deep Learning Era

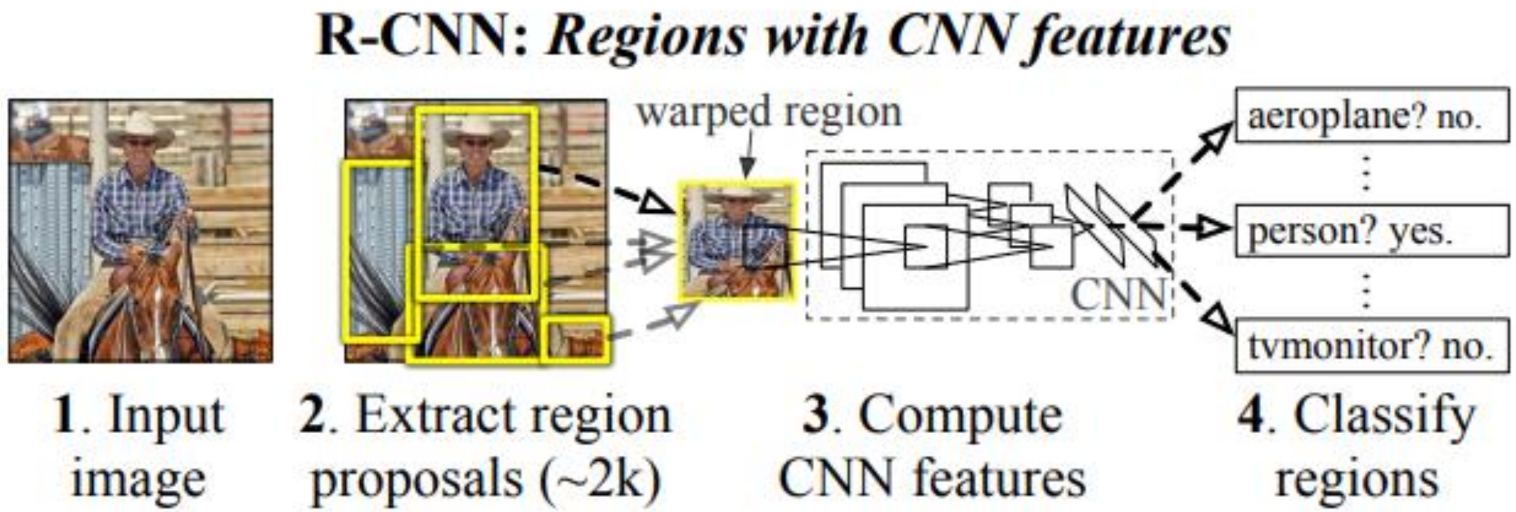


Post Deep Learning



Object Detection: R-CNN

- Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al. (2013)



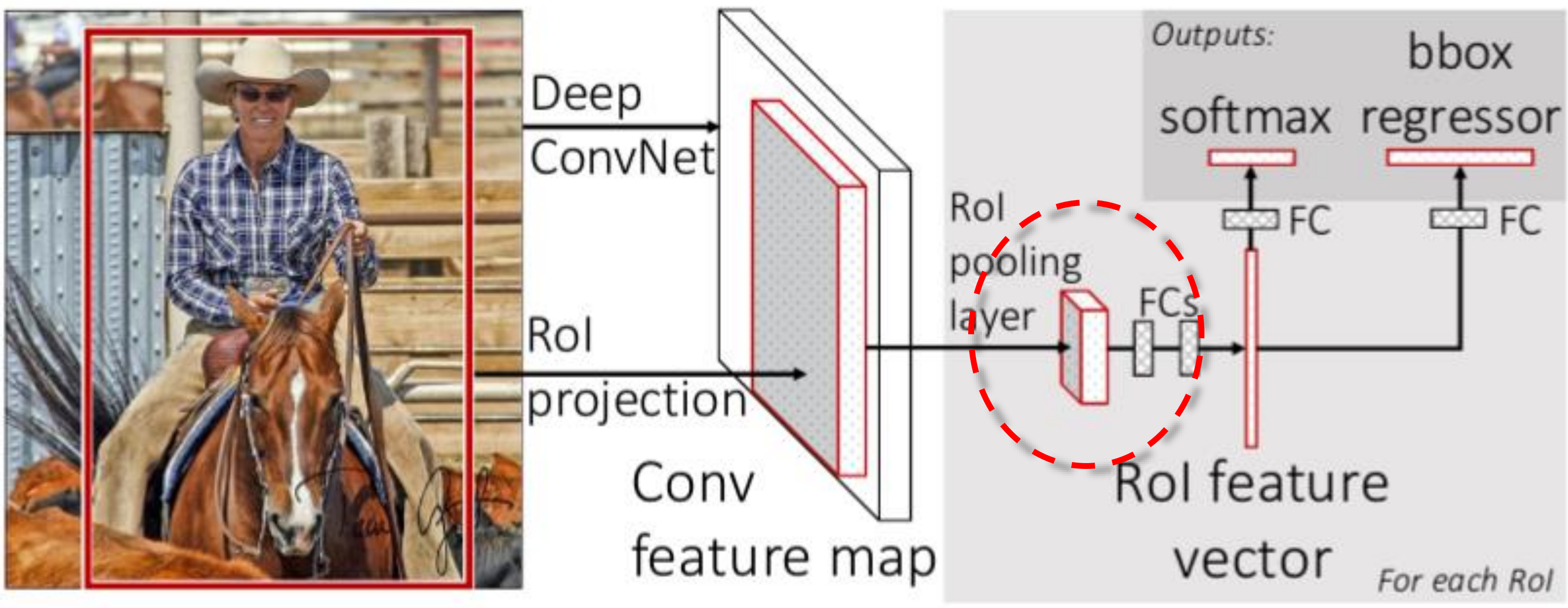
SVM classifier

Exactly the same as Selective Search, except for CNN features instead of HOG

https://arxiv.org/abs/1311.2524

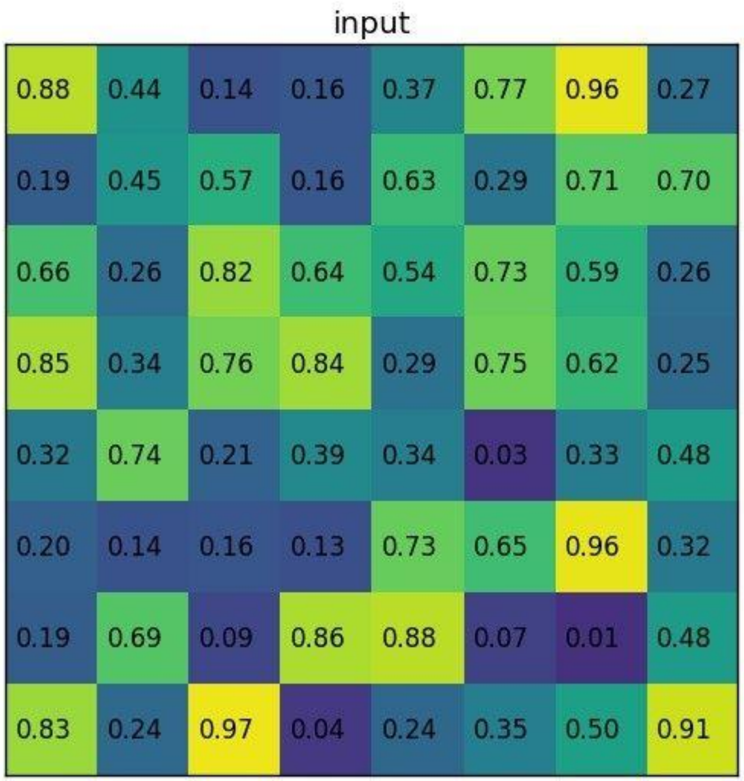
Object Detection: Fast R-CNN

- Fast Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al. (2015)



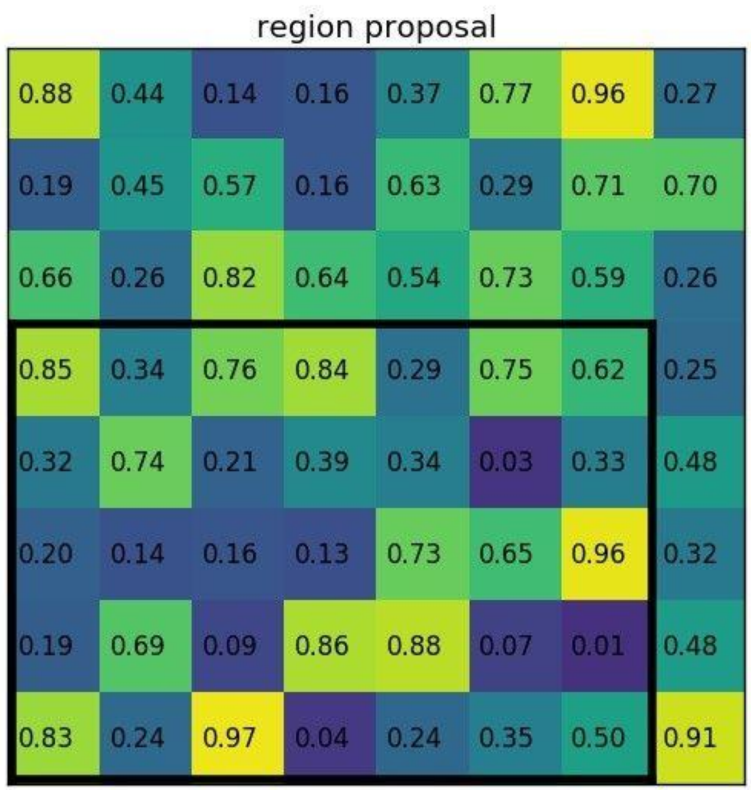
https://arxiv.org/abs/1504.08083

- RoI pooling on a single 8×8 feature map, one region of interest and an output size of 2x2. Our input feature map looks like this:



https://deepsense.ai/region-of-interest-pooling-explained/

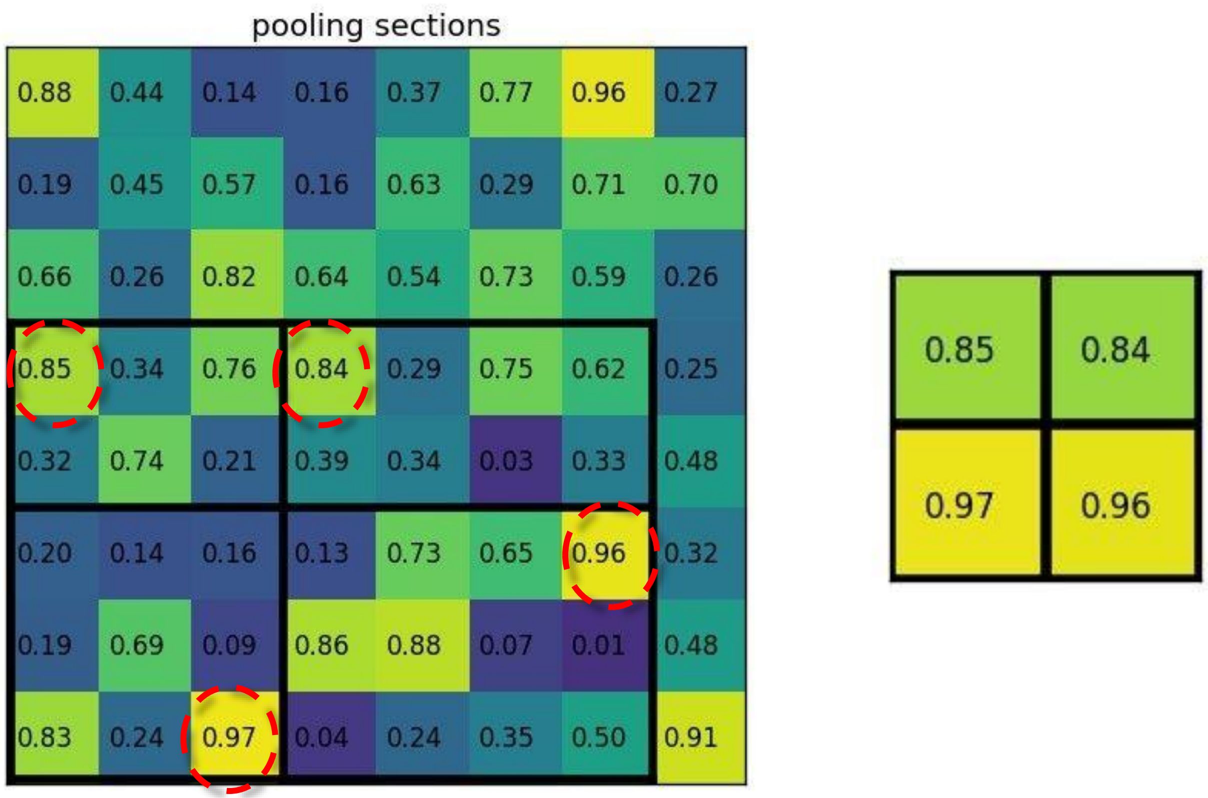
- Let's say we also have a region proposal (0, 3) - (7, 8)



https://deepsense.ai/region-of-interest-pooling-explained/

Object Detection: Fast R-CNN – ROI Pooling

- By dividing it into 2x2 sections (because the output size is 2x2) we get:

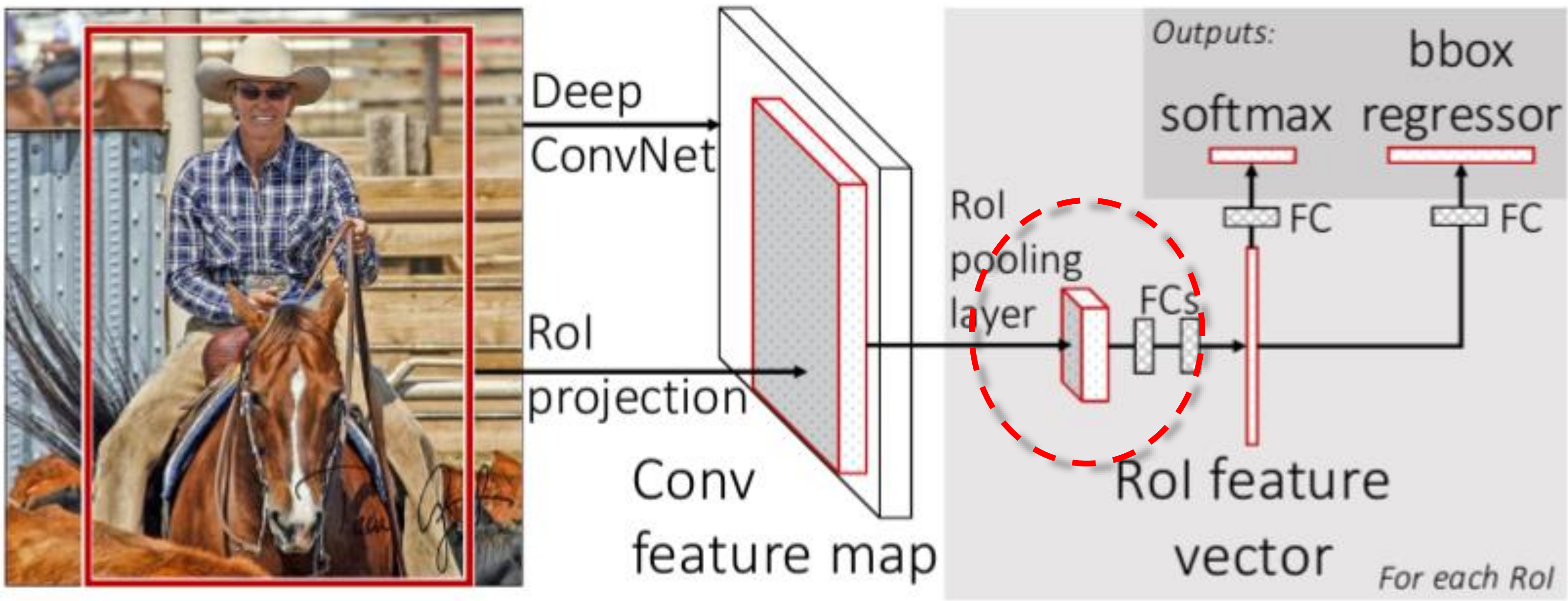


https://deepsense.ai/region-of-interest-pooling-explained/

Object Detection: Fast R-CNN

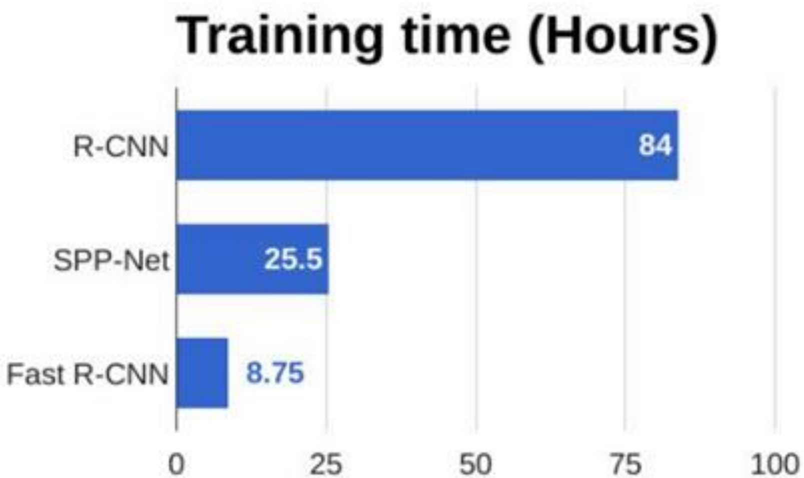
- Fast Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al. (2015)

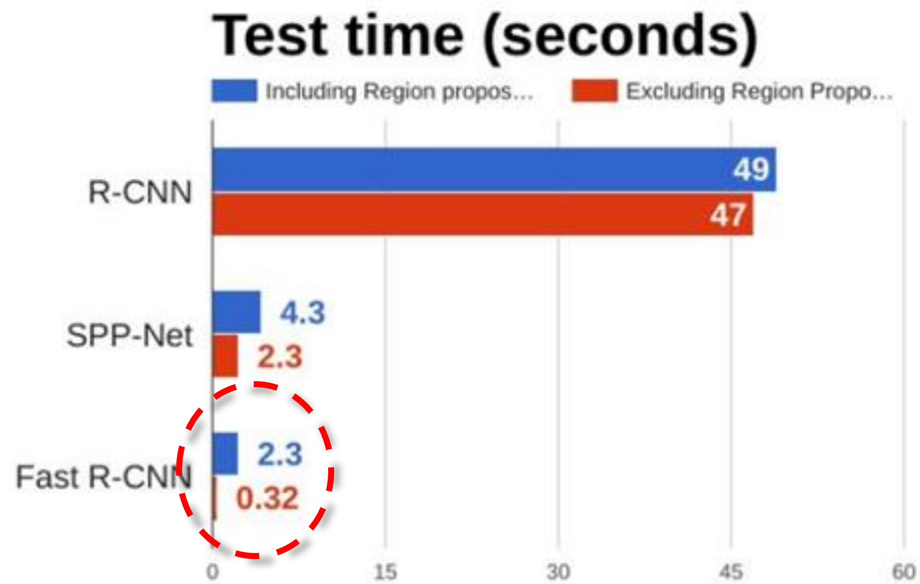
2000 region proposals



https://arxiv.org/abs/1504.08083

Object Detection: Region Proposals





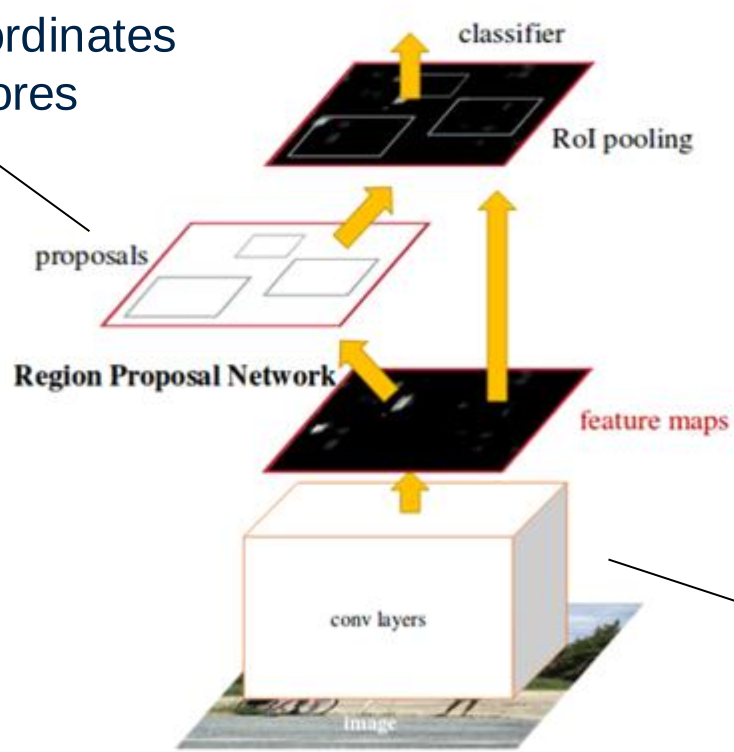
Majority time still taken by Region Proposal generation

Object Detection: Faster R-CNN

https://arxiv.org/abs/1506.01497

- Main idea: Use a CNN for region proposals

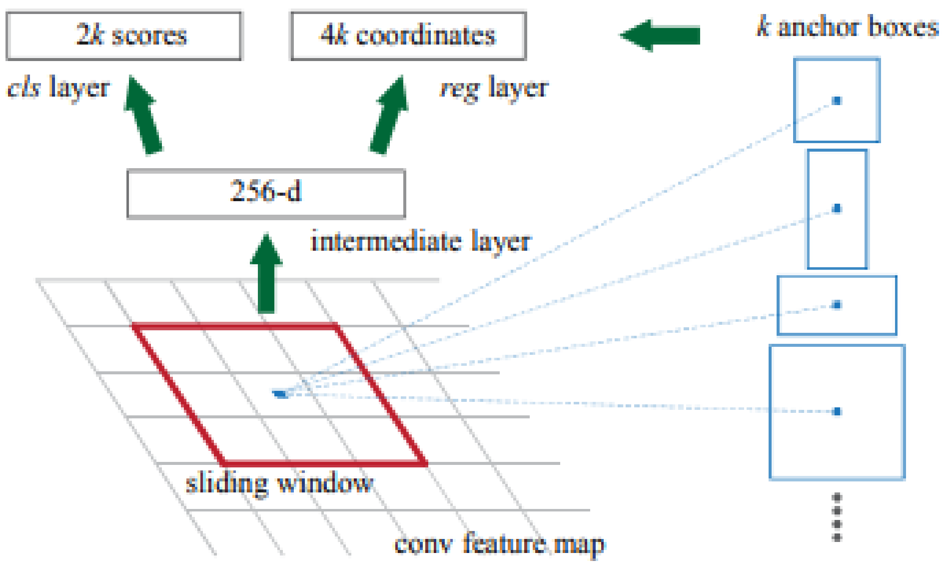
Trained to regress:
- 2400 x 4 Box coordinates
- 2 ***objectness*** scores



Core network computation shared by RPN and classifier head

Object Detection: Region Proposals using CNN

- At each uniformly sampled grid cell, set of anchor boxes with different aspect ratios and scales



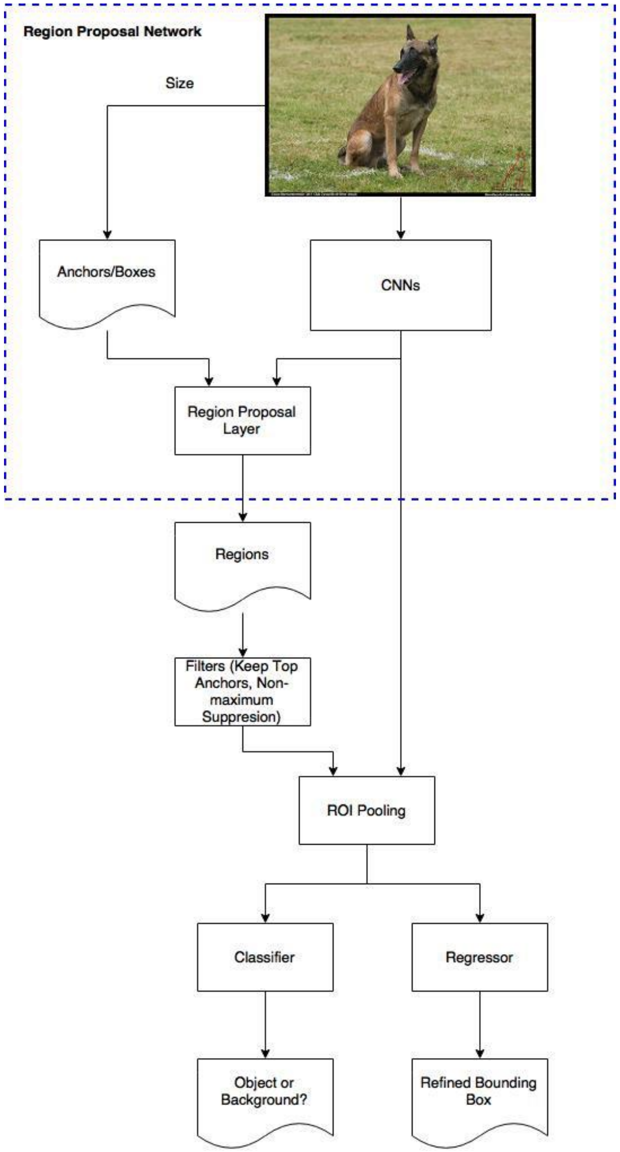
For each anchor box

- Regress to a final box with 4 numbers: (*dx, dy, dh, dw)*
- Predict 2 ***objectness*** scores (including background as a separate class)

For anchors, we use 3 scales with box areas of 1282, 2562, and 5122 pixels, and 3 aspect ratios of 1:1, 1:2, and 2:1

https://arxiv.org/abs/1506.01497

Object Detection: Faster RCNN

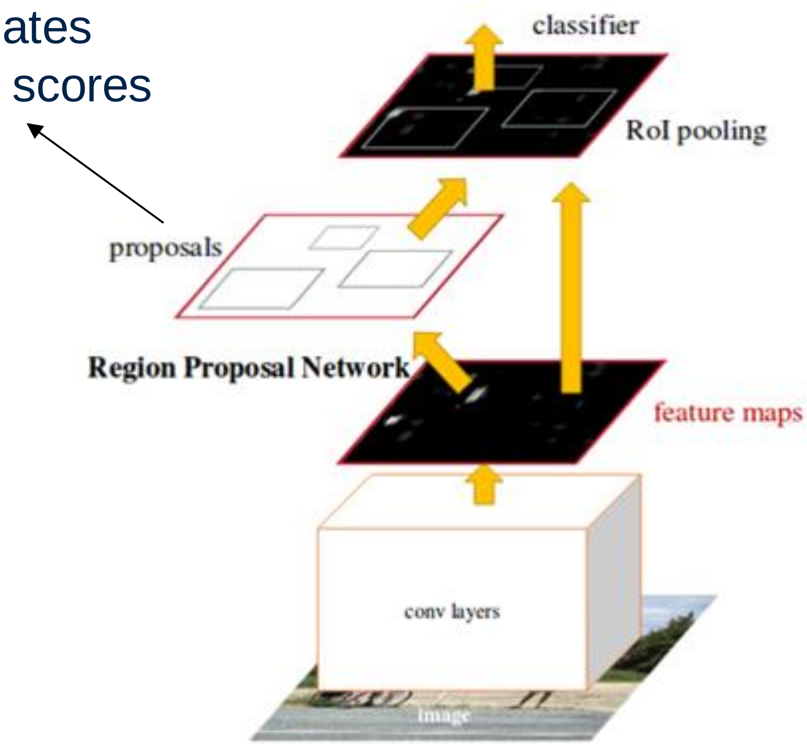


https://arxiv.org/abs/1506.01497

Object Detection: Beyond Faster R-CNN

- Note how we are still looping over all RoIs

Trained to regress:
- 4 Box coordinates
- 2 ***objectness*** scores



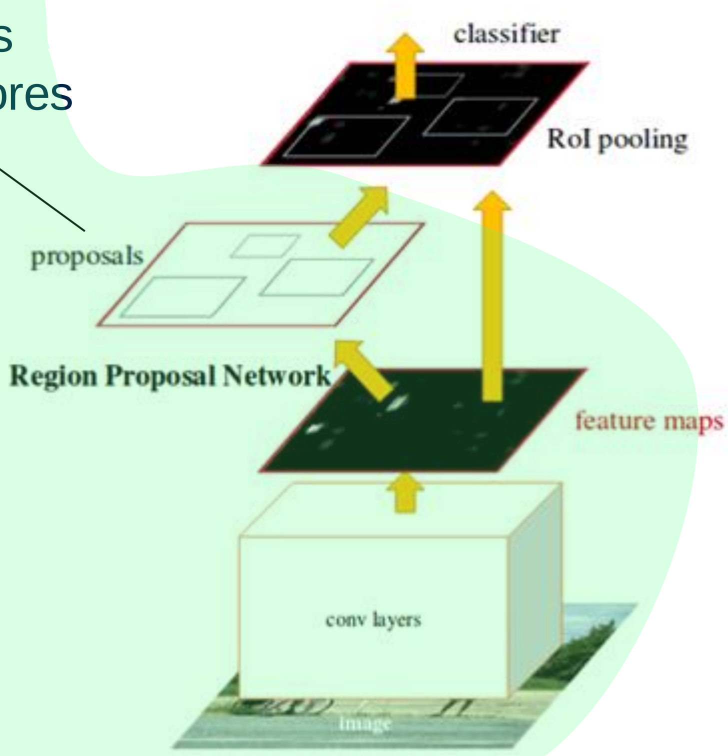
Runs at 200 ms per image

Can this be further simplified?

https://arxiv.org/abs/1506.01497

Object Detection: Beyond Faster R-CNN

Trained to regress:
- 4 Box coordinates
- 2 ***objectness*** scores
- 1 class score



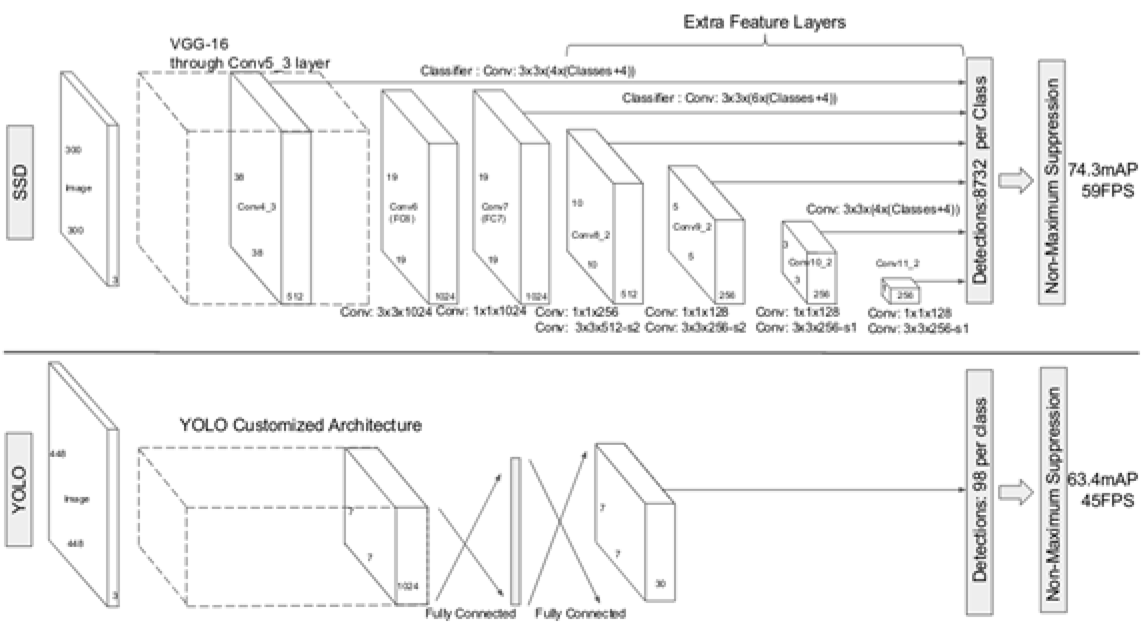
Compress into single stage:
Remove the loop for each region-based computation

https://arxiv.org/abs/1506.01497

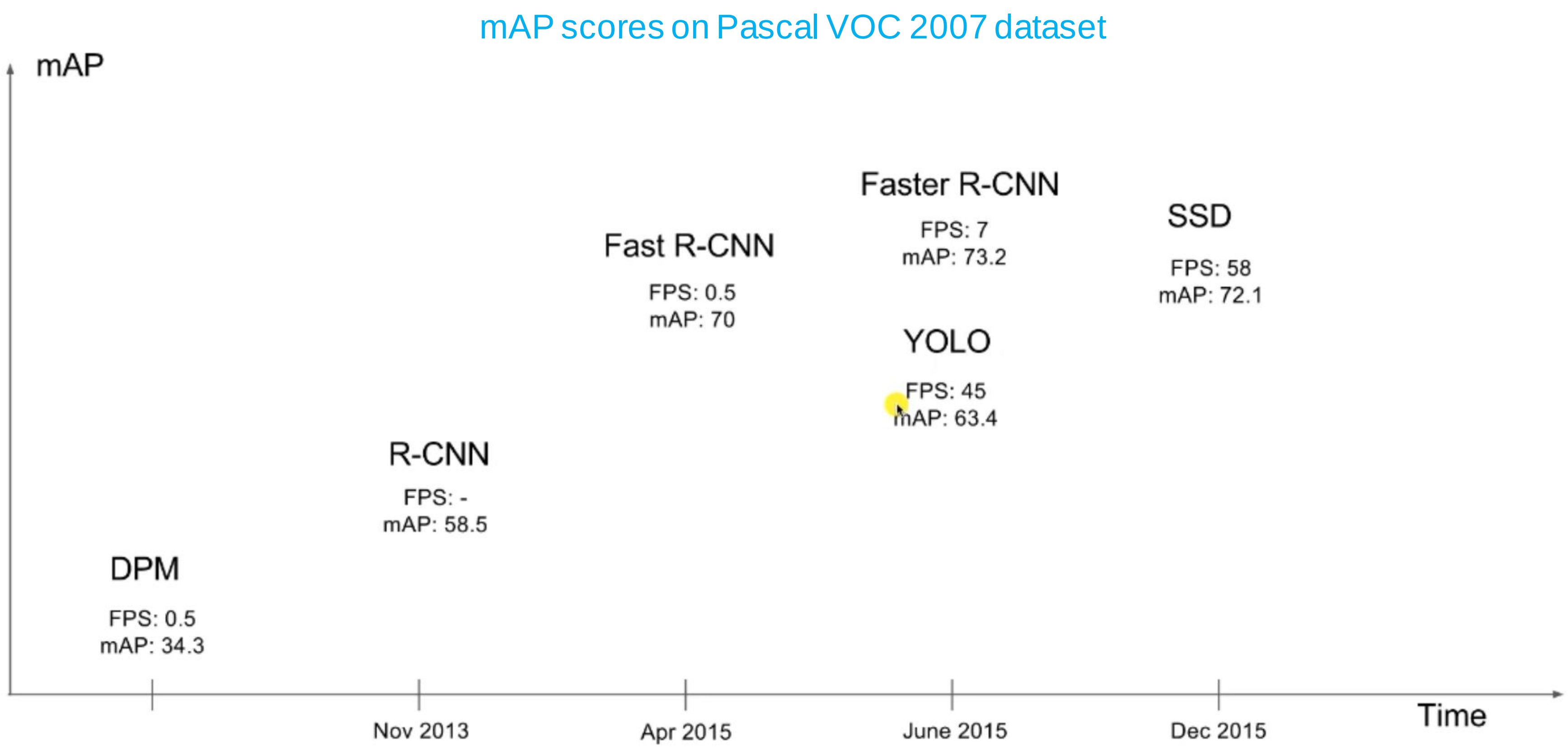
Single Shot Object Detection

- Single Shot Multibox Detector
- YOLO: You Only Look Once & SSD:

Directly estimate box coordinates and class scores in one shot



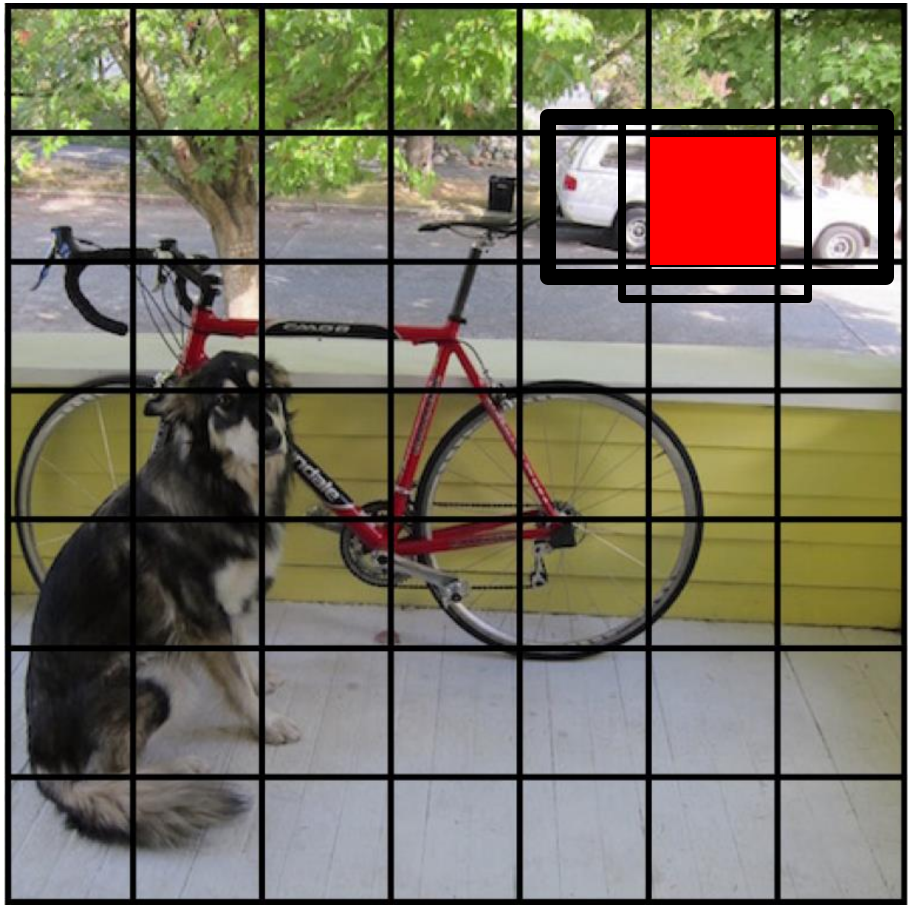
Accuracy vs Speed for Object Detection



YOLO: You Only Look Once

- Predict all bounding boxes for all objects in one shot

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)

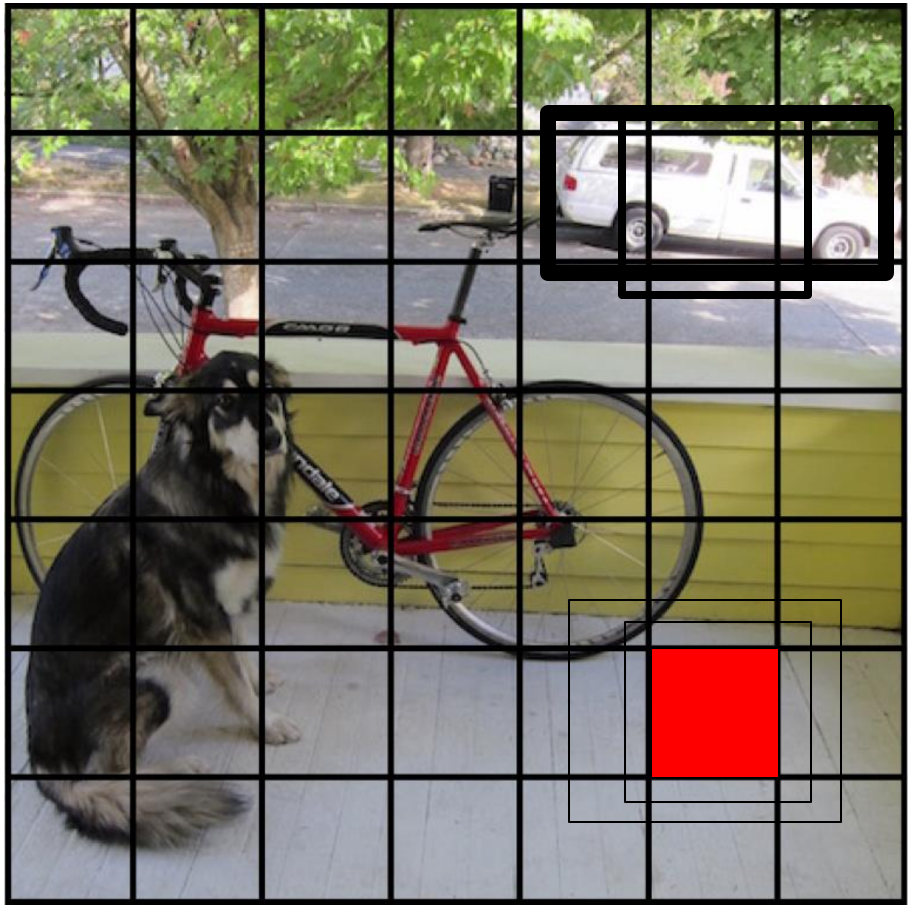


https://arxiv.org/abs/1506.02640

YOLO: You Only Look Once

- Predict all bounding boxes for all objects in one shot

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)

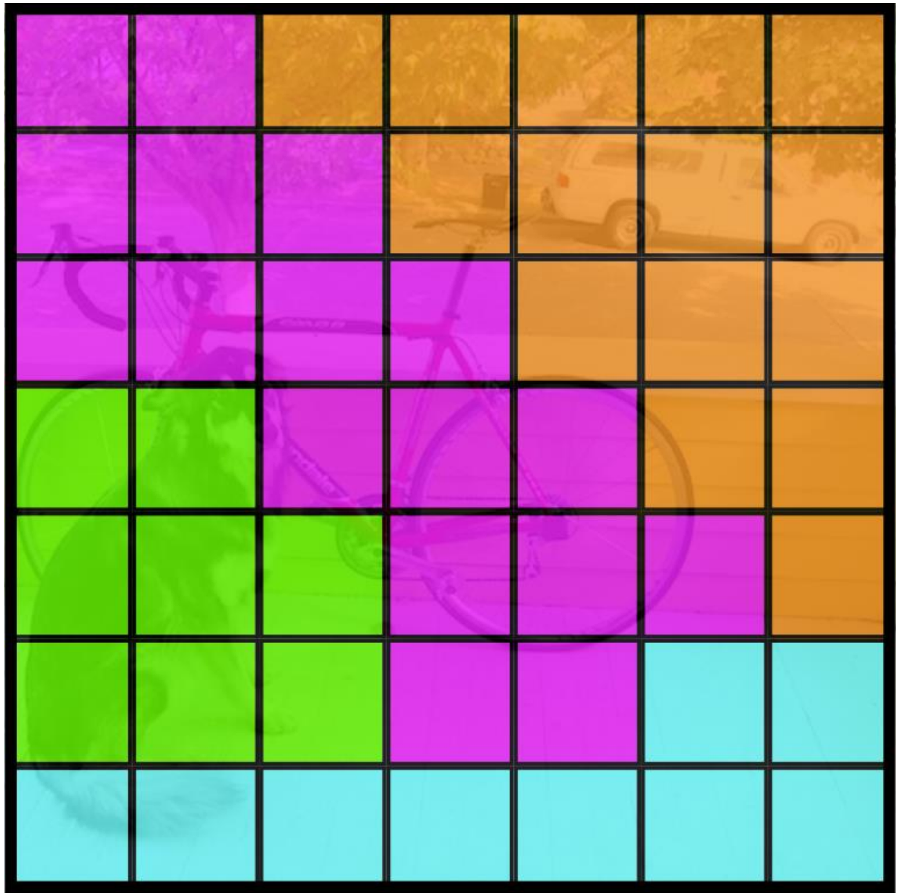


YOLO: You Only Look Once

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)

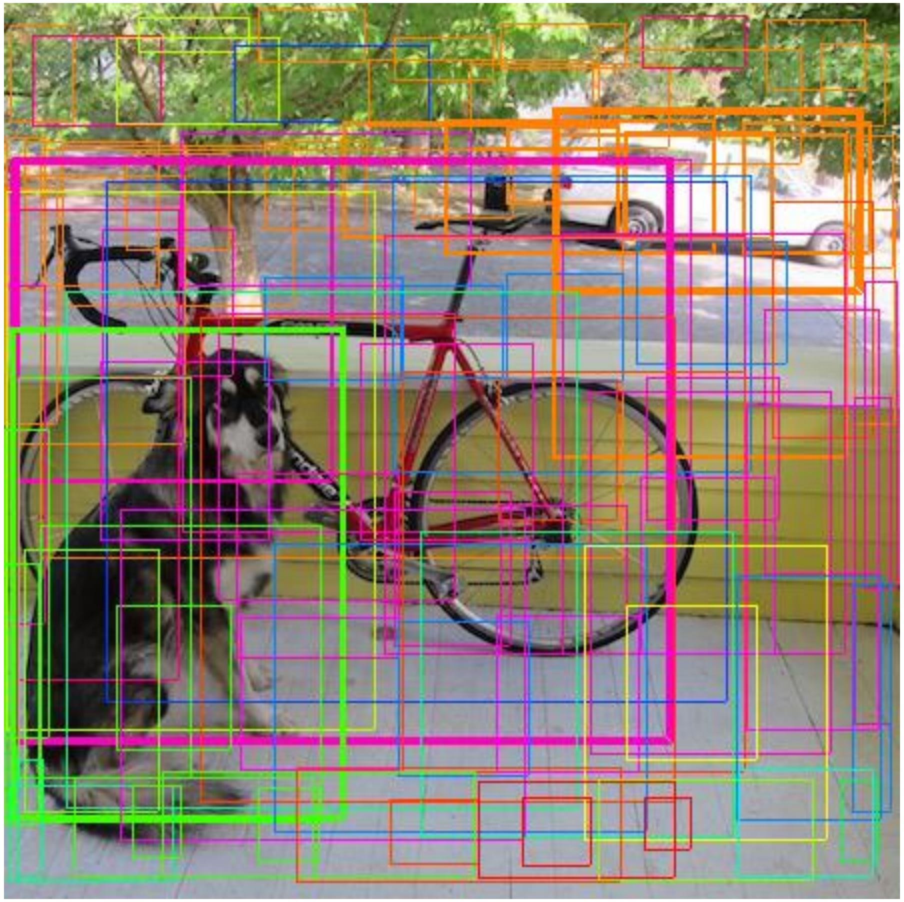
YOLO: You Only Look Once

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)
- Each cell also predicts a class probability
  - **Conditioned on object: P(Car | Object)**



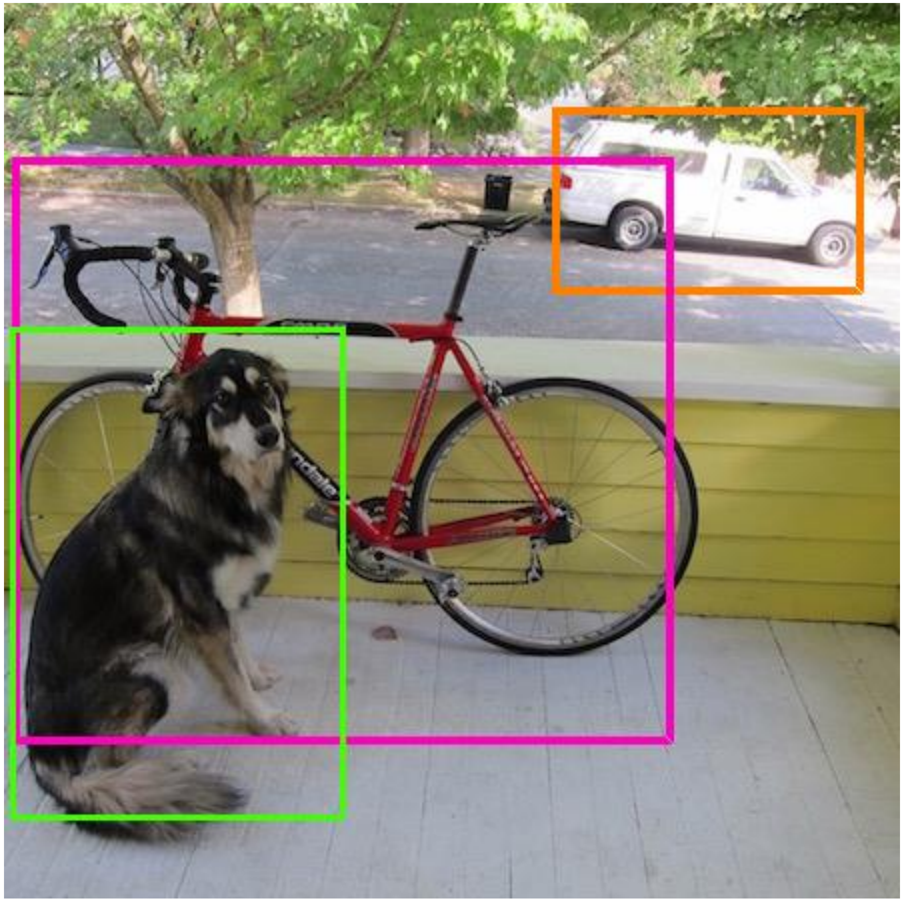
YOLO: You Only Look Once

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)
- Each cell also predicts a class probability
- Then box and class predictions are combined



YOLO: You Only Look Once

- Split the image into a grid
- Each cell predicts boxes and confidences P(Object)
- Each cell also predicts a class probability
- Then box and class predictions are combined
- Followed by NMS



YOLO: You Only Look Once output space

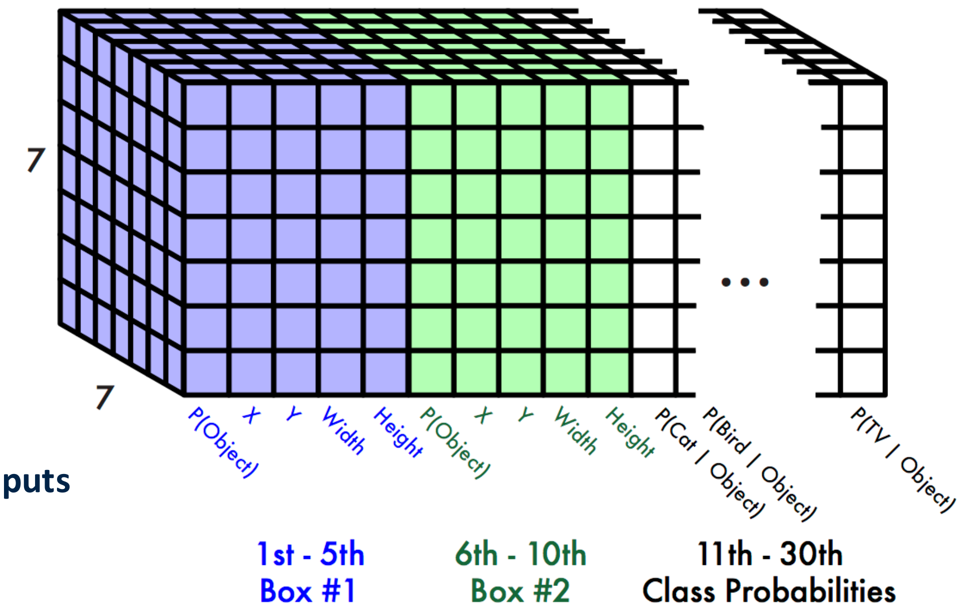
Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

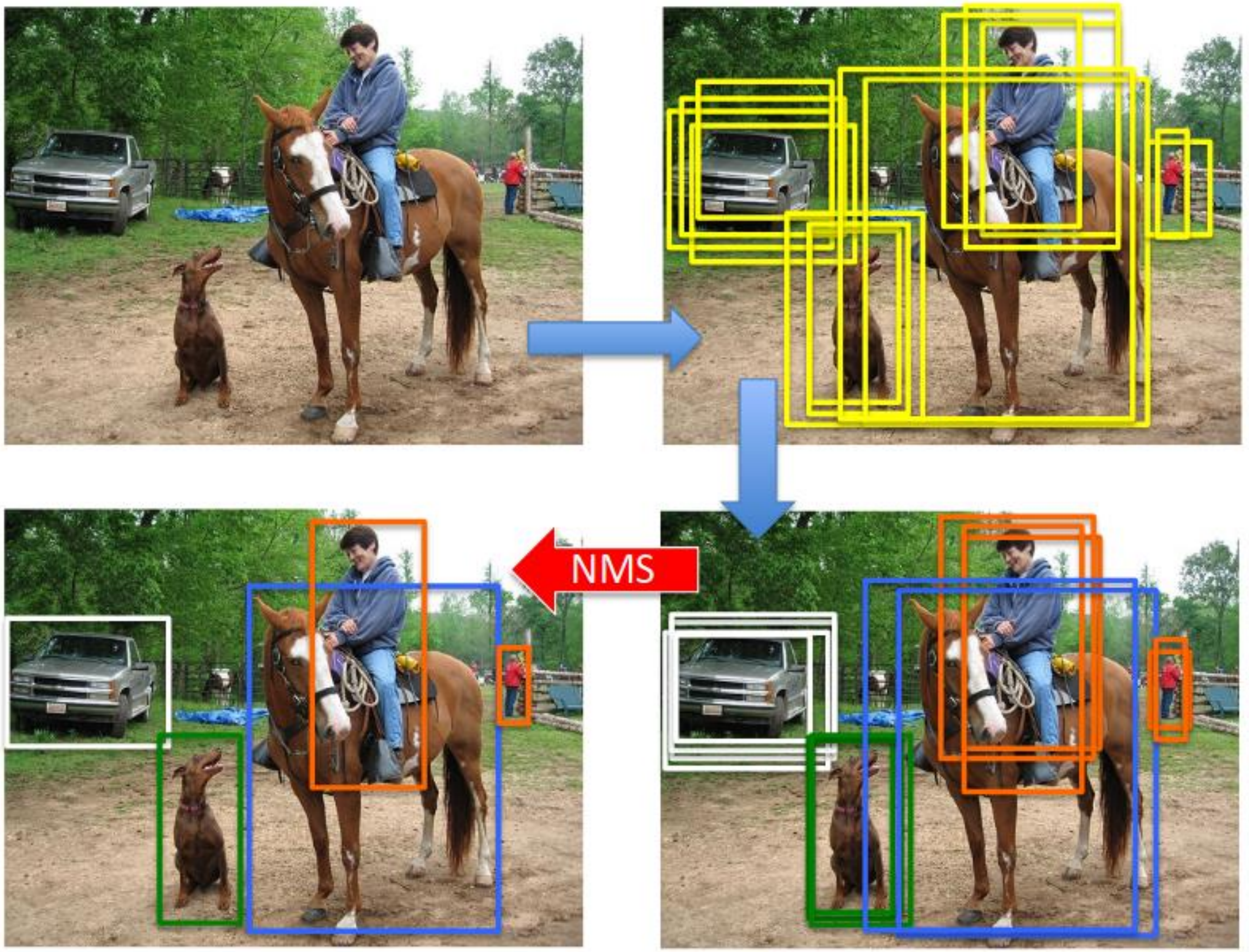
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**



NMS: Non-Maximal Suppression



NMS: Non-Maximal Suppression

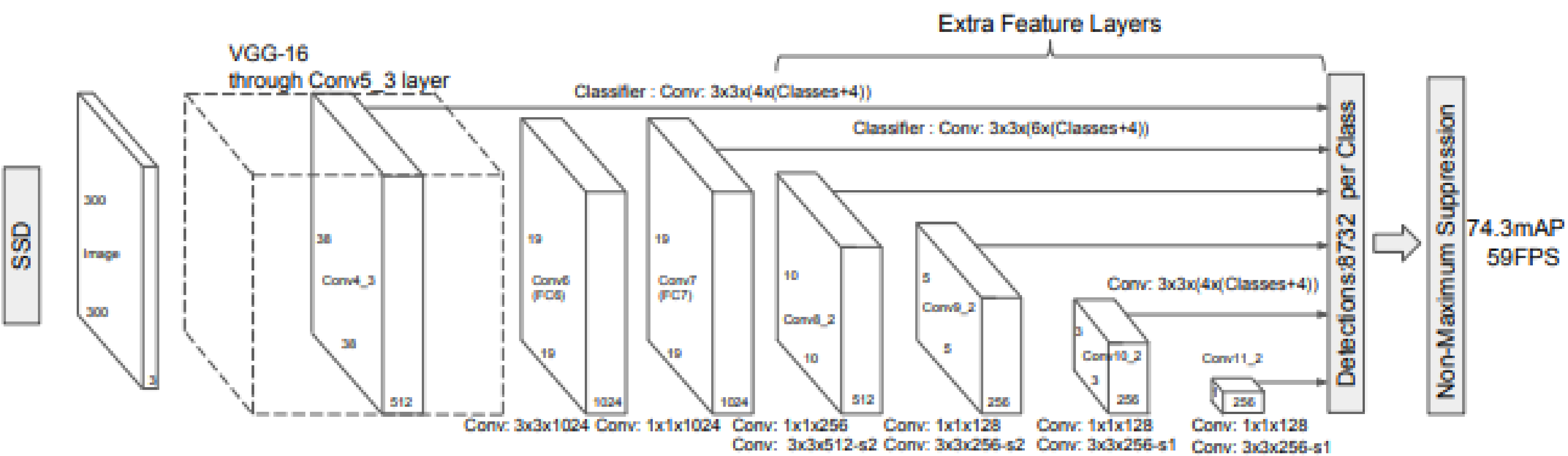
Input : $\mathcal{B} = \{b_1, .., b_N\}$, $\mathcal{S} = \{s_1, .., s_N\}$, N_t
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
$\mathcal{D} \leftarrow \{\}$
while $\mathcal{B} \neq empty$ **do**
$m \leftarrow \text{argmax } \mathcal{S}$
$\mathcal{M} \leftarrow b_m$
$\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}$; $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
for b_i *in* $\mathcal{B}$ **do**
if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
$\mathcal{B} \leftarrow \mathcal{B} - b_i$; $\mathcal{S} \leftarrow \mathcal{S} - s_i$
end
NMS

$s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$
Soft-NMS
end
end
return $\mathcal{D}, \mathcal{S}$
end

SSD: Single Shot Multibox Detector

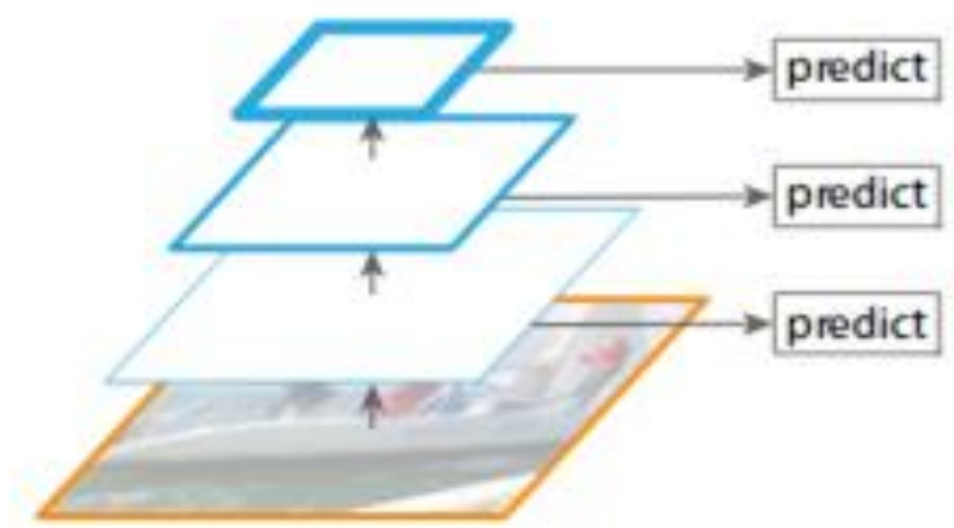
Like YOLO, but predicts using a multi-scale pyramidal feature hierarchy



https://arxiv.org/abs/1512.02325

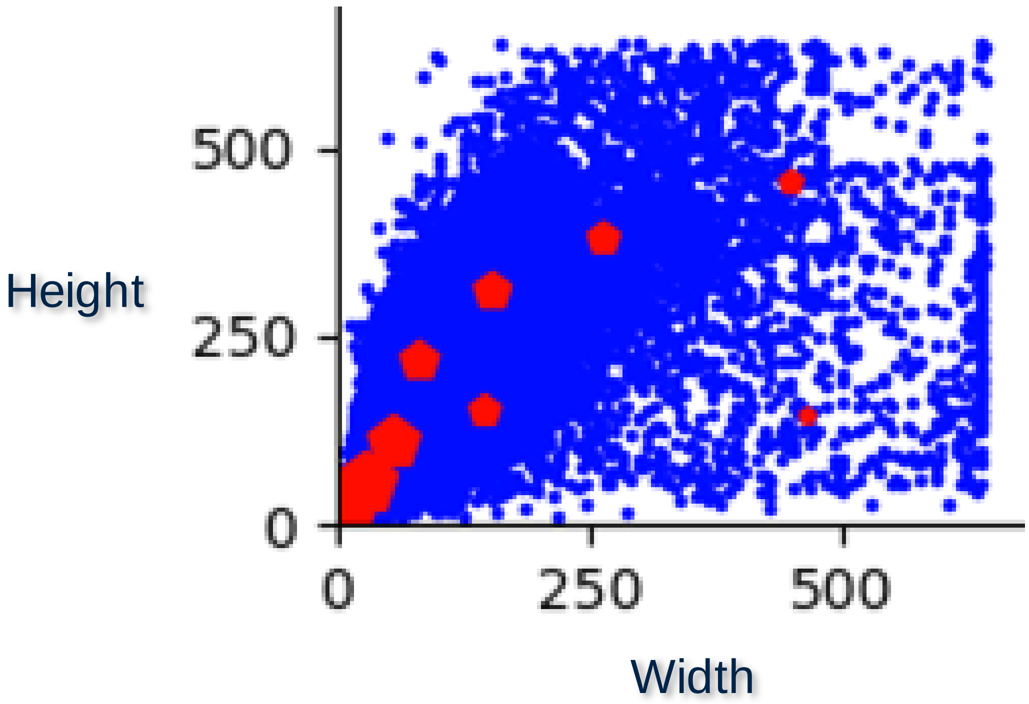
SSD: Single Shot Multibox Detector

Like YOLO, but predicts using a multi-scale pyramidal feature hierarchy



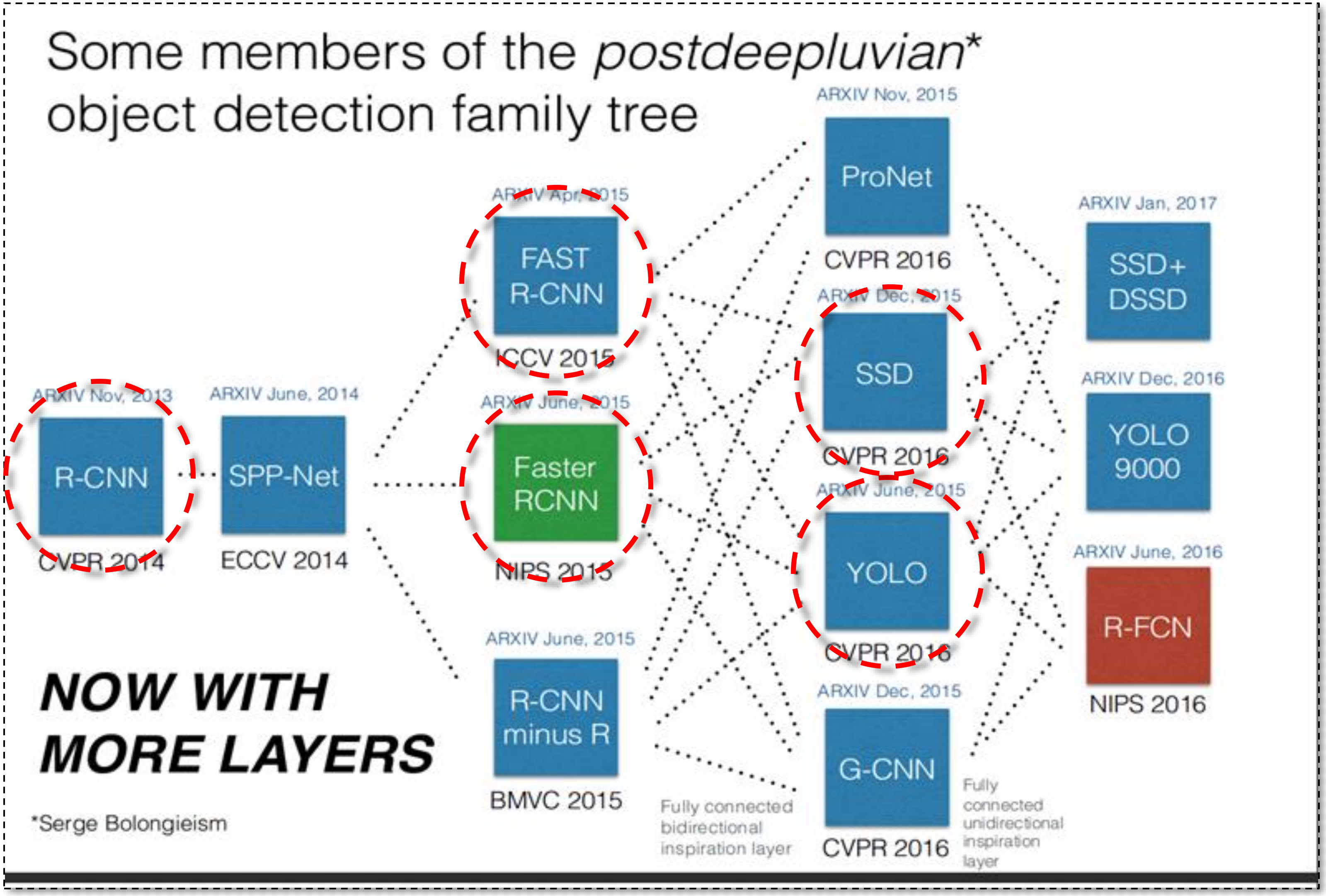
https://arxiv.org/abs/1512.02325

Tuning Anchor Boxes using Priors



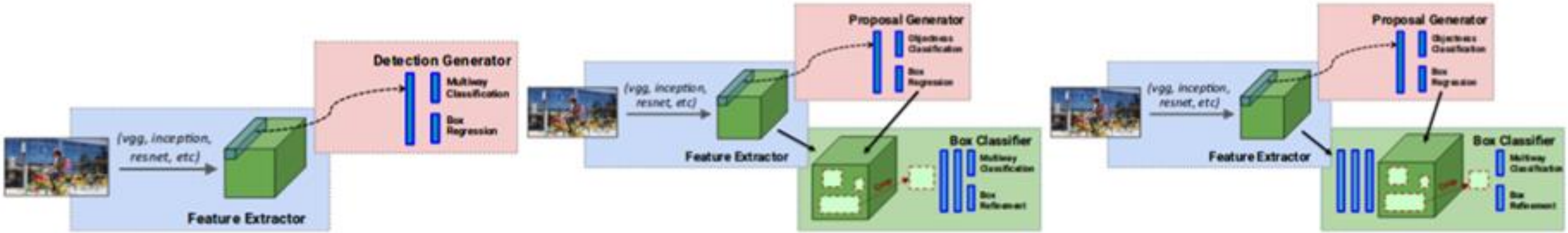
Most important hyperparameter in tuning **Single Stage Object Detectors** and **Region Proposal Networks**

Object Detection



https://www.robots.ox.ac.uk/~vgg/rg/slides/vgg_rg_16_feb_2017_rfcn.pdf

Single Stage vs Two-Stage Object Detectors



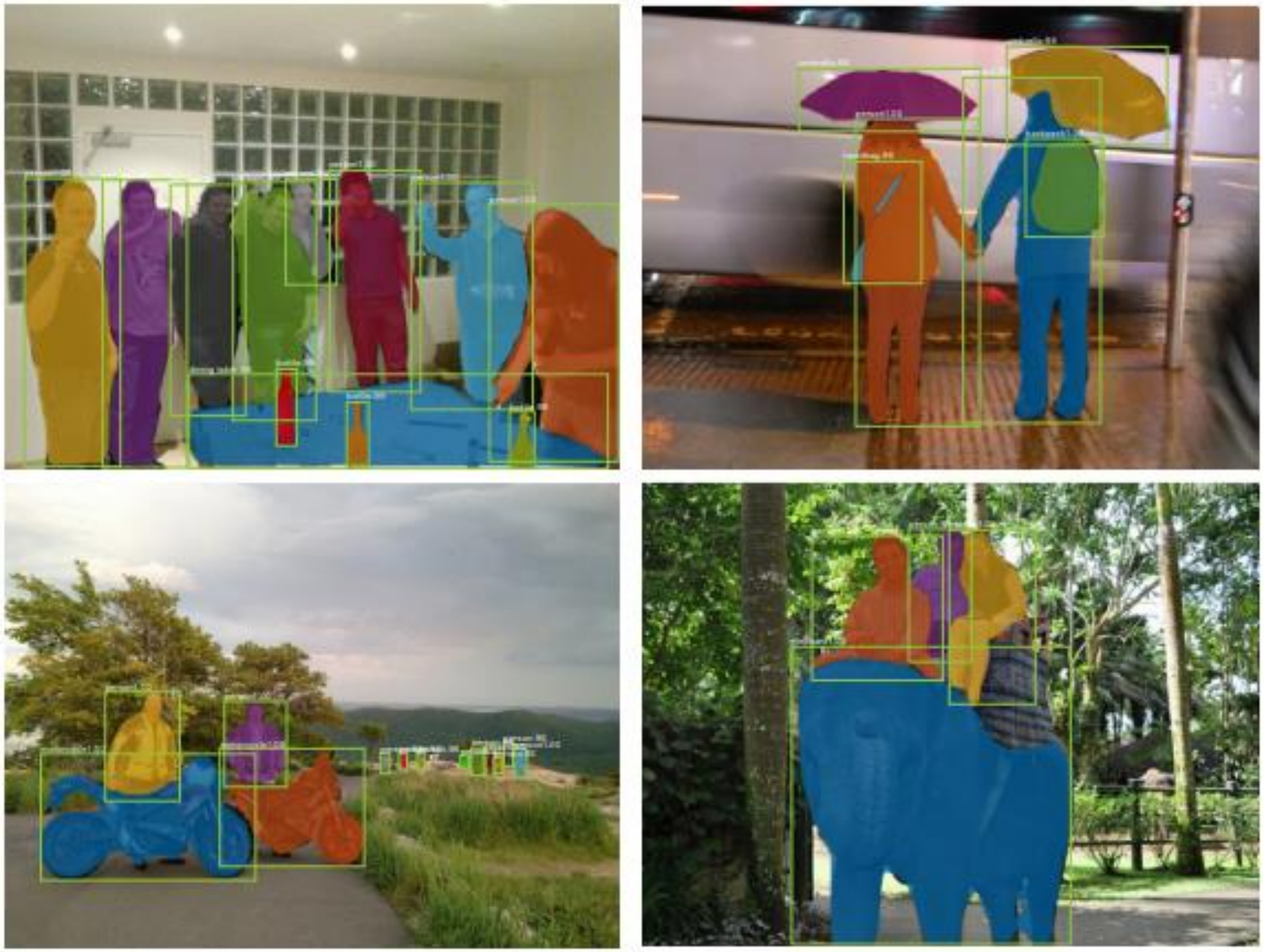
(a) SSD. (b) Faster RCNN. (c) R-FCN.

Speed/accuracy trade-offs for modern convolutional object detectors
https://arxiv.org/abs/1611.10012

Why still use a two-stage object detector?

- Better recall of RPN as compared to SSD/YOLO
  - Trained with all object instances
  - Generic first stage, usable for multitask
- Finer control over training classifier
  - Custom minibatch (sampling 3:1 negative samples)
- Instance-level multitask (Mask-RCNN)

Mask R-CNN – Towards Instance-Level Understanding



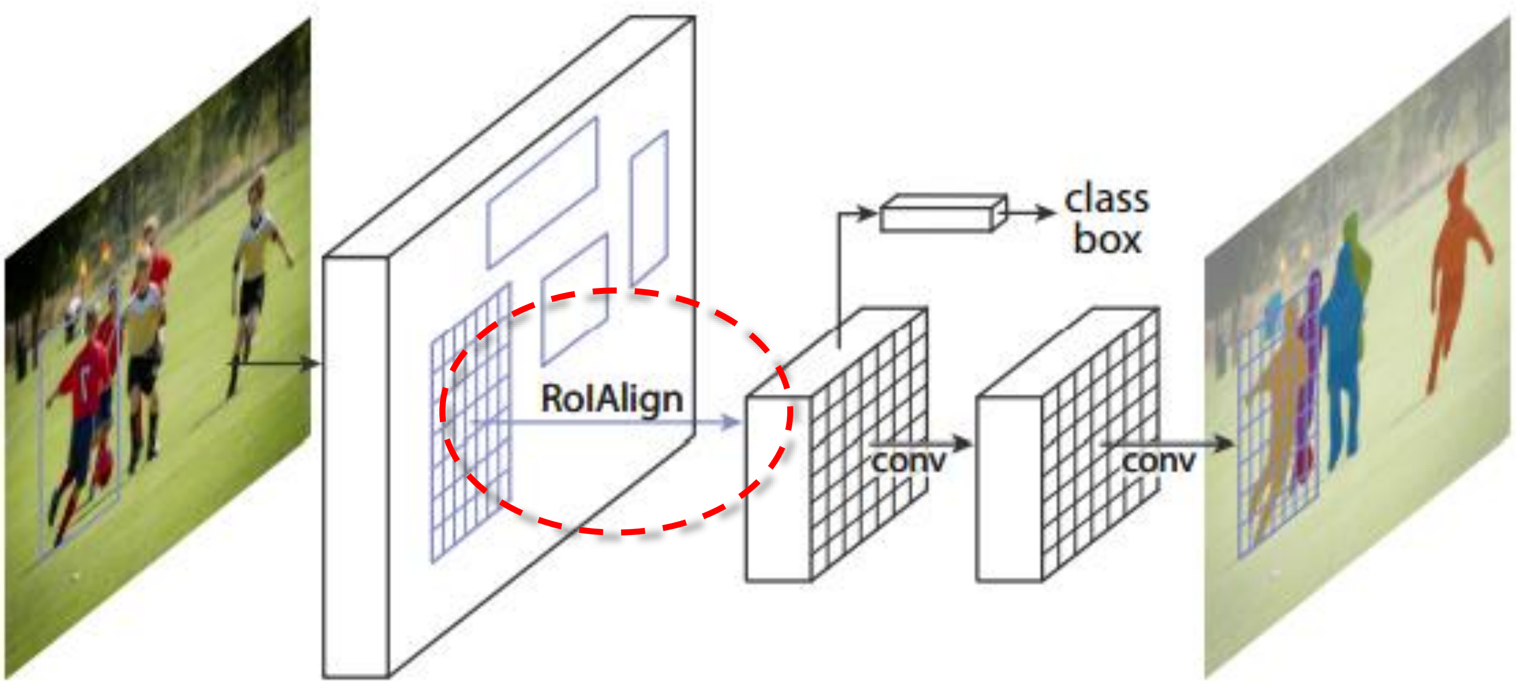
https://arxiv.org/abs/1703.06870

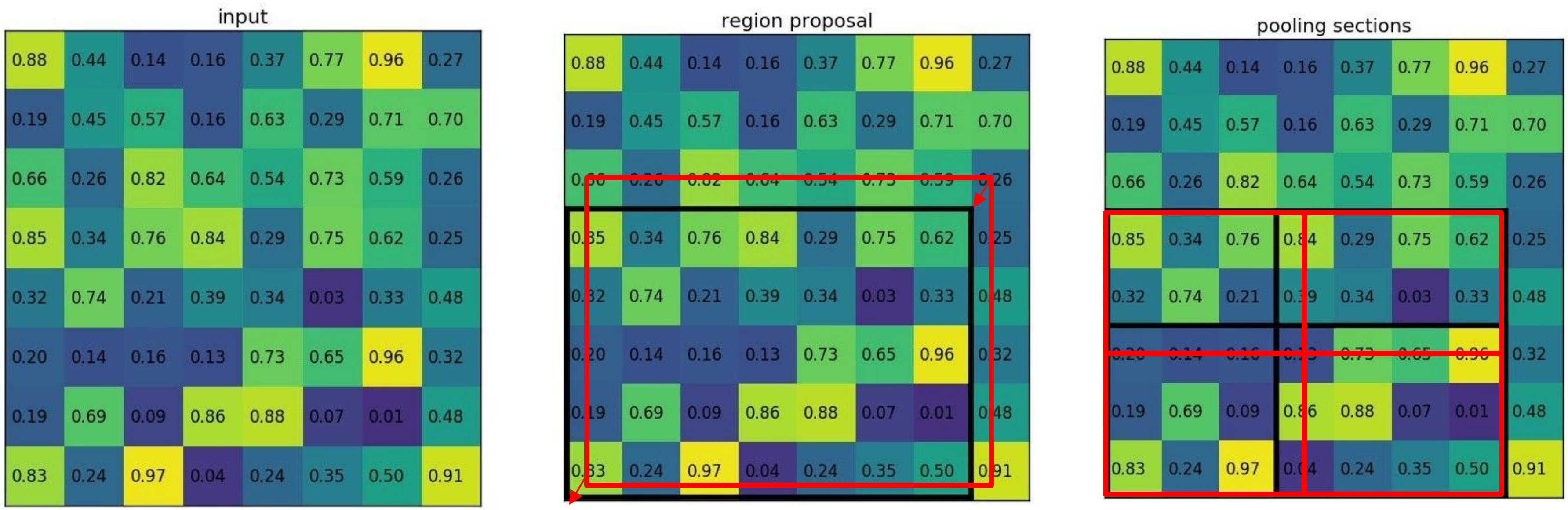
Mask R-CNN – Towards Instance-Level Understanding

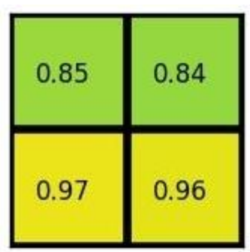
Zoom in on instances

Mask R-CNN

Preserves pixel-to-pixel alignment





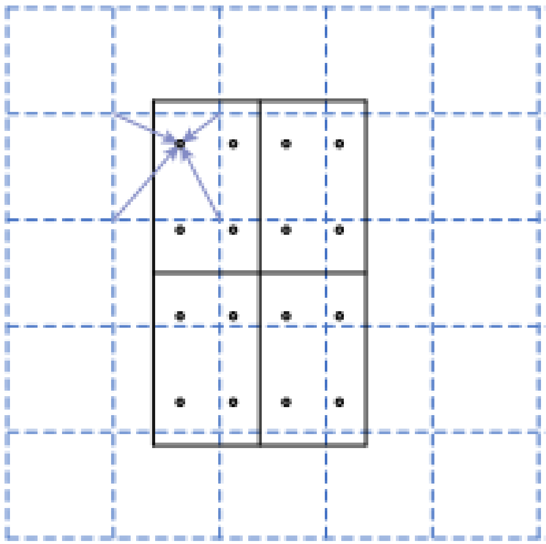


Quantization – loss of pixel-to-pixel alignment

https://deepsense.ai/region-of-interest-pooling-explained/

ROI Align

Improvement on ROI Pooling



- Input: Feature map (5x5 here) and region proposal (normalized float coordinates)
- Output: 2x2 'pooled' bins
- Sample 4 points in every bin uniformly
- Compute value at each bin using bilinear interpolation
- Max or average the 4 bins

Class Imbalance in Training a Classifier

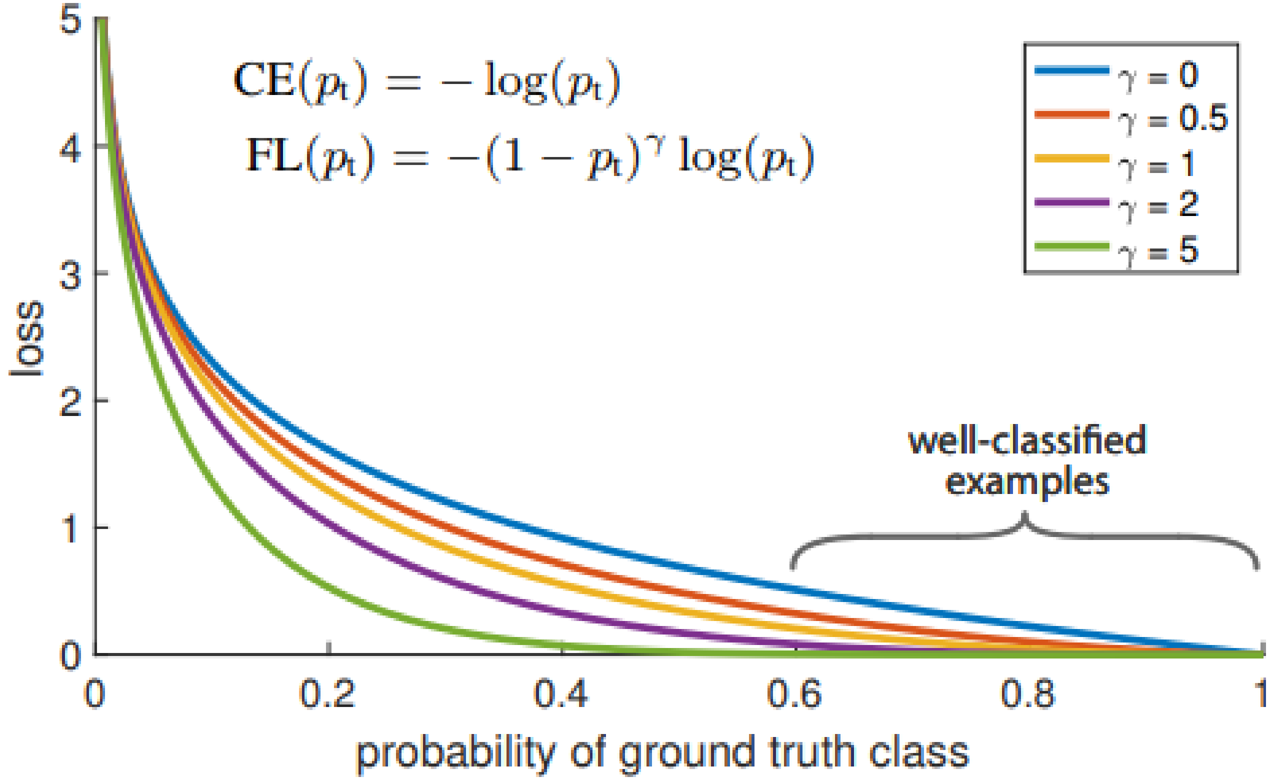
- While training detectors, maximum samples are background (negatives)
- Faster R-CNN: Ratio of 3 negatives to 1 positive is maintained while training classifier head → Custom minibatch
- Not easy in single stage detectors

Class Imbalance in Training a Classifier

- Cross entropy loss $$\mathrm{CE}(p_{\mathrm{t}}) = -\log(p_{\mathrm{t}})$$
- Balanced cross entropy loss $$\mathrm{CE}(p_{\mathrm{t}}) = -\alpha_{\mathrm{t}} \log(p_{\mathrm{t}})$$
- Focal Loss $$\mathrm{FL}(p_{\mathrm{t}}) = -(1 - p_{\mathrm{t}})^{\gamma} \log(p_{\mathrm{t}})$$

https://arxiv.org/abs/1708.02002

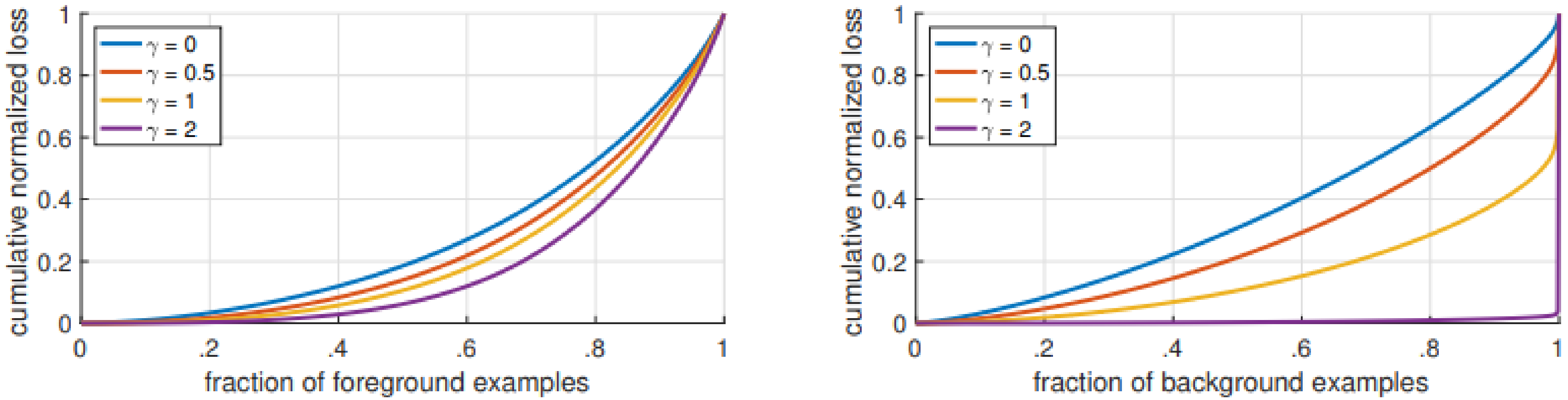
Focal Loss



$\mathrm{CE}(p_t) = -\log(p_t)$
$\mathrm{FL}(p_t) = -(1-p_t)^\gamma \log(p_t)$
$\gamma = 0$
$\gamma = 0.5$
$\gamma = 1$
$\gamma = 2$
$\gamma = 5$
well-classified examples
loss
probability of ground truth class
0
1
2
3
4
5
0
0.2
0.4
0.6
0.8
1

Focal Loss

$$\mathrm{FL}(p_t) = -(1-p_t)^\gamma \log(p_t)$$



Multitask Network for Autonomous Driving

- Common Feature Extractor or Core Network
  - Fast network, < 20ms for 2MP image – Mobilenet V2, Shufflenet V2, InceptionV1
  - Feature Pyramid Network / RetinaNet architecture
- Single stage detector heads for different tasks (*things*)
  - Fork from core network at different points for different tasks
  - Handling high variation in scale and aspect ratio
  - Different anchors for traffic lights, pedestrians, cars
  - Tuning anchor box and network strides using clustering
- Semantic segmentation head for *stuff* – roads, buildings, etc
- ROI-heads for zooming in and understanding objects in detail
- Efficient hardware implementation
  - INT8 conversion and calibration
  - Interfacing with fusion and tracking, finally control
  - C++ runnables

max planck institut informatik

NYU | $C_o u(r) a_n(t)$

IIT Bombay

Thank you!

http://cse.iitb.ac.in/~ajain/