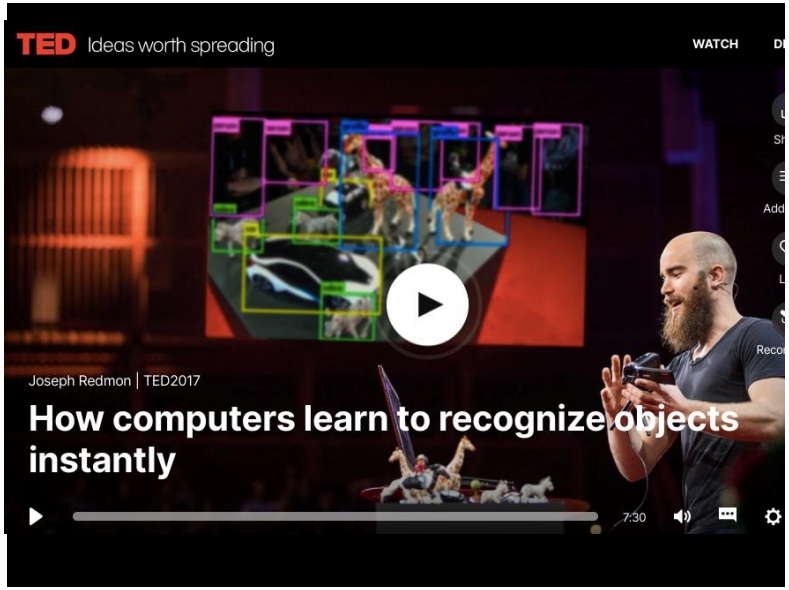
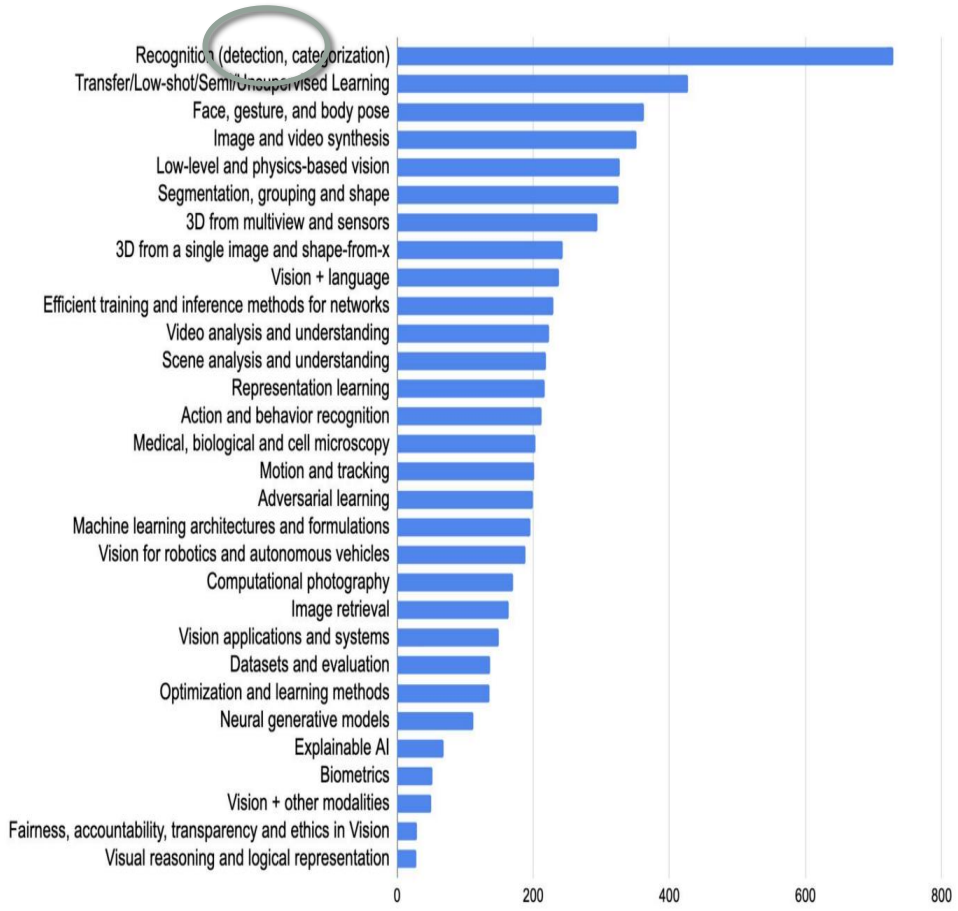
Computational vision: object detection

Petia Radeva



CVPR'2019 vs 2022



MENU | Select Primary Subject Area | # papers | Oral | Poster

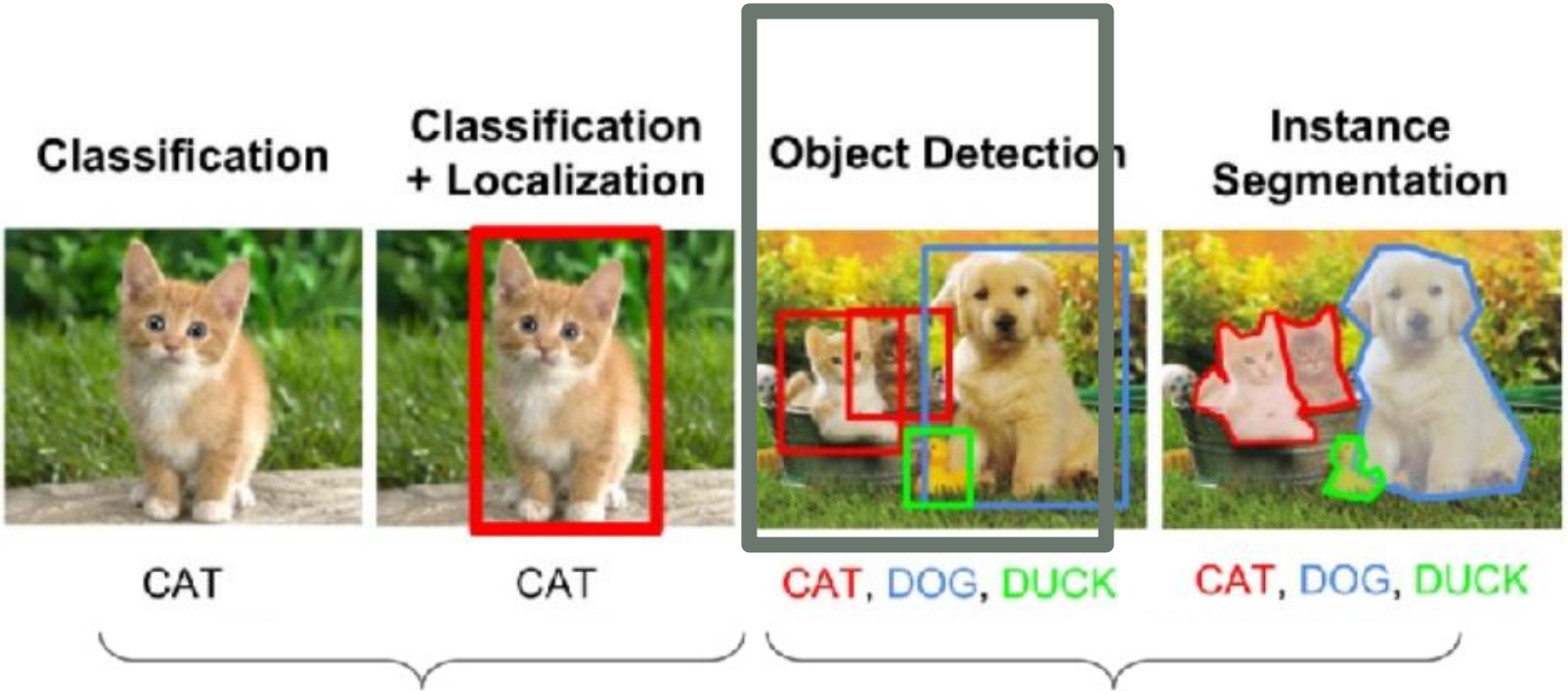
1	Recognition: detection, categorization, retrieval	177	25	152
2	Image and video synthesis and generation	157	26	131
3	3D from multi-view and sensors	137	26	111
4	Low-level vision	110	19	91
5	Vision + language	105	20	85
6	Segmentation, grouping and shape analysis	99	16	83
7	Transfer/ low-shot/ long-tail learning	86	15	71
8	Deep learning architectures and techniques	85	20	65
9	Self-& semi-& meta- & unsupervised learning	84	7	77
10	Video analysis and understanding	77	15	62
11	Pose estimation and tracking	62	14	48
12	Representation learning	61	11	50
13	3D from single images	60	10	50
14	Scene analysis and understanding	56	9	47
15	Face and gestures	54	7	47
16	Computational photography	53	10	43
17	Motion and tracking	53	8	45
18	Adversarial attack and defense	52	10	42
19	Datasets and evaluation	52	7	45
20	Machine learning	41	7	34
21	Action and event recognition	40	8	32
22	Efficient learning and inferences	40	3	37
23	Medical, biological and cell microscopy	37	5	32
24	Vision applications and systems	32	4	28
25	Navigation and autonomous driving	31	2	29
26	Vision + graphics	24	6	18
27	Privacy and federated learning	21	3	18
28	Vision + X	20	4	16
29	Physics-based vision and shape-from-X	16	2	14
30	Robot vision	16	3	
31	Explainable computer vision	15	2	13
32	Demo	15		15
33	Optimization methods	14	2	12
34	Transparency, fairness, accountability, privacy and ..	14	4	
35	Document analysis and understanding	12		12
36	Biometrics	11	2	9

Categories at CVPR 2022 ranked by number of papers accepted

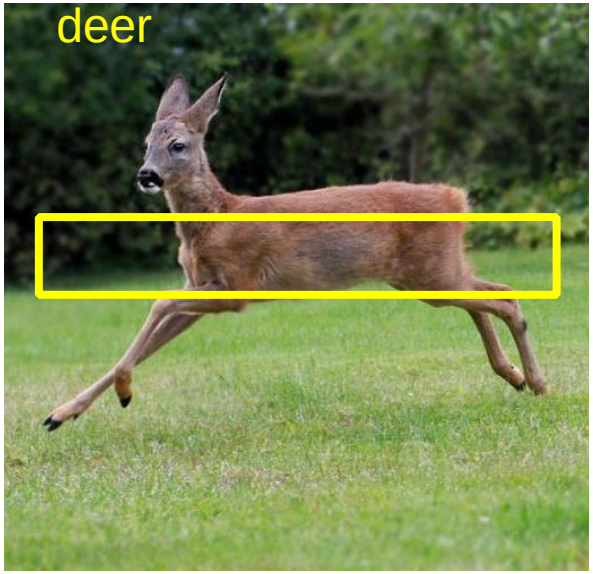
The concept of Object detection

Image classification
(what? *I don't care where*)

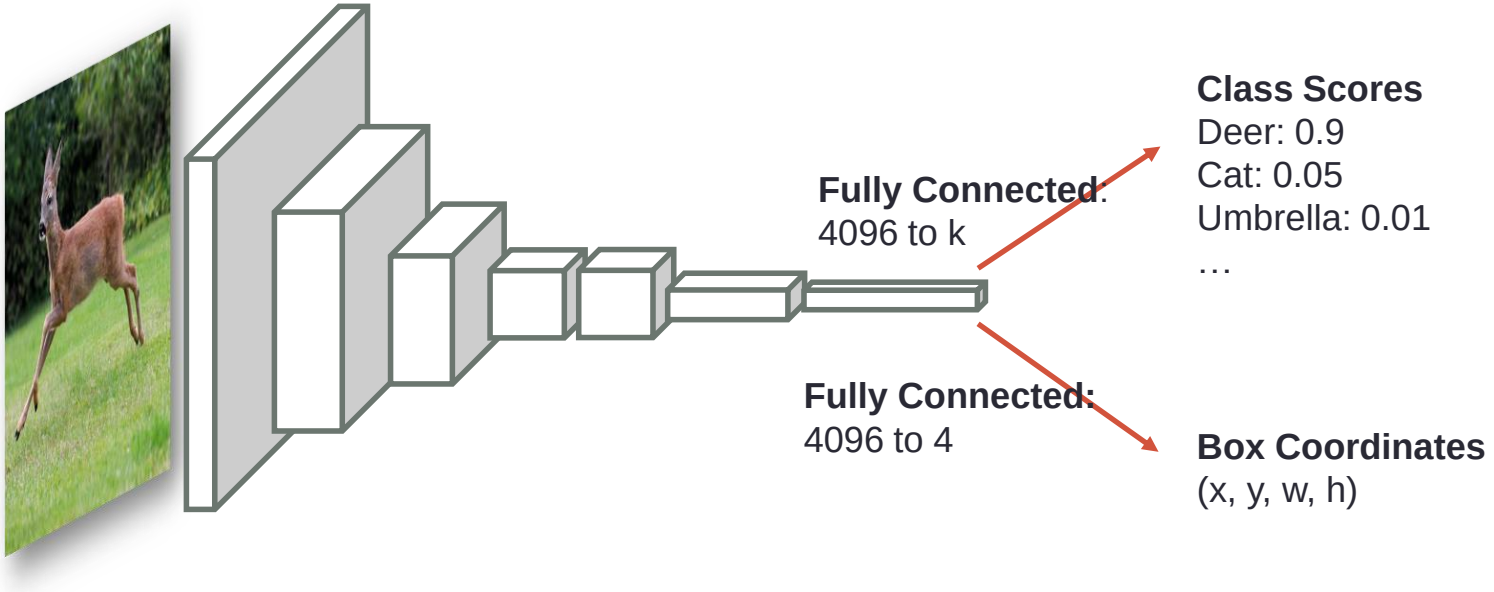
Object detection
(what + where?)



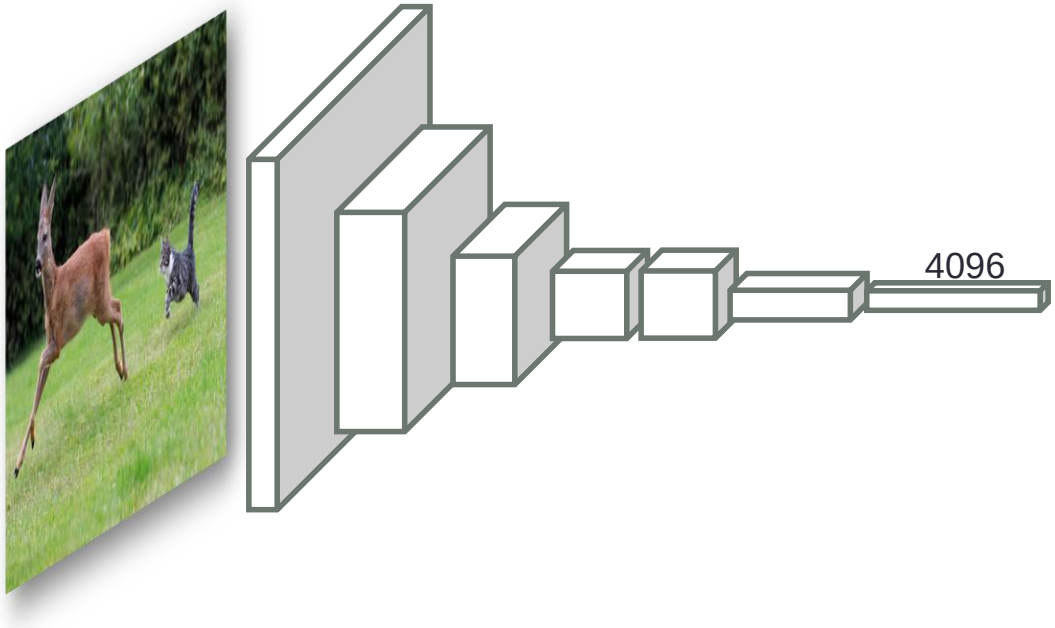
Object Detection



Object Detection

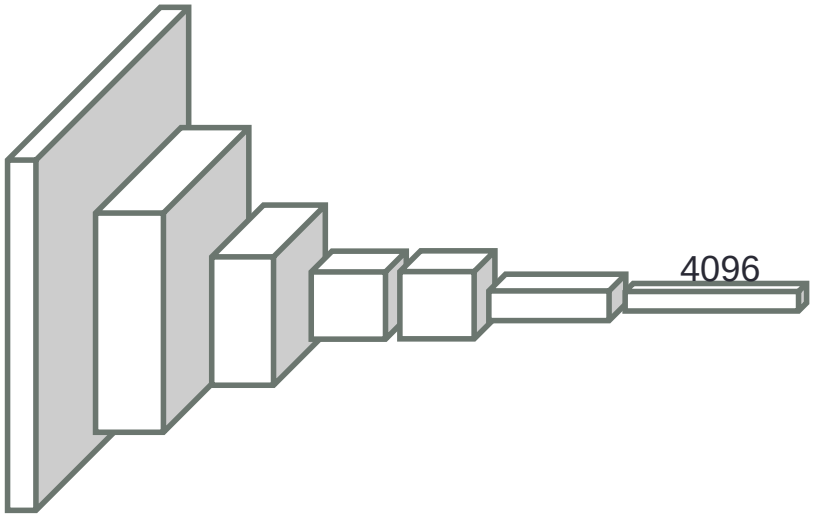


Object Detection



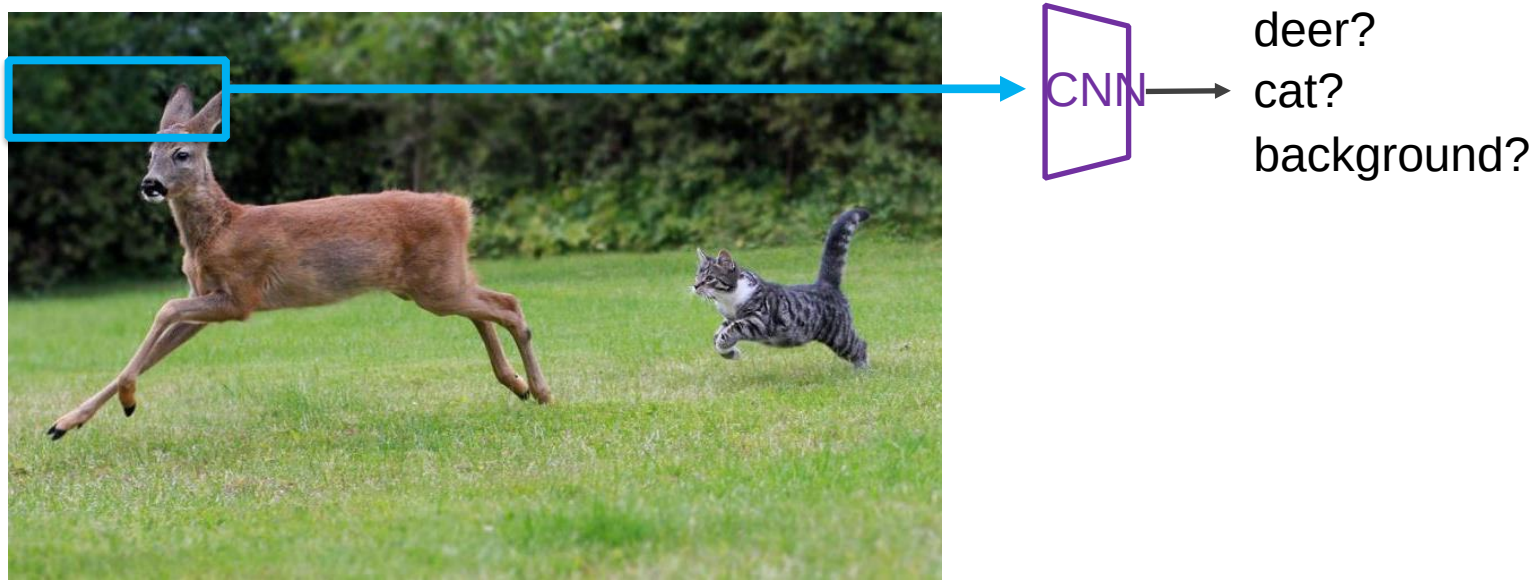
Deer: (x, y, w, h)
Cat: (x, y, w, h)

Object Detection

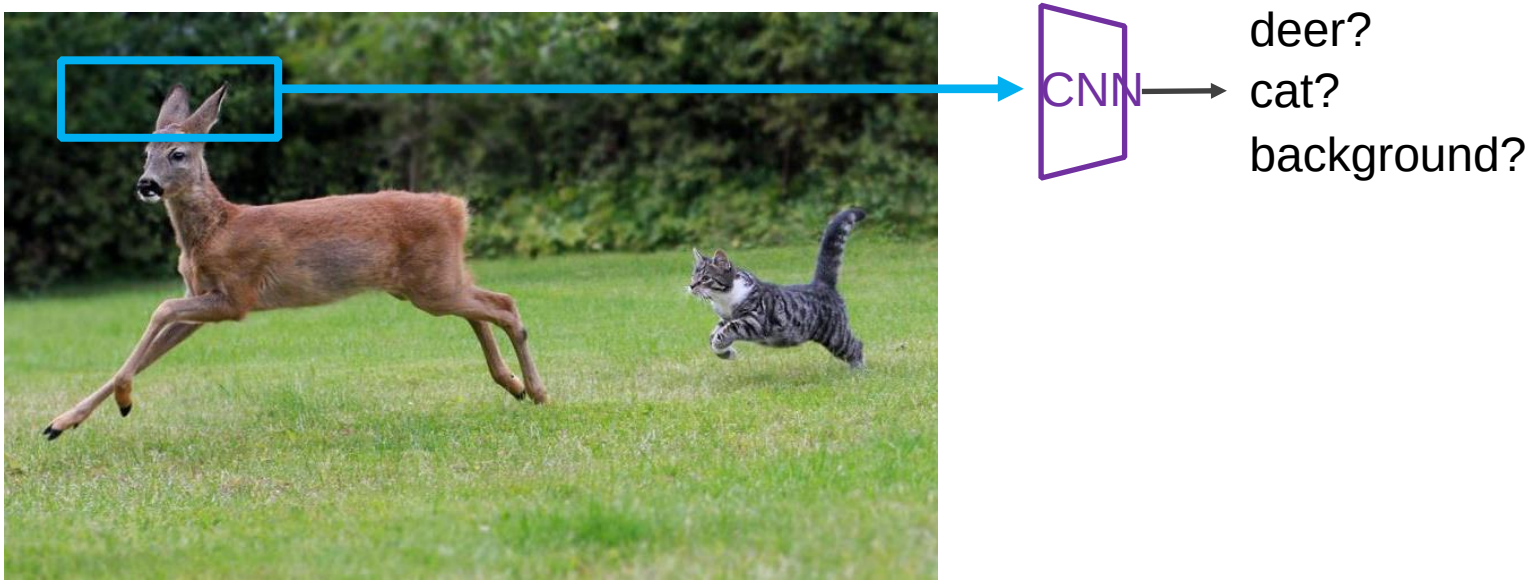


Penguin: (x, y, w, h)
Penguin: (x, y, w, h)
Penguin: (x, y, w, h)
Penguin: (x, y, w, h)
…

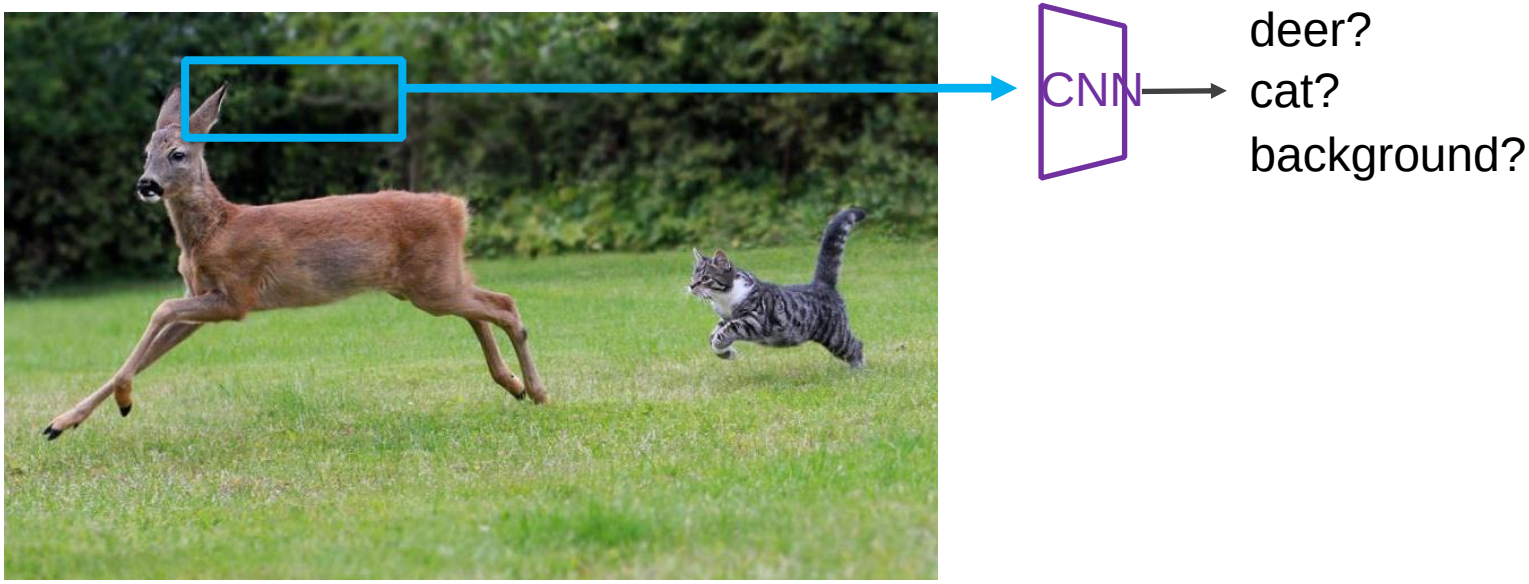
Object Detection as Classification



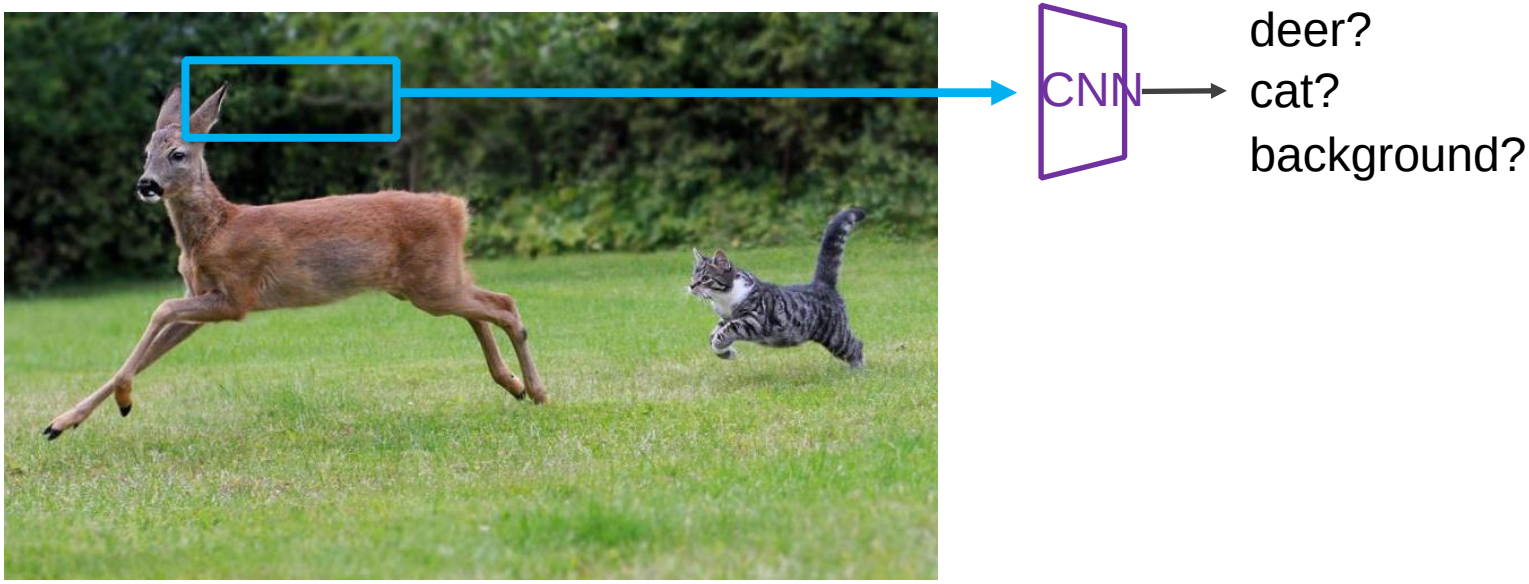
Object Detection as Classification



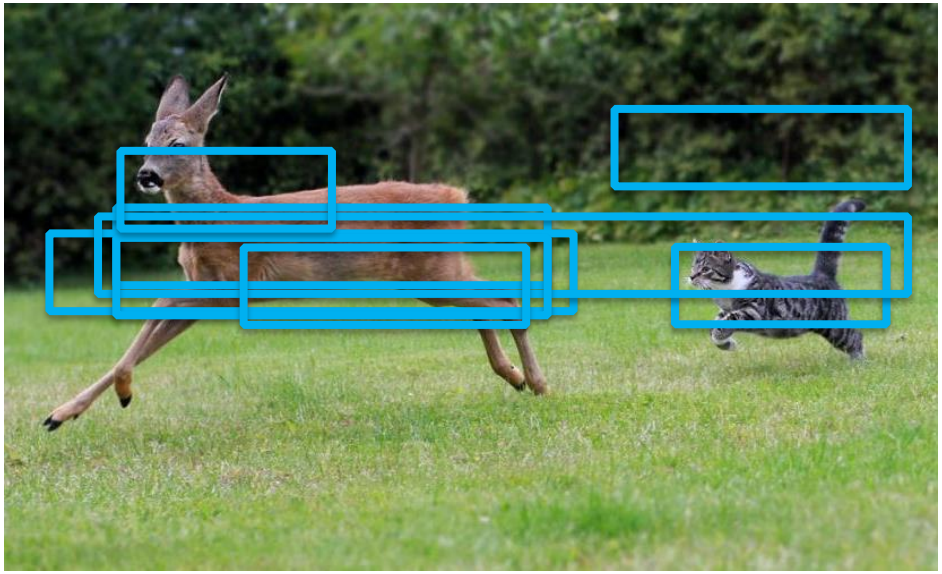
Object Detection as Classification



Object Detection as Classification with Sliding Window



Object Detection as Classification with Box Proposals



Object detection today is based on Deep learning

He, Gkioxari, Dollár, Girshick. Mask R-CNN. In ICCV 2017

Datasets for Object detection (2014)

- **80 object categories**
- **330K images (>200K labelled)**
- Recognition in context
- Object segmentation
- Super-pixel stuff segmentation
- 1.5 million object instances
- 91 stuff categories
- 5 captions per image
- 250,000 people with key-points

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

MS COCO

COCO (Microsoft Common Objects in Context)

Edit

Introduced by Lin et al. in Microsoft COCO: Common Objects in Context

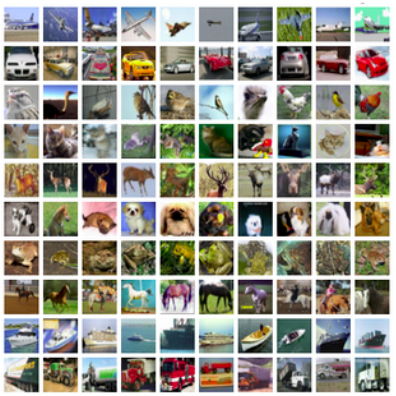
The MS **COCO** (**Microsoft Common Objects in Context**) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.

Splits: The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.

Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.

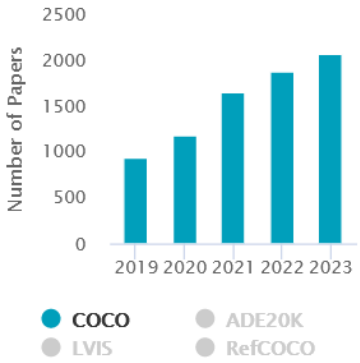
Annotations: The dataset has annotations for

- object detection: bounding boxes and per-instance segmentation masks with 80 object categories,
- captioning: natural language descriptions of the images (see MS COCO Captions),
- keypoints detection: containing more than 200,000 images and 250,000 person instances labeled with keypoints (17 possible keypoints, such as left eye, nose, right hip, right ankle),
- stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff),
- panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),
- dense pose: more than 39,000 images and 56,000 person instances labeled with DensePose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images.



Source: https://cocodataset.org/.

Usage



Paperswithcode

Object Detection

Edit

3382 papers with code • 82 benchmarks • 245 datasets

Object detection is the task of detecting instances of objects of a certain class within an image. The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN and Cascade R-CNN.

The most popular benchmark is the MSCOCO dataset. Models are typically evaluated according to a Mean Average Precision metric.

(Image credit: Detectron)

Benchmarks

Add a Result

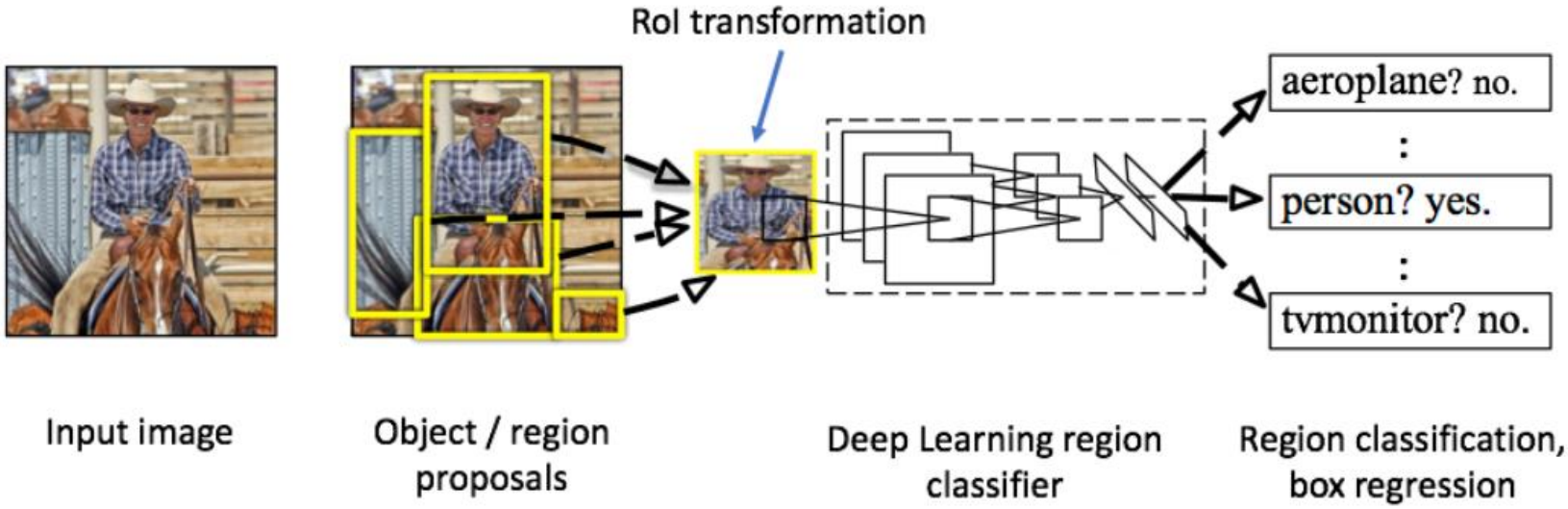
These leaderboards are used to track progress in Object Detection

Trend	Dataset	Best Model	Paper	Code	Compare
	COCO test-dev	InternImage-DCNv3-H			See all
	COCO minival	InternImage-DCNv3-H			See all
	PASCAL VOC 2007	Cascade Eff-B7 NAS-FPN (Copy Paste pre-training, single-scale)			See all
	CPPE-5	TridentNet			See all

Content

2014: R-CNN - An Early Application of CNNs to Object Detection

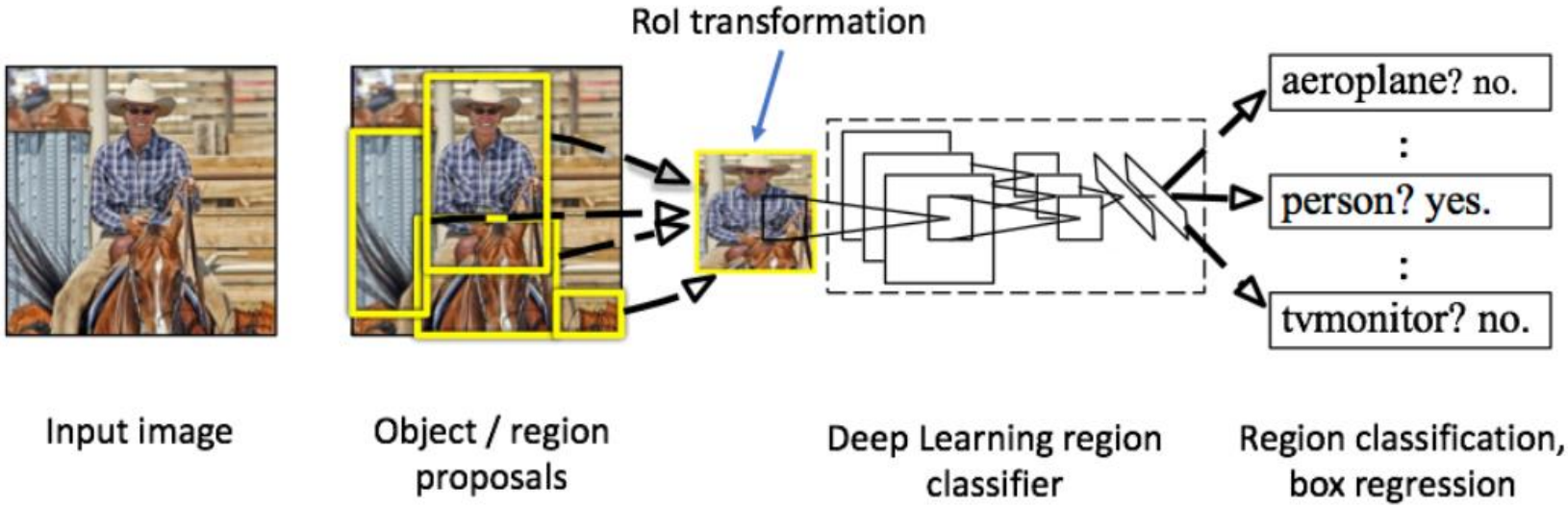
To what extent do CNN generalize to object detection?



Recognition using regions paradigm

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

2014: R-CNN - An Early Application of CNNs to Object Detection



R-CNN combines region proposals with CNNs:

(1) applying **high-capacity convolutional neural networks** (CNNs) to bottom-up region proposals in order to localize objects,

(2) when **labelled training data is scarce**, supervised pre-training for an auxiliary task followed by domain-specific fine-tuning, yields a significant performance boost.

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

R-CNN

First stage: generate category-independent region proposals.

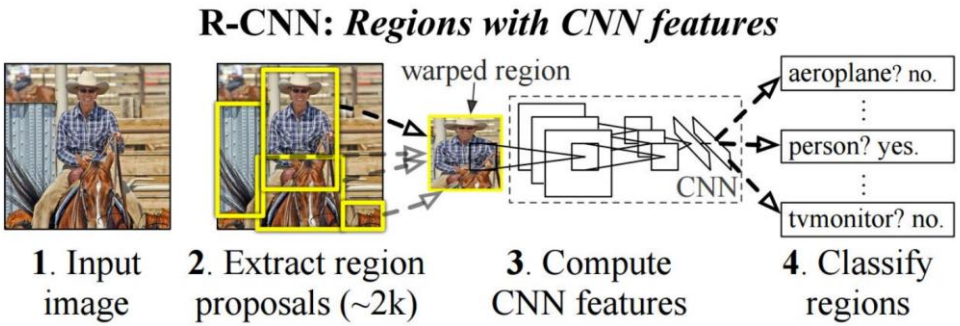
- 2000 Region proposals for every image

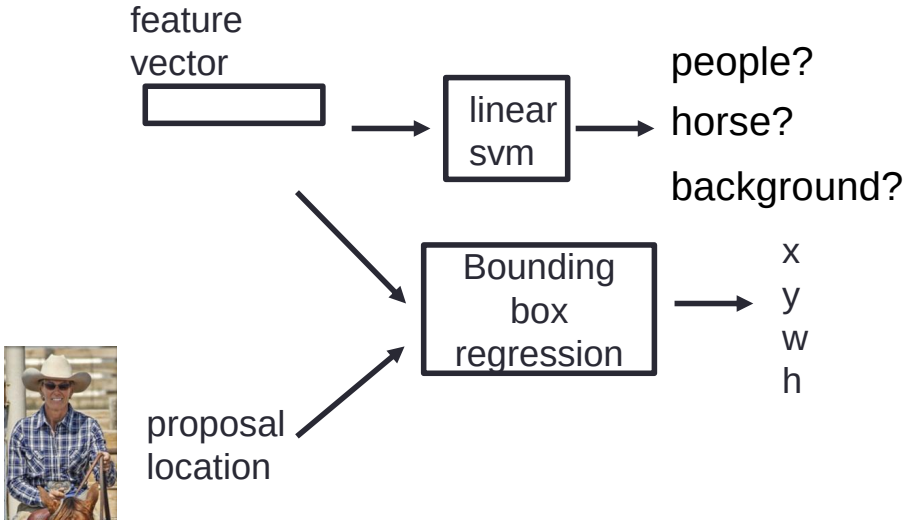
Second stage: extracts a fixed-length feature vector from each region.

- a 4096-dimensional feature vector from each region proposal

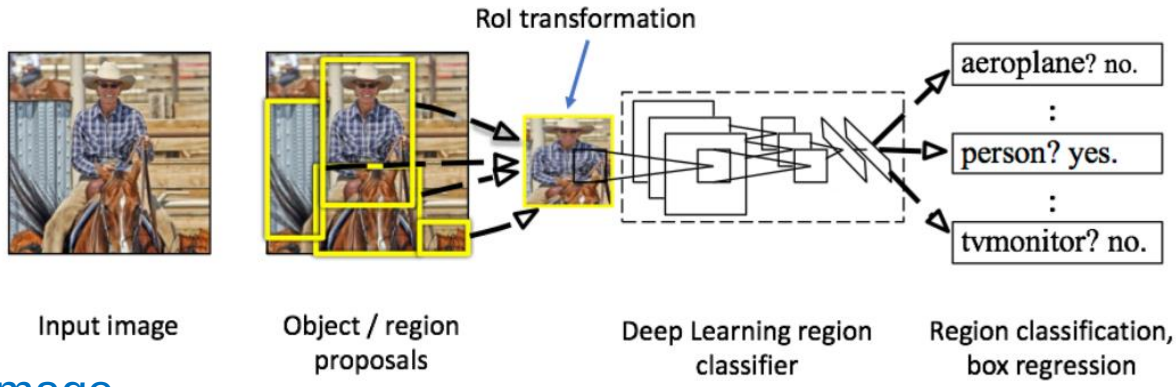
Third stage: a set of class-specific linear SVMs.

- object category and location





2014: R-CNN - An Early Application of CNNs to Object Detection



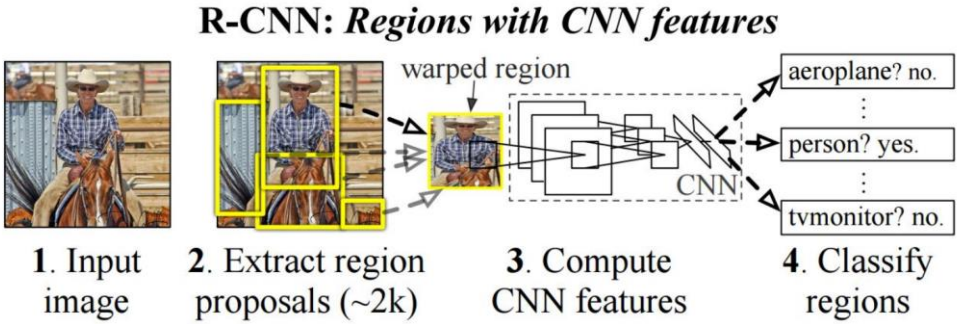
- **Inputs**: Image
- **Outputs**: Bounding boxes + labels for each object in the image.

Object detection system overview. The system:

- (1) takes an input image,
- (2) extracts around 2000 bottom-up region proposals,
- (3) computes features for each proposal using a large convolutional neural network (CNN), and
- (4) classifies each region using class-specific linear SVMs.

R-CNN

First stage: generate category-independent region proposals.



- **2000 Region proposals** for every image

Affine image warping to compute a fixed-size CNN output (4096-dim) from each region proposal, regardless of the region's shape.

Selective Search for Object Recognition

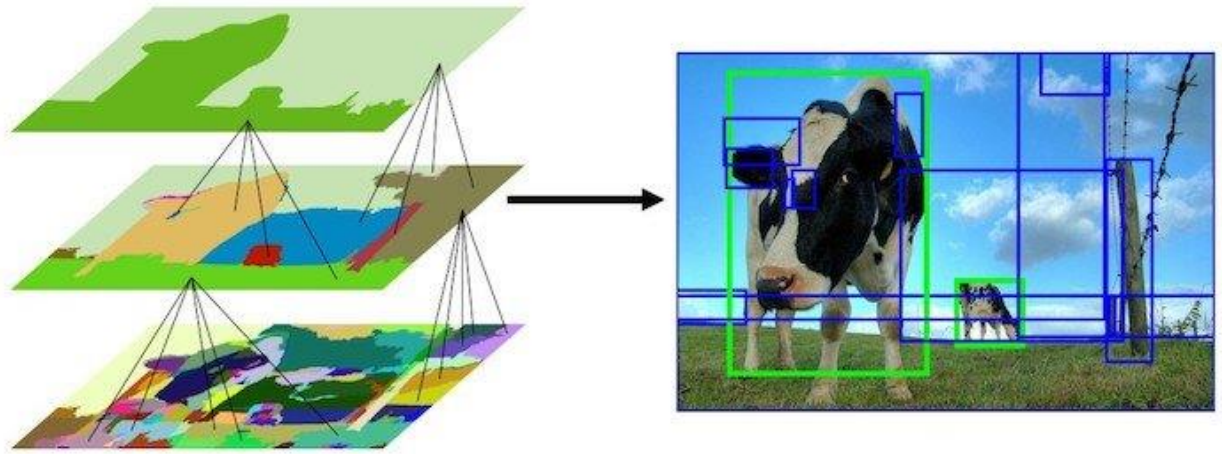
Original Image

Segmented Image

Oversegmented Image

- Region proposal algorithm used in object detection.
- Fast with a very high recall.
- Based on computing hierarchical grouping of similar regions based on colour, texture, size and shape compatibility.

Selective Search for Object Recognition



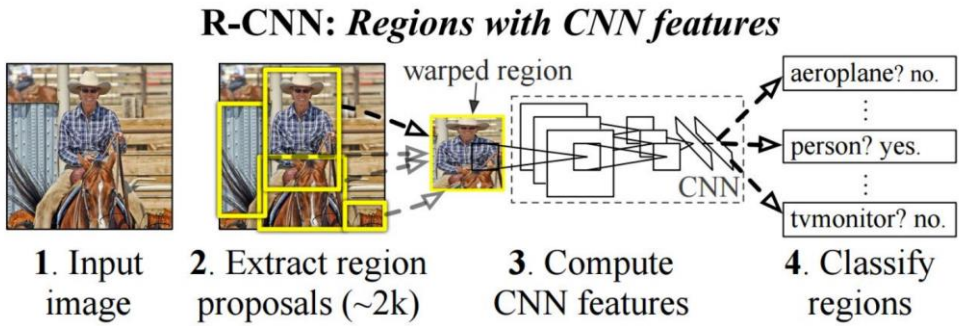
“Hierarchical” segmentations using Felzenszwalb and Huttenlocher’s oversegments.

Selective Search algorithm takes these oversegments as initial input and performs:

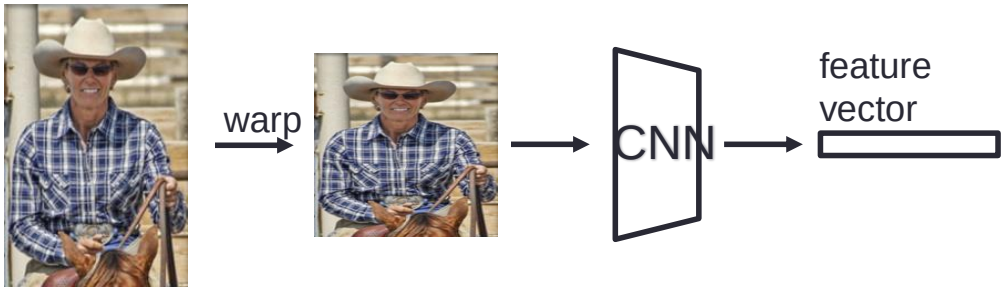
1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1

At each iteration, larger segments are formed and added to the list of region proposals in a bottom-up approach.

R-CNN



Second stage: extracts a fixed-length feature vector from each region.

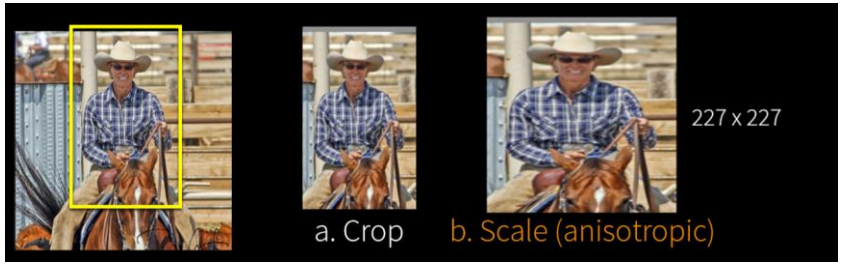


- a 4096-dimensional feature vector from each region proposal

Arbitrary rectangles?
A fixed size input?
227 x 227

5 conv layers + 2 fully connected layers

R-CNN: Object proposal transformations



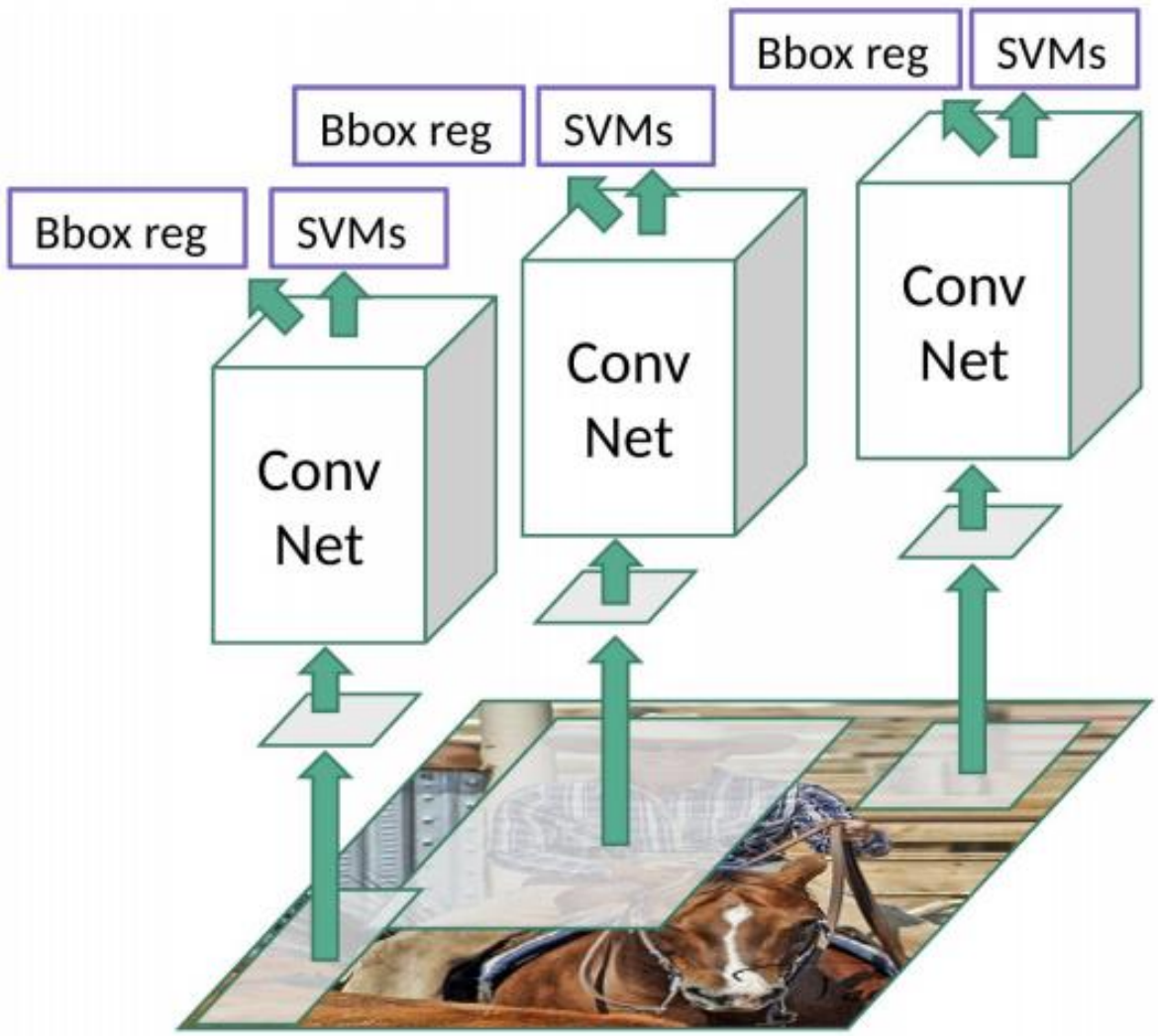
Convolutional neural network requires a fixed-size input of 227 × 227 pixels (Pre-trained on Imagenet for image classification)

Figure 7: Different object proposal transformations. (A) the original object proposal at its actual scale relative to the transformed CNN inputs; (B) tightest square with context; (C) tightest square without context; (D) warp. Within each column and example proposal, the top row corresponds to $p = 0$ pixels of context padding while the bottom row has $p = 16$ pixels of context padding.

Prior to warping, **dilate the tight bounding box** so that at the warped size there are exactly p pixels of warped image context around the original box (we use p = 16)

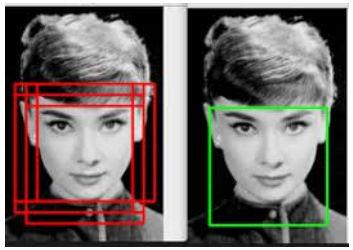
A pilot set of experiments showed that warping with context padding (p = 16 pixels) outperformed the alternatives by a large margin

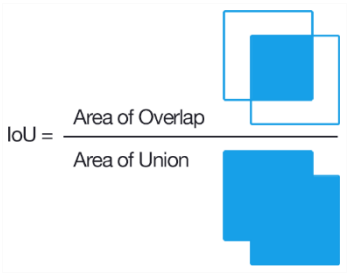
R-CNN: Third module – Feature classification/detection



R-CNN: Third module – Feature classification/detection

- Classify the feature of each region using category specific linear SVMs
- For each class, score each extracted feature vector using the SVM trained for that class.
- Given all scored regions in an image, apply a **greedy non-maximum suppression** (for each class independently) that rejects a region if it has an intersection-over union (IoU) overlap with a higher scoring selected region larger than a **learned threshold**.

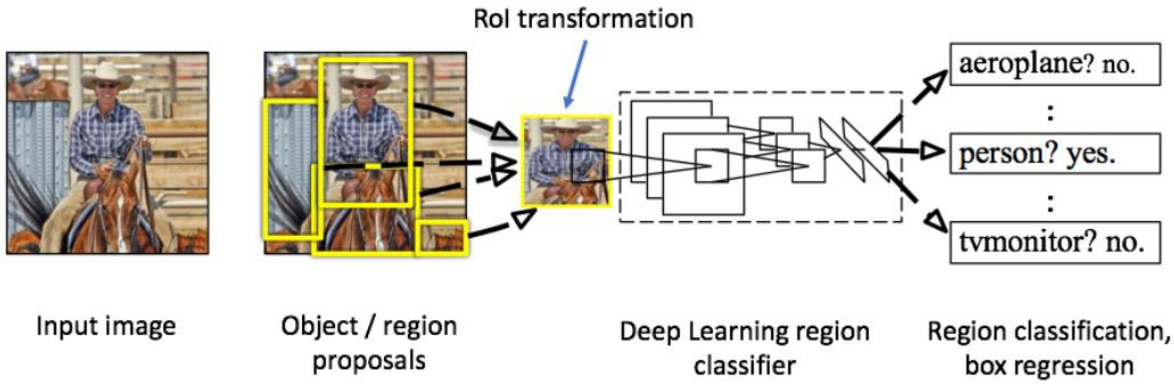




IoU overlap threshold, below which regions are defined as negatives examples for the SVM

Timing (detection): **47/s image on GPU**

R-CNN - An Early Application of CNNs to Object Detection (2014)



- **Inputs**: Image
- **Outputs**: Bounding boxes + labels for each object in the image.

Test time:
A simple **bounding box regression** method significantly reduces mislocalizations, which are the dominant error mode.

How to validate it?
What is the mean Average Precision (mAP)?

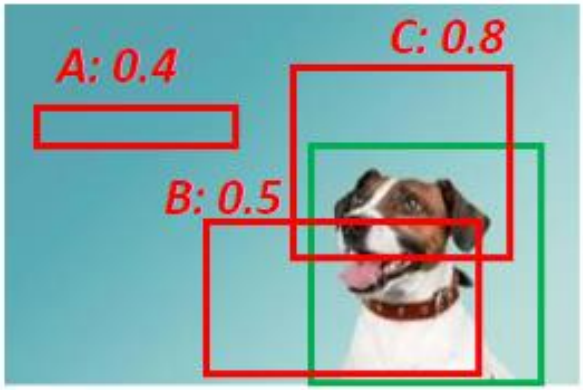


Image 1

Image 2

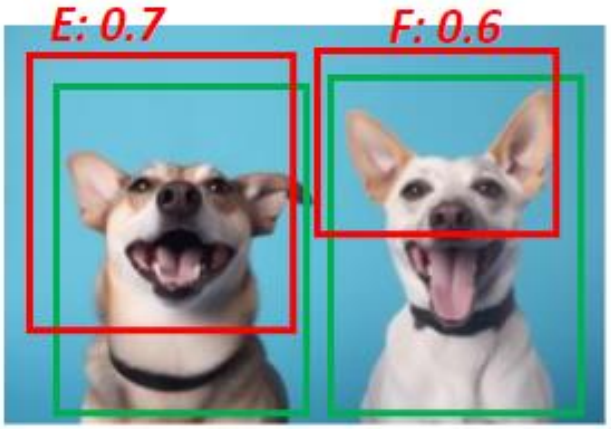
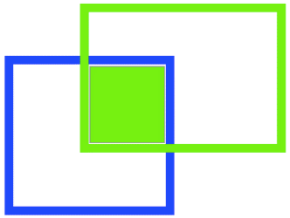


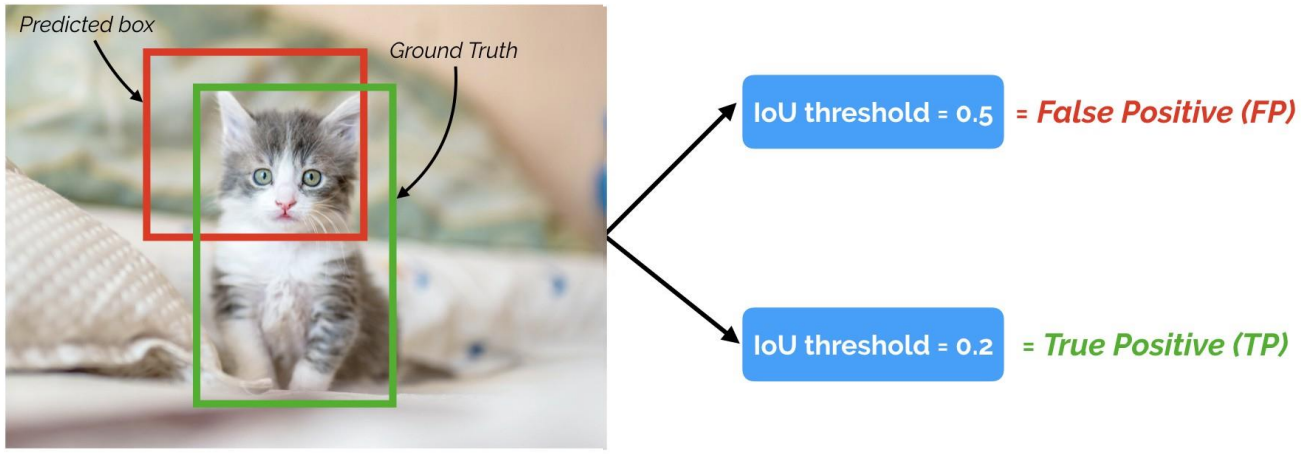
Image 3

How to validate it?
What is the mean Average Precision (mAP)?

- What parameters shell a detection depend on?
- Let us consider the Intersection of Union (IoU):



- Picking the right single threshold for the IoU metric seems arbitrary.
  - One researcher might justify a 60 percent overlap, and another is convinced that 75 percent seems more reasonable.
- **So why not having all of the thresholds considered in a single metric?**



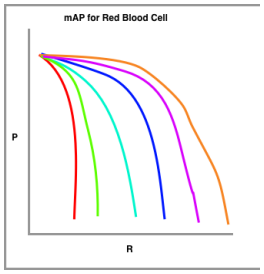
IoU for the prediction = ~0.3

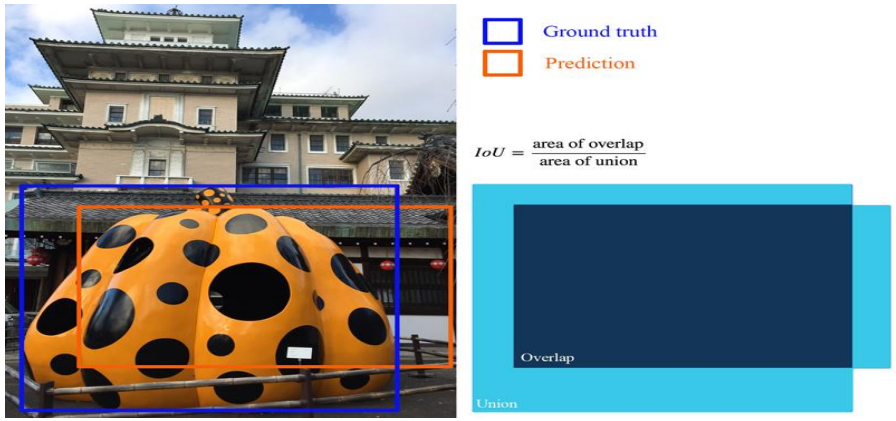
What is the mAP?

- Draw a precision recall curve by considering different confidence levels
- Draw a series of precision recall curves with the IoU threshold set at varying levels of difficulty (90%, 80%, etc).

$$\text{Precision} = \frac{tp}{tp + fp}$$

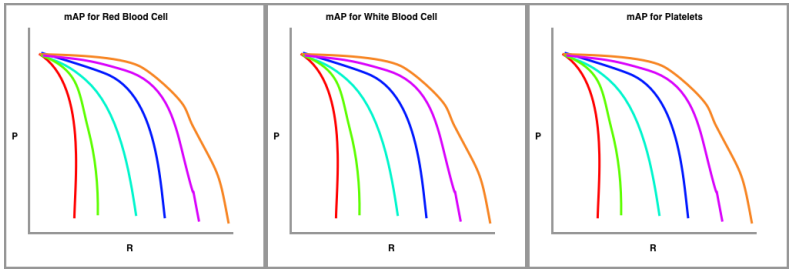
$$\text{Recall} = \frac{tp}{tp + fn}$$

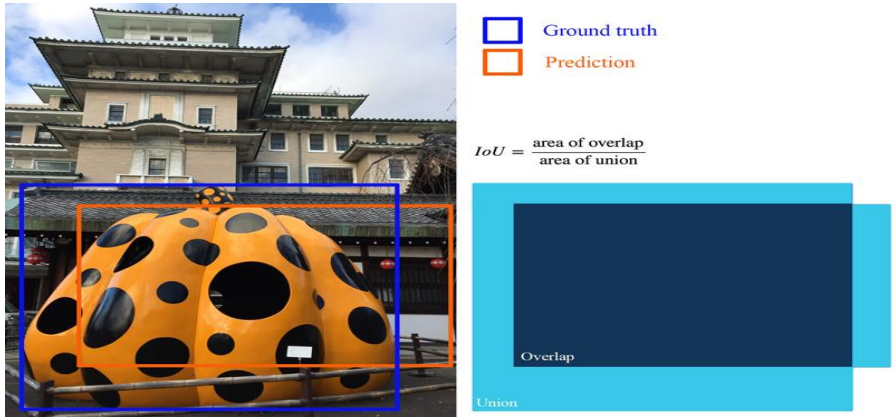




What is the mAP?

- Draw a precision recall curve by considering different confidence level
- Draw a series of precision recall curves with the IoU threshold set at varying levels of difficulty (90%, 80%, etc).





mAP metrics calculates the average precision (AP) for **each class individually** across all of the IoU thresholds.
Then the metric **averages the mAP for all classes** to arrive at the final estimate.

Problems with R-CNN

- It still takes **a huge amount of time to train and run** the network as you would have to **classify 2000 region** proposals per image.
- It **cannot be implemented real time** as it takes around 47 seconds for each test image.
- The selective search algorithm is a **fixed algorithm**. Therefore, **no learning is happening** at that stage. This could lead to the generation of bad candidate region proposals.
-

2015: Fast R-CNN - Speeding up and Simplifying R-CNN

R-CNN works really well, but is really quite slow for a few simple reasons:

- It requires a forward pass of the CNN (AlexNet) **for every single region proposal** for every single image (that's around 2000 forward passes per image!), without sharing computation.
- It has to ***train three different models*** separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train.
- **Intuition**: run the CNN just once per image and then find a way to share that computation across the ~2000 proposals

RCNN

RCNN

- Simple and scalable.
- Improves mAP.

- A multistage pipeline.

- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).

- Object detection is slow.

Fast-RCNN

?

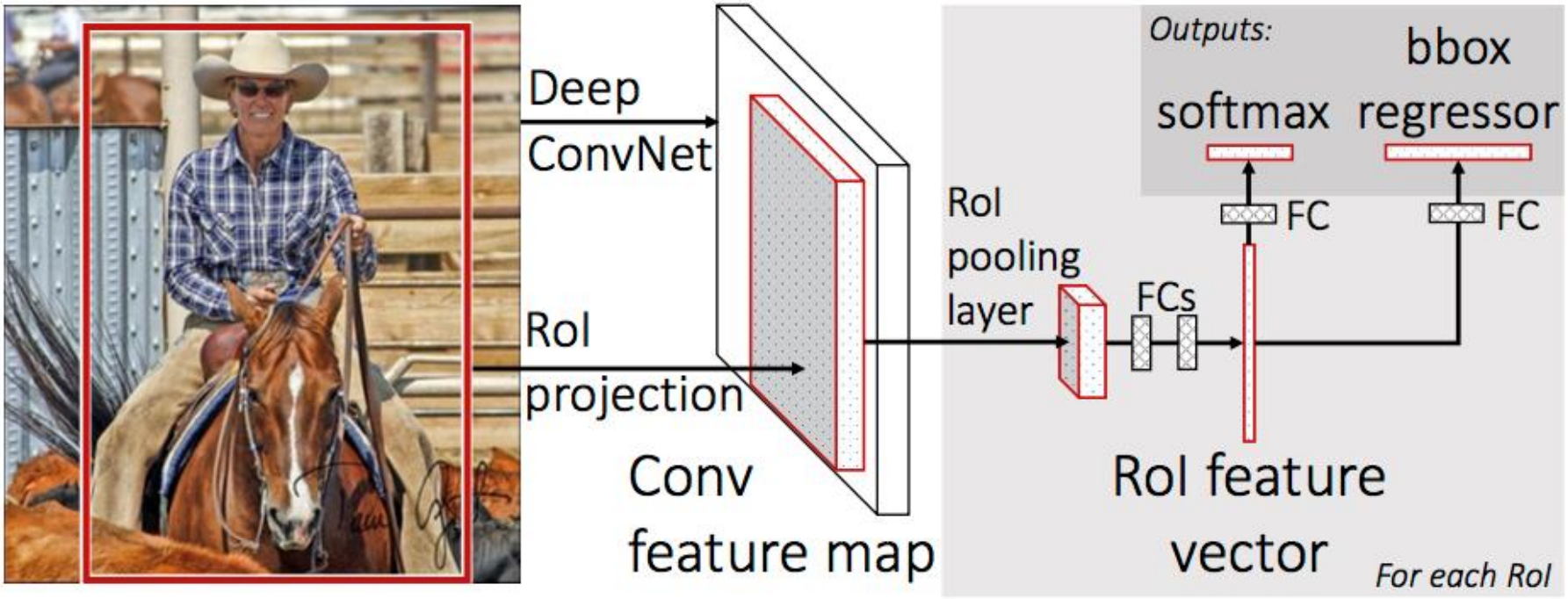
2015: Fast R-CNN - Speeding up and Simplifying R-CNN

Input: an entire image and a set of object proposals

- The network first processes the **whole image** with several convolutional (conv) and max pooling layers to produce a conv feature map.

- Then, for each object proposal a region of interest **(RoI) pooling layer** extracts a fixed-length feature vector from the feature map.

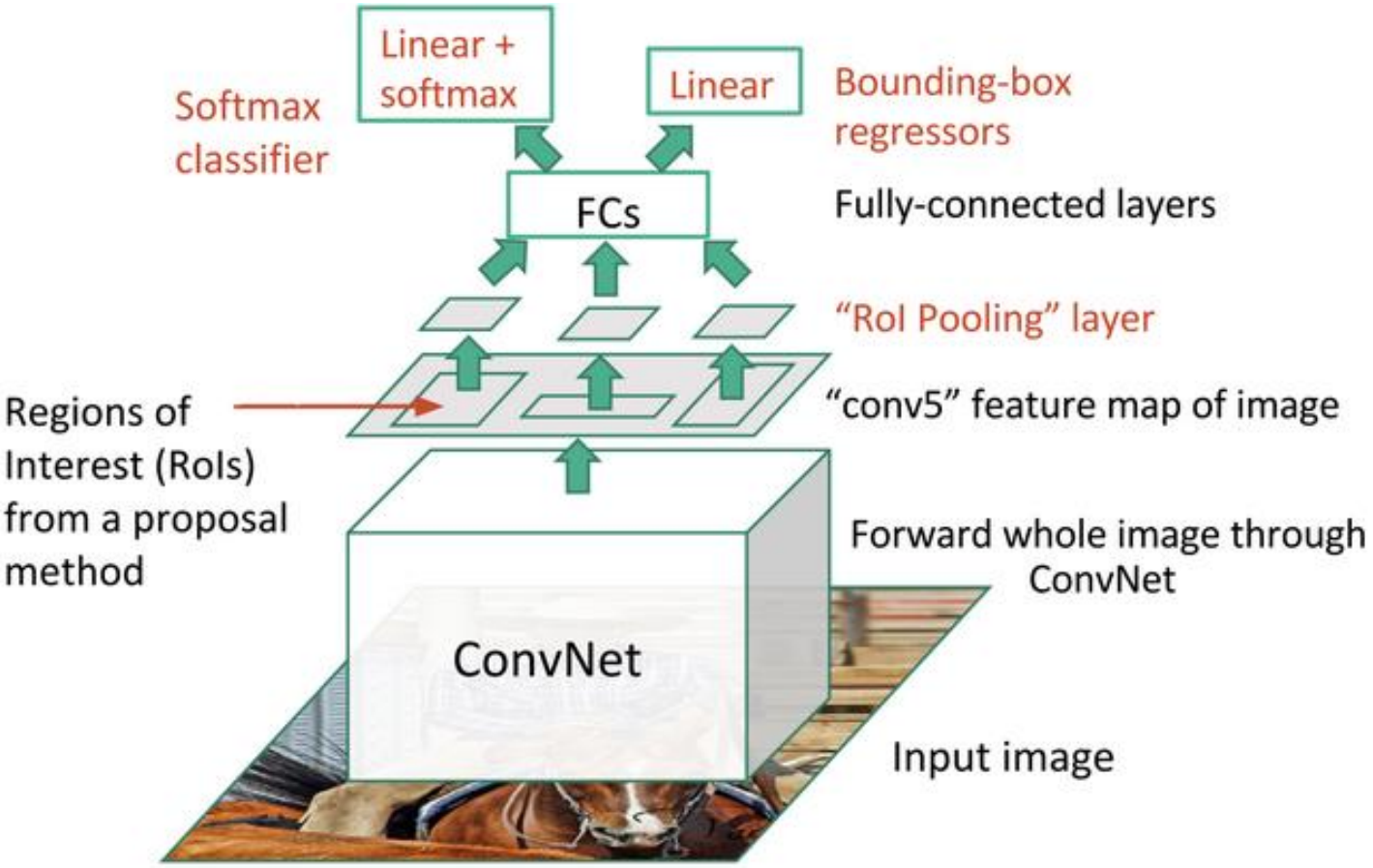
Fast-RCNN



Idea: **No need to recompute features for every box independently**

Fast R-CNN. Girshick. ICCV 2015. https://arxiv.org/abs/1504.08083

Fast-RCNN


Linear + softmax
Linear
Softmax classifier
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

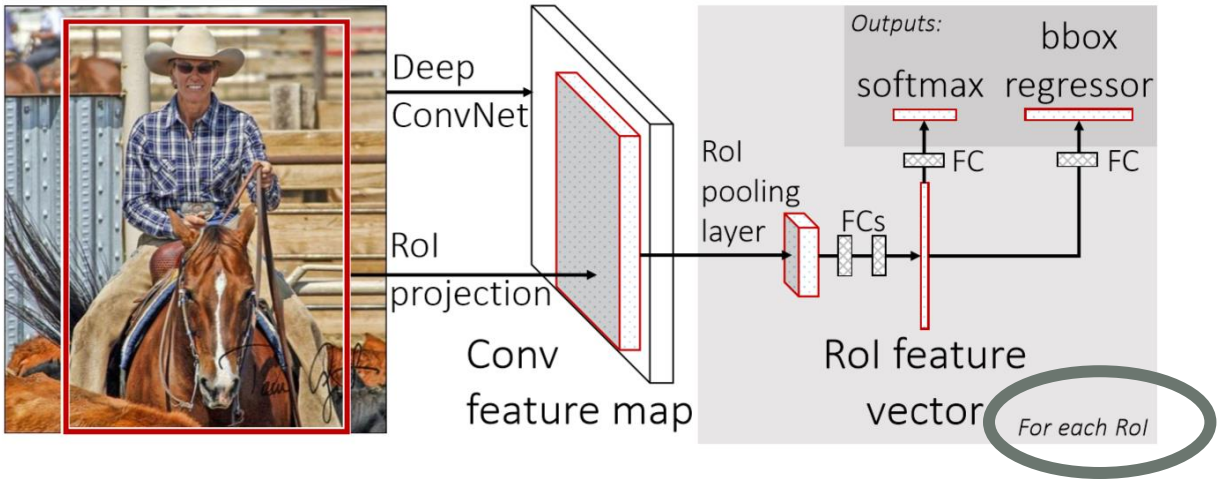
OUTPUTS

Fully connected (fc) layers elaborate feature vectors, that finally branch into two sibling output layers:

- softmax probability estimates over K object classes plus a catch-all “background” class

- another layer that outputs 4*K real-valued numbers: 4 for each of the K object classes.

Each set of 4 values encodes refined bounding-box positions for one of the K classes.

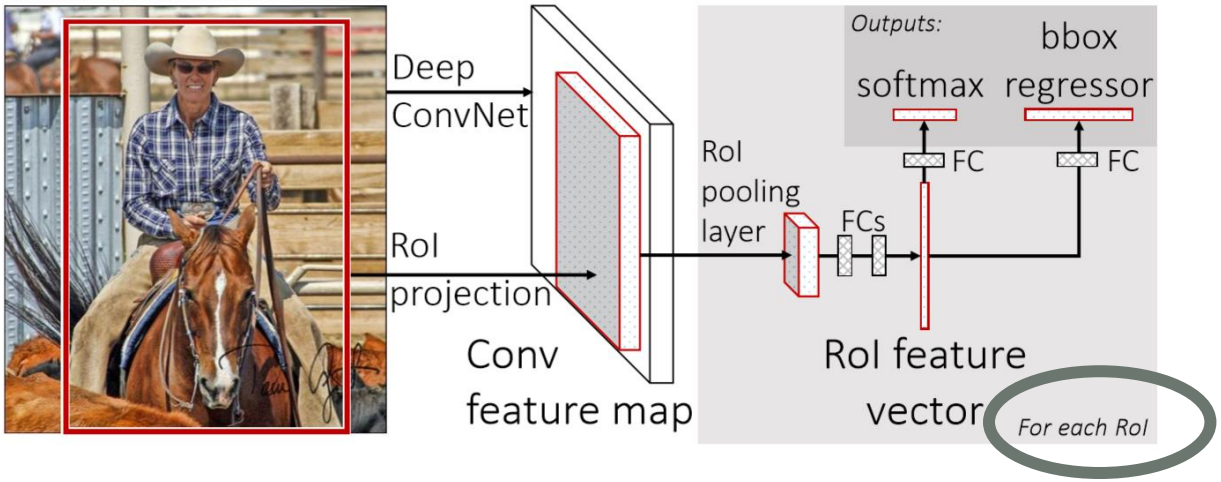


OUTPUTS

Jointly train the CNN, classifier, and bounding box regressor in a single model:

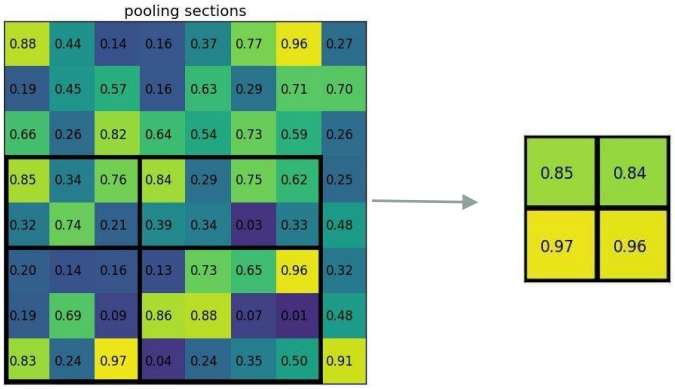
- Fast R-CNN replaced the one-vs-rest SVM classifiers with a softmax layer on top of the CNN to output a classification (introduce competition among classes).

- It also adds a linear regression layer parallel to the softmax layer to output bounding box coordinates.



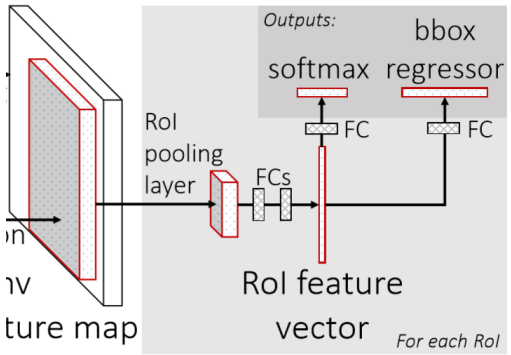
RoI pooling layer

Max pooling to convert the features inside any valid region of interest (region proposal) into a small feature map with a fixed spatial extent of H ×W (layer hyper-parameters that are independent of any particular RoI.)



$$t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$$

$$p = (p_0, \ldots, p_K)$$



It works by dividing a h×w RoI window into an H × W grid of sub-windows of approximate size (h/H) × (w/W) and then max-pooling the values in each sub-window into the corresponding output grid cell

Multi-task loss

Background

As already stated, the outputs are:

- a discrete probability distribution (per RoI) over **K+1 categories**
- bounding-box regression offsets, for each of the K object classes

$$p = (p_0, \ldots, p_K) \quad t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$$

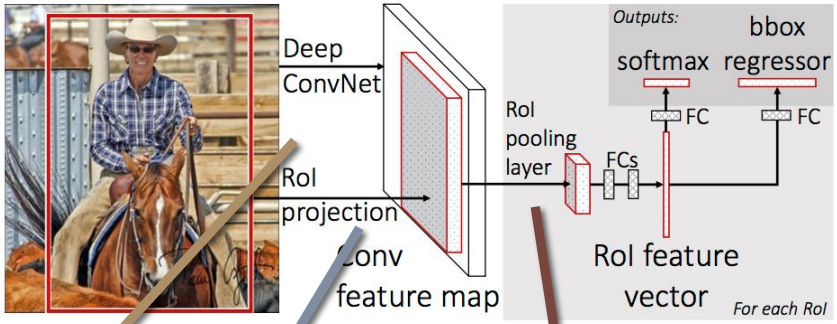
Each training RoI is labeled with a ground-truth class u and a ground-truth bounding-box regression target

$$L(p, u, t^u, v) = L_{\mathrm{cls}}(p, u) + \lambda[u \geq 1] L_{\mathrm{loc}}(t^u, v)$$

Multi-task loss L on each labeled RoI to jointly train for classification and bounding-box regression:
also proved a **task influence**, so multitask help training

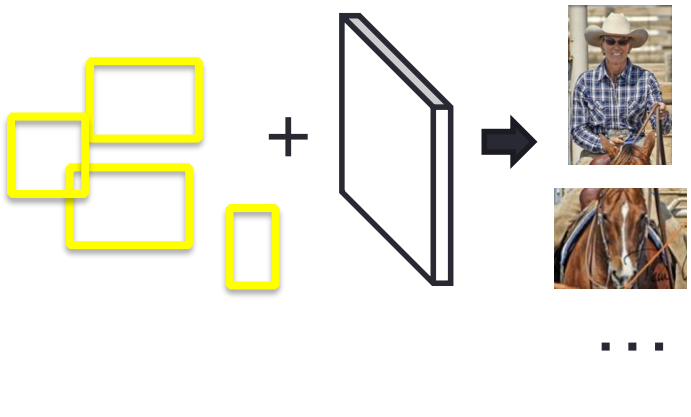
$$L_{\mathrm{cls}}(p, u) = -\log p_u \qquad L_{\mathrm{loc}}(t^u, v) = \sum_{i \in \{\mathrm{x,y,w,h}\}} \mathrm{smooth}_{L_1}(t^u_i - v_i)$$

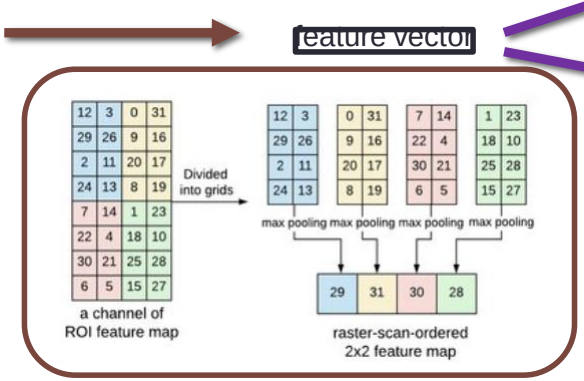
Fast-RCNN



Process the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map.

a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the region feature map.





FC+
softmax

K + 1 categories

FC+
regressor

four real-valued numbers for each of the K object classes.

RCNN vs Fast-RCNN

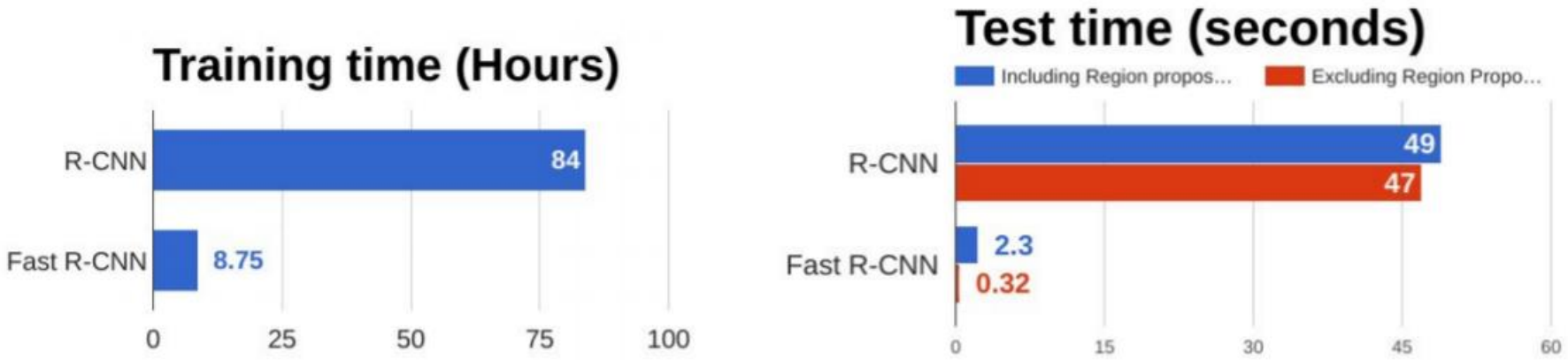


Figure adapted from: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

RCNN vs Fast-RCNN

RCNN

- Simple and scalable.
- improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.

Fast-RCNN

- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.

- proposals are the computational bottleneck in detection systems.

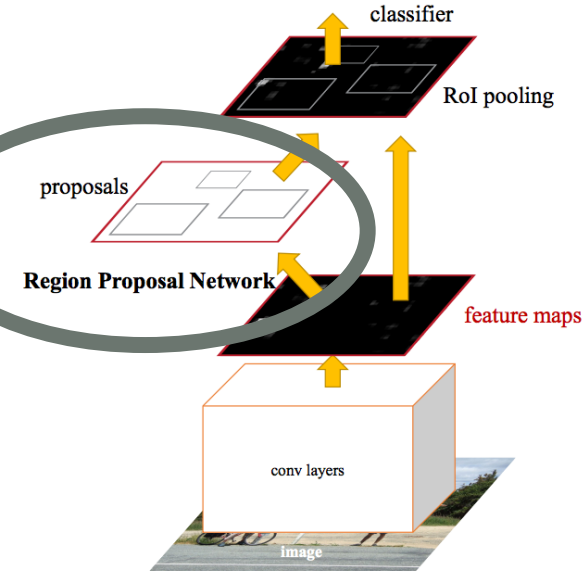
Faster-RCNN

?

2016: Faster R-CNN - Speeding Up Region Proposal

One remaining **bottleneck** in the Fast R-CNN process - the region proposer, **Selective search**. Timing: 2 second per image on CPU implementation

Insight: convolutional feature maps used by region-based detectors, like Fast RCNN, **can also be used for generating region proposals**.



Faster R-CNN

Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what's known as the **Region Proposal Network**.

- Timing : 10ms per image for the proposal
- Execution: **5 fps**

Most influential Google Scholar papers 2020

Google Scholar reveals its most influential papers for 2020

Artificial intelligence papers amass citations more than any other research topic.

13 July 2020

Bec Crew

VCG / Contributor / Getty

Chinese Go player Ke Jie (L) attends a press conference after his second match against Google's artificial intelligence programme AlphaGo on day two of Future of Go Summit in Wuzhen on May 25, 2017 in Jiangxi, Zhejiang Province of China.

Google Scholar has released its annual ranking of most highly cited publications. Artificial intelligence (AI) research **dominates once again**, accumulating huge numbers of citations over the past year.

Related articles

Google Scholar reveals its most influential papers for 2019

2 August 2019

Bec Crew

The 5 most popular scientific papers of April 2020

29 May 2020

Bec Crew

The 5 most popular scientific papers of January 2020

18 February 2020

Bec Crew

Pairing with an influential co-author gives young researchers a career-long boost

7 January 2020

Bec Crew

1. **"Adam: A Method for Stochastic Optimization" (2015)**
International Conference on Learning Representations
47,774 citations

Adam is a popular optimization algorithm for deep learning – a subset of machine learning that uses artificial neural networks inspired by the human brain to imitate how the brain develops certain types of knowledge.

Adam was introduced in this paper at the 2014 International Conference on Learning Representations (ICLR) by Diederik P. Kingma, today a machine learning researcher at Google, and Jimmy Ba from the Machine Learning Group at the University of Toronto, Canada. Adam has since been **widely used in deep learning applications** in computer vision and natural language processing

The ICLR, one of the most prestigious conferences on machine learning, is an important platform for researchers whose papers are accepted. In May 2020, the conference drew 5,600 participants from nearly 90 countries to its virtual sessions – **more than double the turnout in 2019**, at 2,700 physical attendees.

2. **"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" (2015)**
Neural Information Processing Systems
19,507 citations

Presented at the 2015 Neural Information Processing Systems annual meeting in Canada, this paper describes what has now become the most widely-used version of an object detection algorithm called **R-CNN**.

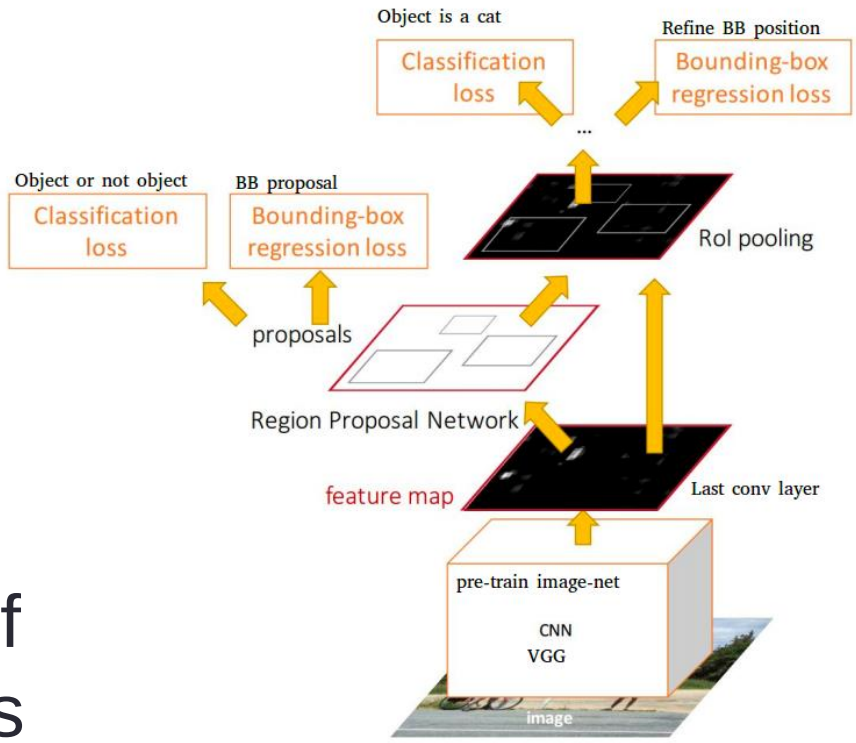
Object detection is a major part of computer vision research, used to identify objects such as humans, cars, and buildings in images and videos.

The lead author, Shaoqing Ren, is also a co-author on Google's most-cited paper for 2020, "Deep Residual Learning for Image Recognition", which has amassed almost 50,000 citations. **Read more about it here**.

That paper was co-authored by Ross Girshick, one of the invertors of R-CNN and now a research scientist at Facebook AI.

In the same week that "Faster R-CNN" was presented by Ren and his colleagues, Girshick presented a paper on "Fast R-CNN", another version of R-CNN, at a different conference. That paper, presented at the 2015 IEEE International Conference on Computer Vision in Chile, has amassed **more than 10,000 citations**.

Faster-RCNN

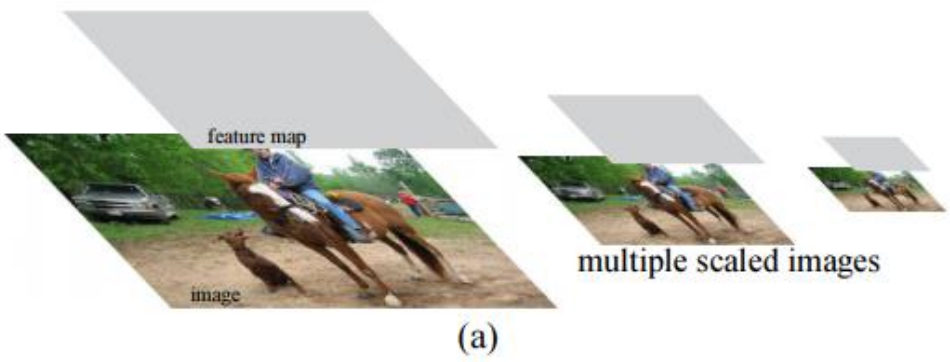


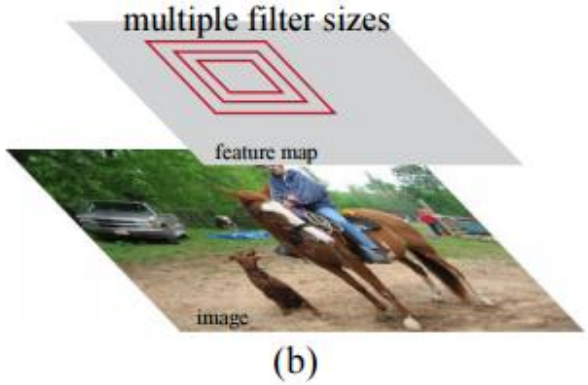
Idea: Integrate the Bounding Box Proposals as part of the CNN predictions

Ren et al. NIPS 2015. https://arxiv.org/abs/1506.01497

2016: Faster R-CNN - Speeding Up Region Proposal

Different schemes for addressing multiple scales and sizes





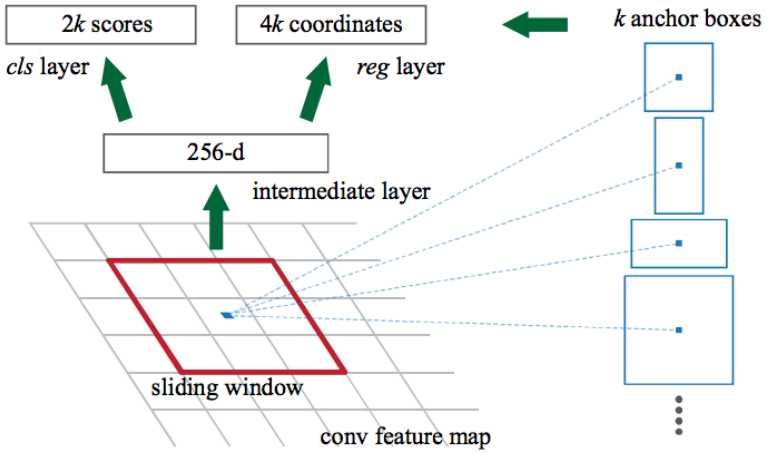
(a) Pyramids of images and feature maps are built, and the classifier is run at all scales

(b) Pyramids of filters with multiple scales/sizes are run on the feature map.

Region Proposal Network

‘Attention’ mechanisms, the RPN module tells the Fast R-CNN module where to look.

Slide a small network over the convolutional feature map output by the last shared convolutional layer, with input an *n x n* spatial window, mapped to a lower-dim feature



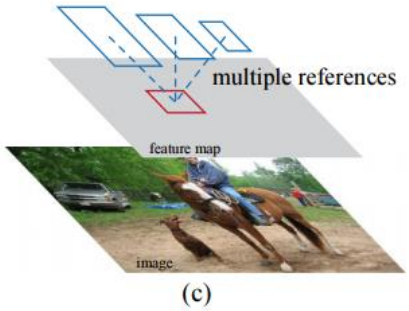
This feature is fed into two sibling fully connected layers - a box-regression layer (**reg**) and a box-classification layer (**cls**)

Region Proposal Network

At each sliding-window location (center of the window called **anchor**), we simultaneously predict multiple k region proposals, called ***anchor boxes***, with different sizes and aspect ratios (k=9, 3 sizes and 3 ratios)

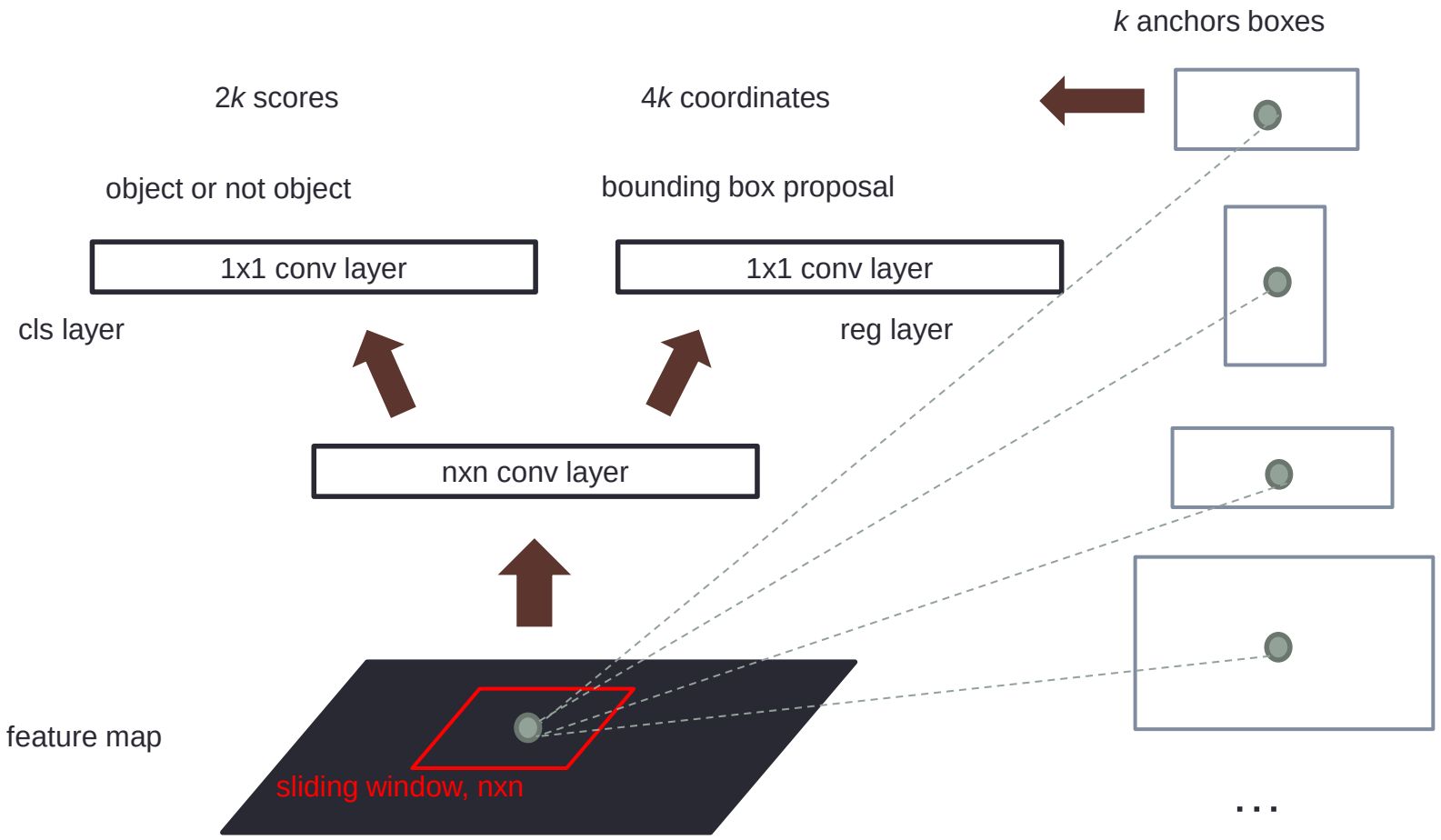
For a convolutional feature map of a size W × H (typically ~2,400), there are W*H*k anchors in total

(c) a pyramid of anchors



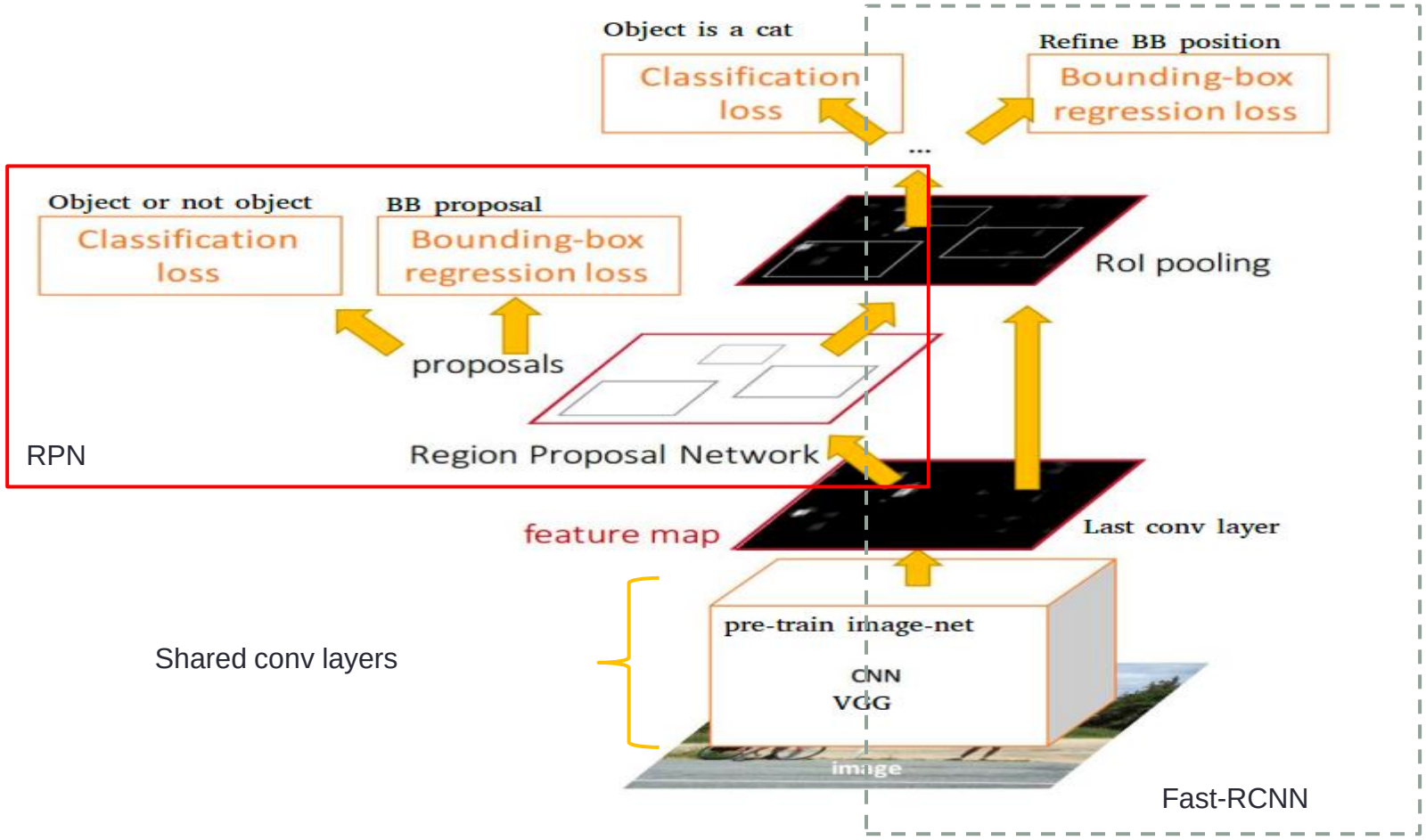
The **reg** layer has 4k outputs encoding the coordinates of k boxes (**relative to the anchor**), and the **cls** layer outputs 2k scores that estimate probability of **object or not object** for each proposal

Faster-RCNN



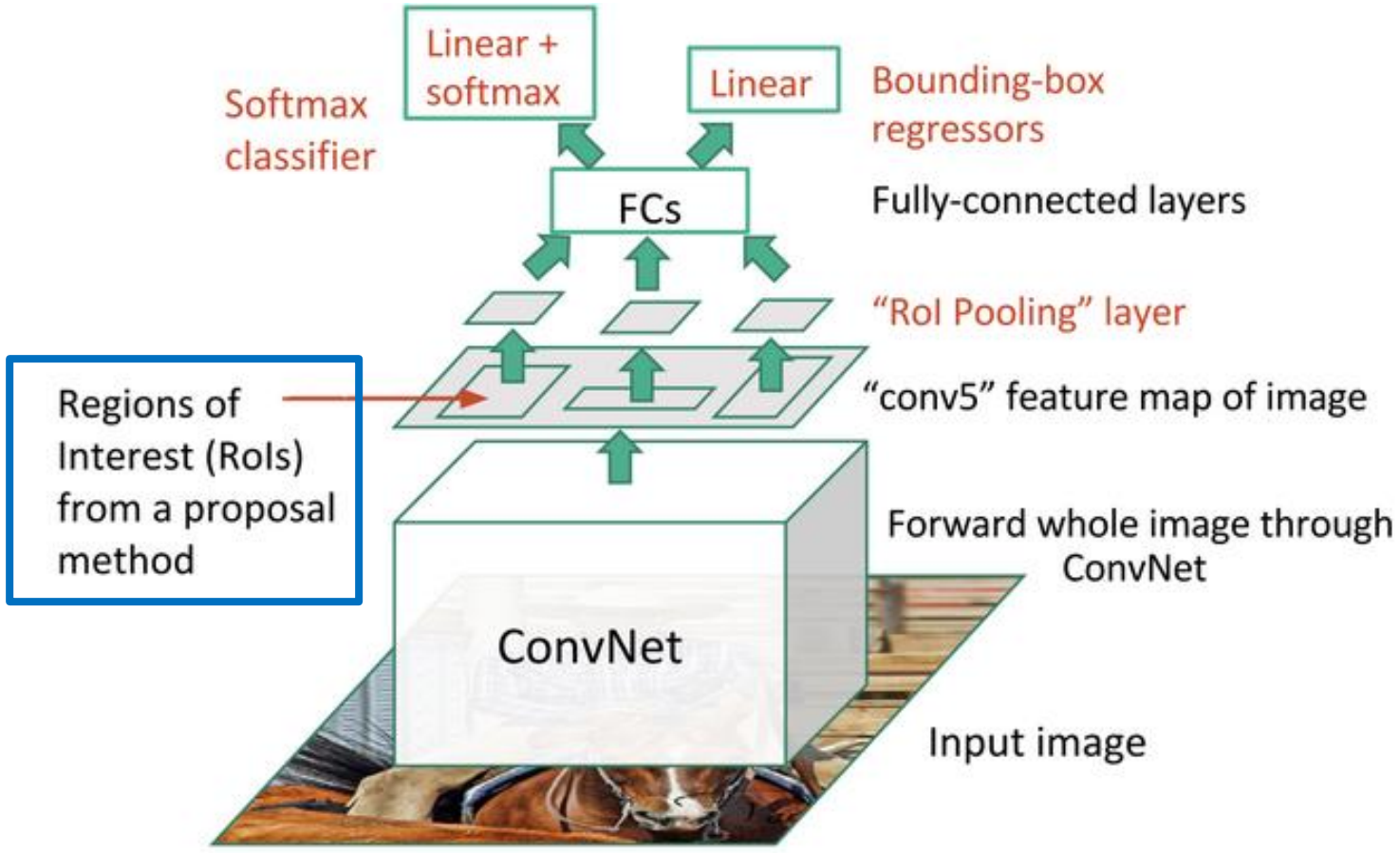
Region Proposal Networks

Faster-RCNN



Region Proposal Networks

Faster-RCNN



Region Proposal Networks

RPN Loss function

Assigned a binary class label (of being an object or not) to each anchor (highest IoU with a ground-truth box or over a threshold).

Note that a single ground-truth box may assign positive labels to multiple anchors.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) \quad +\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

The regression loss is activated only for positive anchors ($p^*_i = 1$) and is disabled otherwise ($p^*_i = 0$)

Global training

For the detection network, adopted Fast R-CNN

4-Step Alternating Training:

1. CNN network is initialized with an ImageNet-pre-trained model, RPN fine-tuned end-to-end for the region proposal task.
2. train a separate detection network (same on Imagenet) by Fast R-CNN using the proposals generated by the step-1 RPN
3. use the detector network to initialize RPN training fixing x the shared convolutional layers and only fine-tune the layers unique to RPN
4. keeping the shared convolutional layers fixed, we fine-tune the unique layers of Fast R-CNN

RCNN vs Fast-RCNN

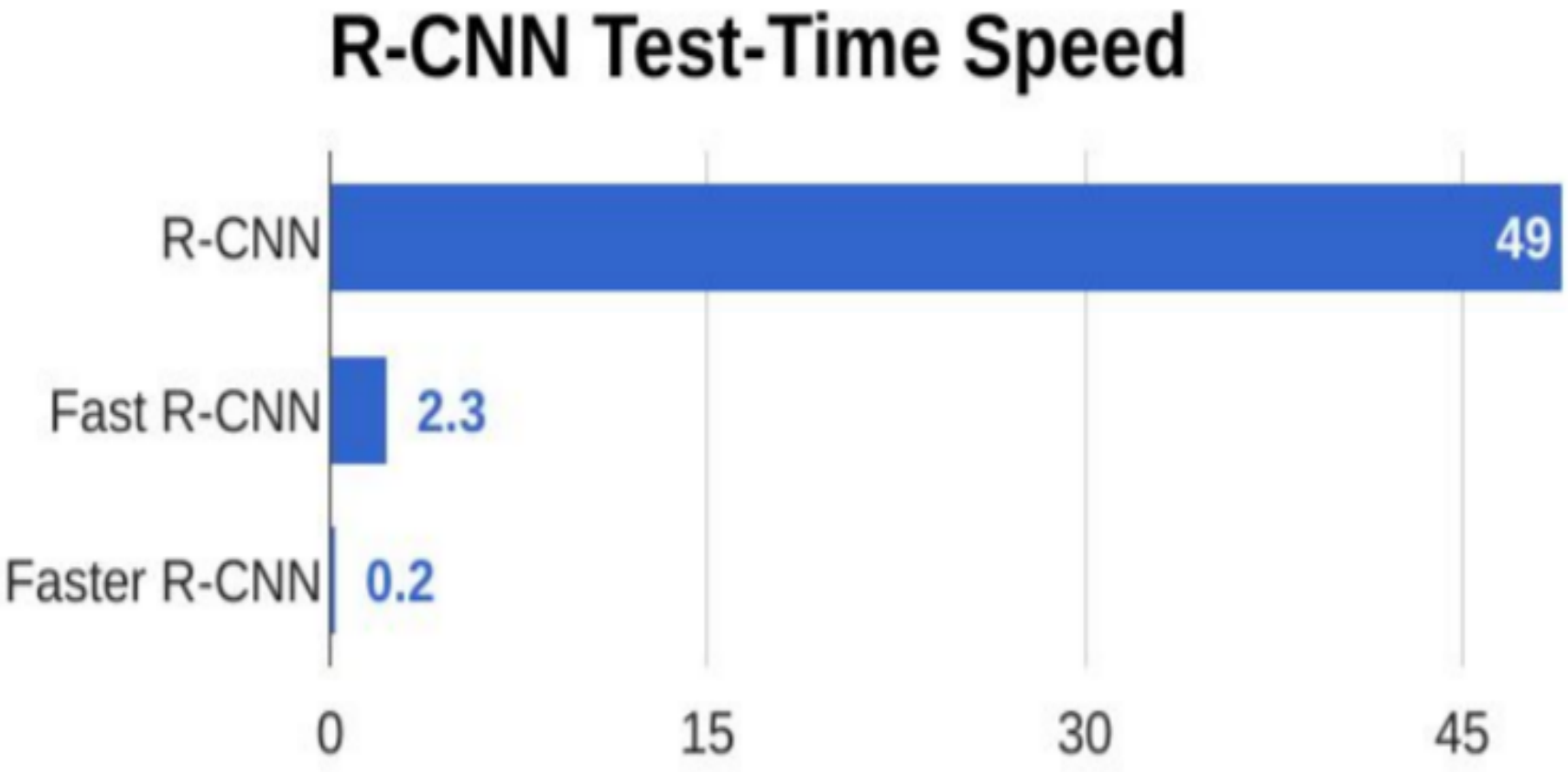


Figure adapted from: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

RCNN vs Fast-RCNN vs Faster-RCNN

RCNN

- Simple and scalable.
- improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.

Fast-RCNN

- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.

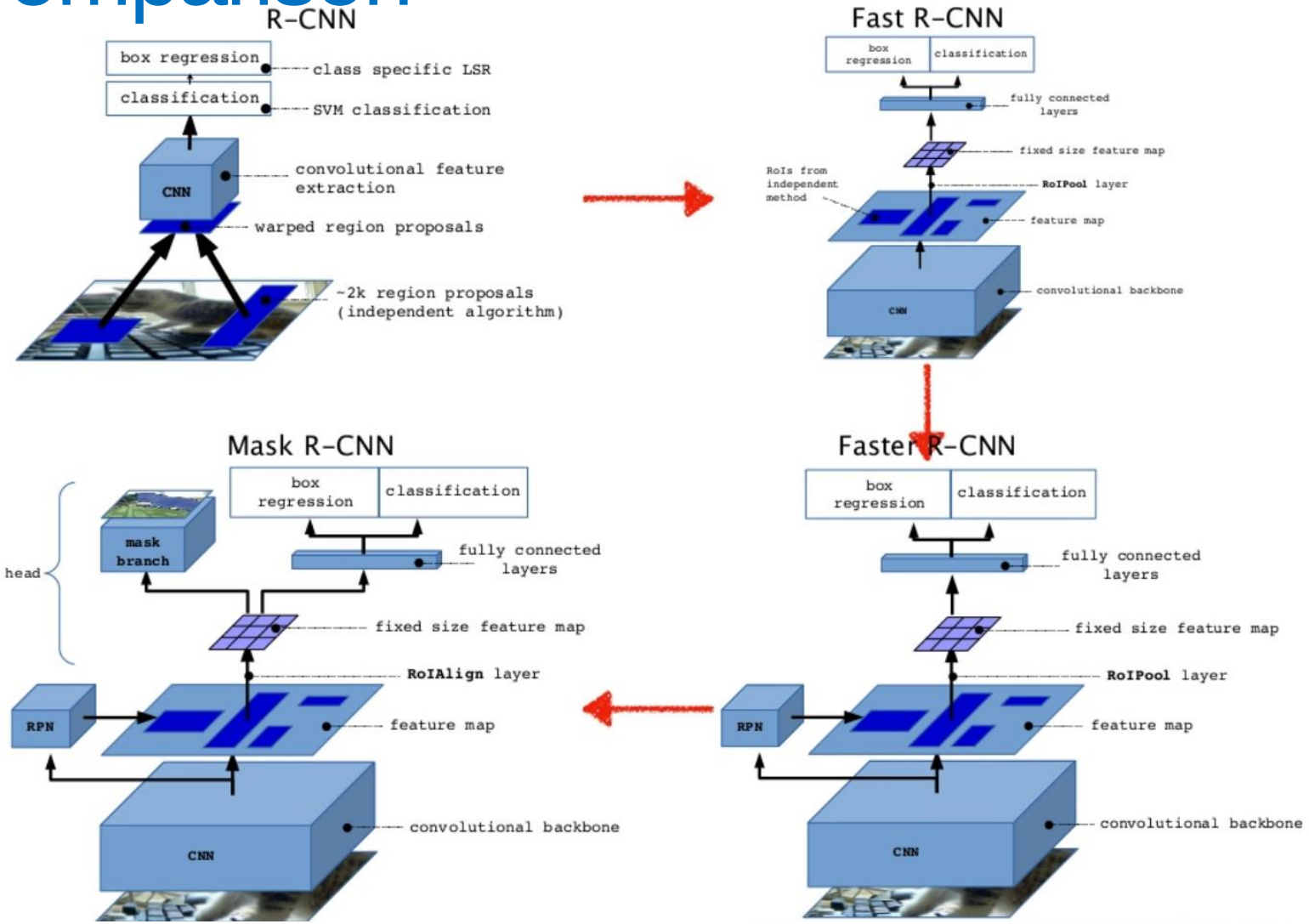
- Proposals are the computational bottleneck in detection systems.

Faster-RCNN

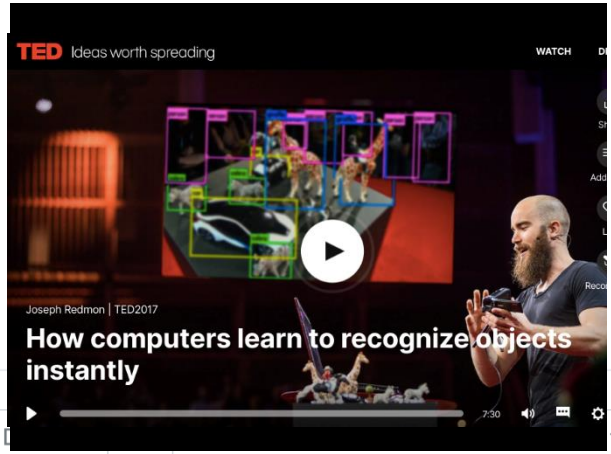
- Compute proposals with a deep convolutional neural network --*Region Proposal Network* (RPN)
- Merge RPN and Fast R-CNN into a single network, enabling nearly cost-free region proposals.

?

Comparison


R-CNN
box regression
class specific LSR
classification
SVM classification
CNN
convolutional feature extraction
warped region proposals
~2k region proposals (independent algorithm)
Fast R-CNN
box regression
classification
fully connected layers
fixed size feature map
RoIs from independent method
RoIPool layer
feature map
CNN
convolutional backbone
Faster R-CNN
box regression
classification
fully connected layers
fixed size feature map
RoIPool layer
RPN
feature map
CNN
convolutional backbone
Mask R-CNN
box regression
classification
mask branch
head
fully connected layers
fixed size feature map
RoIAlign layer
RPN
feature map
CNN
convolutional backbone

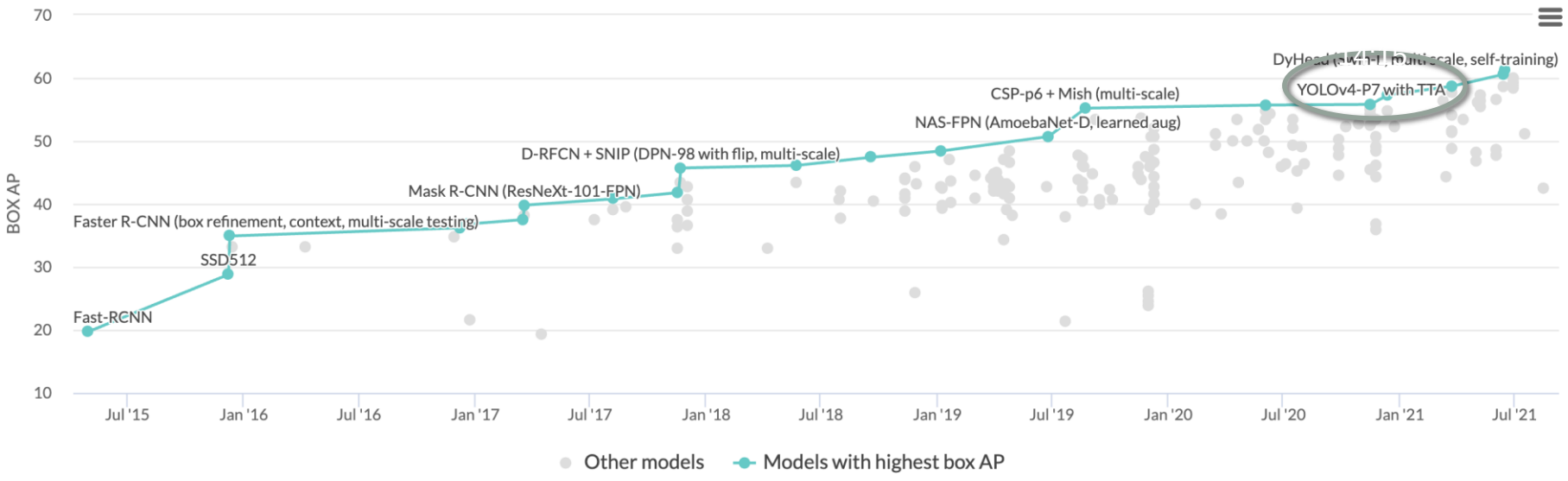
Object detection with YOLO



Object Detection on COCO test-dev

Leaderboard Dataset

View box AP by



YOLO- You Only Look Once

Idea: No bounding box proposal.

A single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

- extremely **fast**
- reason **globally**
- learn **generalizable** representations.

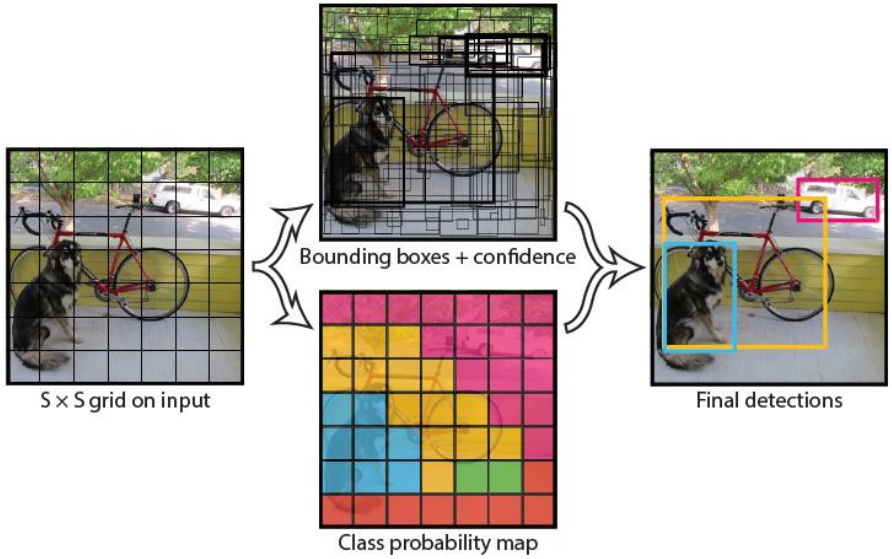
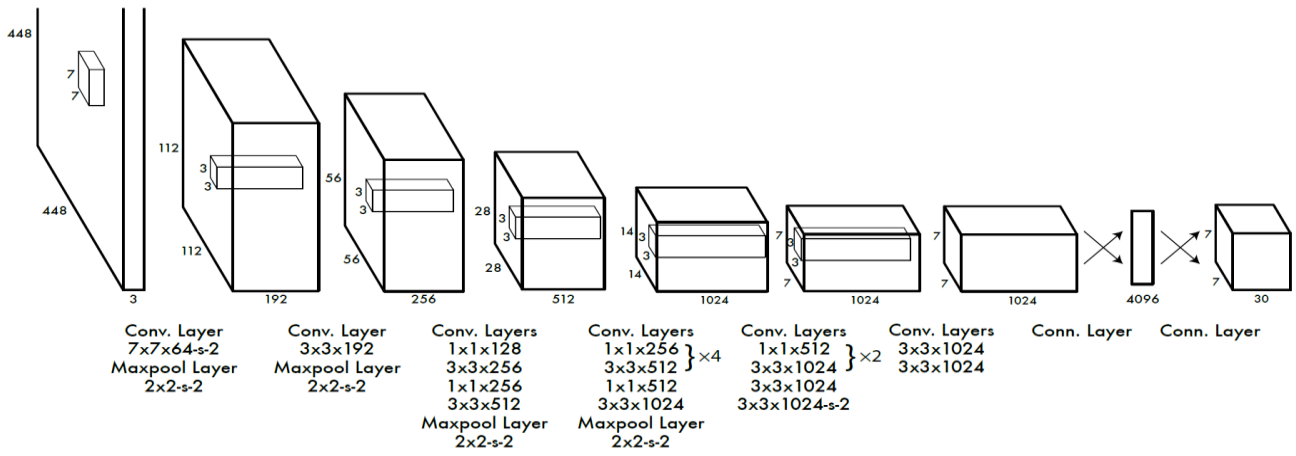


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

https://arxiv.org/abs/1506.02640
Redmon et al. CVPR 2016.

YOLO- You Only Look Once



- Divide the image into 7x7 cells.
- Each cell trains a detector.
- The detector needs to predict the object's class distributions.
- The detector has B bounding-box predictors to predict bounding-boxes and confidence scores.

2016: Yolo – you only look once

Prior work on object detection repurposes classifiers to perform detection.

- Object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.
  - A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation.

- Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance ***~ 45 fps, real time***

YOLO sees the entire image during training and test time so it implicitly **encodes contextual information** about classes as well as their appearance.

2016: Yolo – you only look once

Input image

↓

Split into grids

↓

Create bounding boxes and predict confidence P(object) for each box

↓

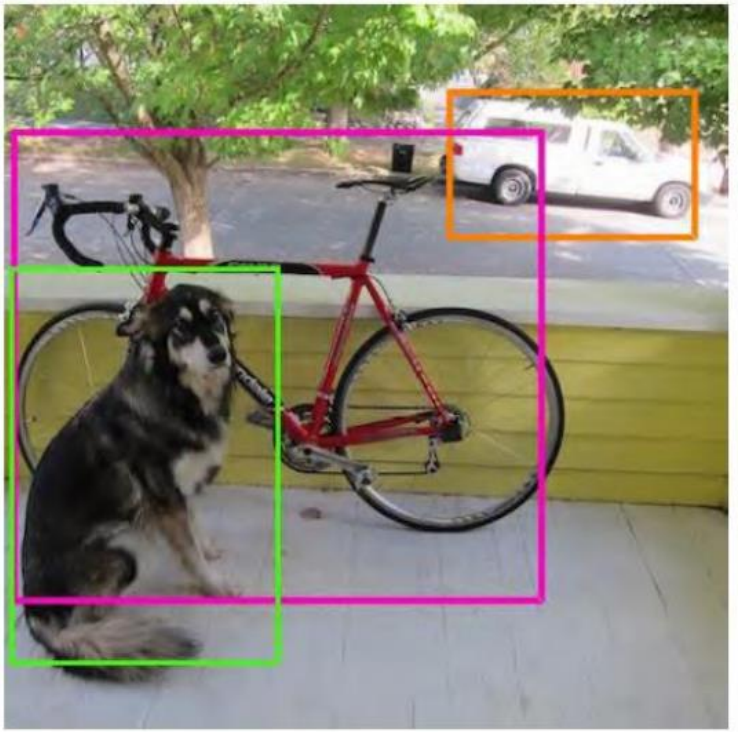
Predict class probability: P(Class | Object) for each cell

↓

Multiply the confidence value and the class probability

↓

Choose best prediction using non-maximal suppression



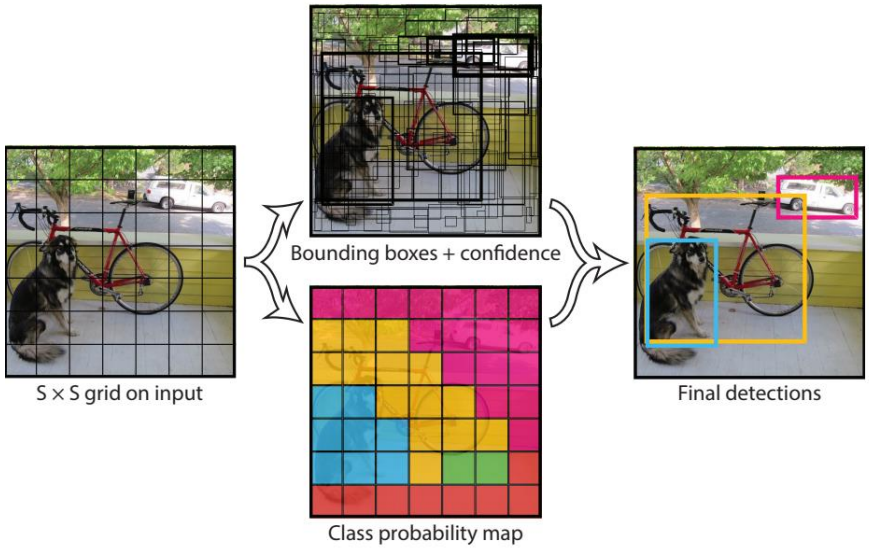
Unified detection

- Divides the input image into an S × S grid. If the **center** of an object falls into a grid cell, that grid cell is **responsible** for detecting that object.

- Each grid cell predicts **B** bounding boxes (4 predictions, x,y,w,h) and confidence scores for those boxes

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- Each grid cell also predicts **C** conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$, regardless of the number of boxes B.

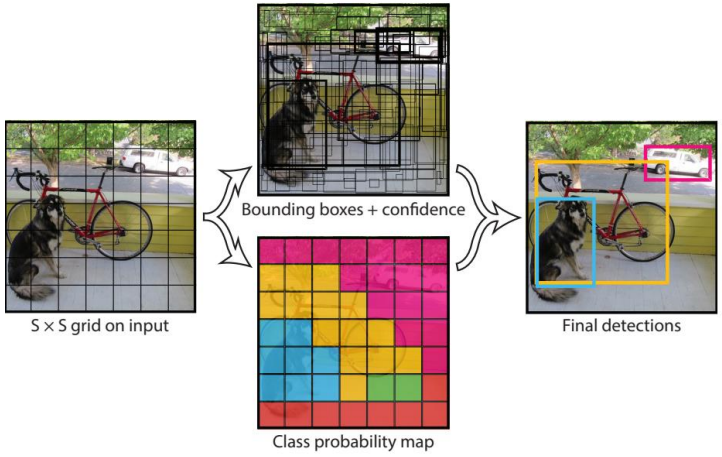


Unified detection

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

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

These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.



These predictions are encoded as an S × S × (B ∗ 5 + C) tensor.

Loss function

- One predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth.
- This leads to specialization between the bounding box predictors.
- Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

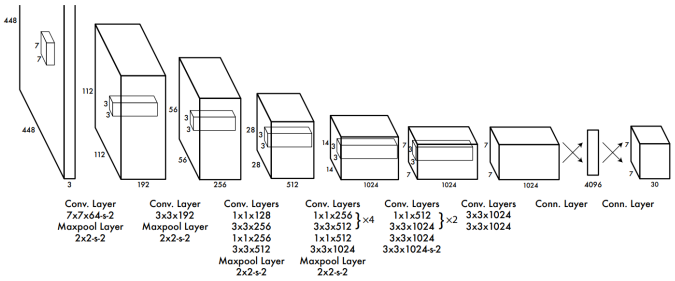
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

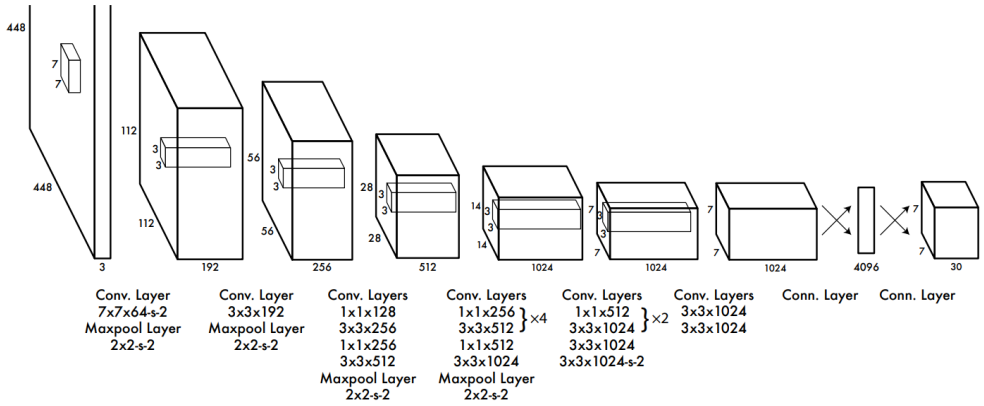
where 1^{obj}_{i} denotes if object appears in cell i and 1^{obj}_{ij} denotes that the j-th bounding box predictor in cell i is “responsible” for that prediction.

Net design and training/problems



- Pretrain first 20 convolutional layers on the ImageNet 1000-class competition dataset.
- Add four convolutional layers and two fully connected layers with randomly initialized weights.
- Optimize sum-squared error, but it weights localization error equally with classification error which may not be ideal => introduced parameters to weight.

Net design and training/problems



- Many grid cells do not contain any object pushing the “confidence” scores of those cells towards zero, often overpowering the gradient from cells that do contain object

- Equally weights errors in large boxes and small boxes.

  - Metric should reflect that small deviations in large boxes matter less than in small boxes.

2016: Yolo – you only look once

YOLO **learns generalizable representations** of objects.

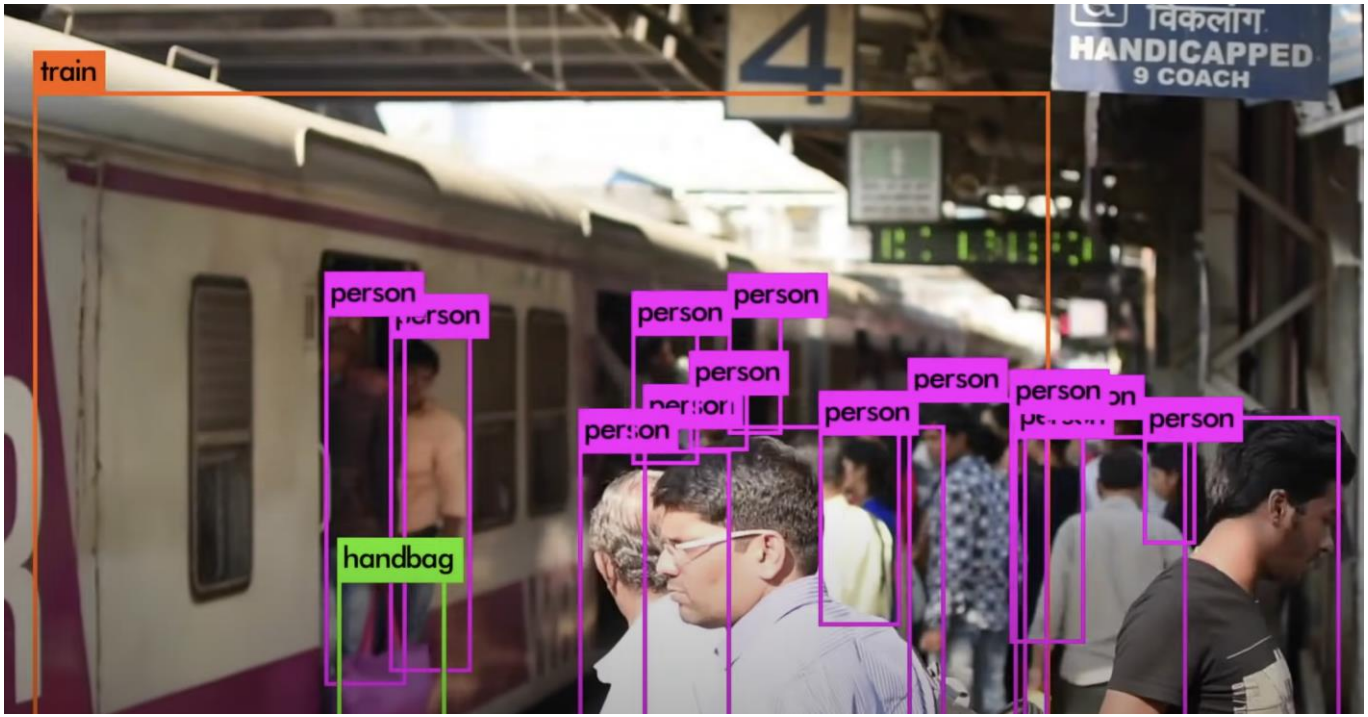
When trained on natural images and tested on artwork, YOLO outperforms top detection methods

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
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

LIMITATIONS OF YOLO

- Each grid cell **only predicts B boxes** and can only have **one class** (limits the number of nearby objects that model can predict)
- it struggles to generalize to objects in **new or unusual** aspect ratios or configurations
- loss function treats **errors the same in small bounding boxes versus large** bounding boxes (partially addressed using square roots).

Real-Time Detection In The Wild



- Possible to connect YOLO to a **webcam** and verify that it maintains real-time performance.

- While YOLO processes images individually, when attached to a webcam, it **acts like a tracking** system, detecting objects as they move around and change in appearance.
- https://pjreddie.com/darknet/yolo/

2017: yolov2 -YOLO9000: Better, Faster, Stronger

- The original YOLO trains the classifier network at 224 × 224 on Imagenet and increases the resolution to 448 for detection.

- YOLOv2 first **finetunes the classification network** at the full 448 × 448 resolution for 10 epochs on ImageNet.
- This gives the network time to adjust its filters to work better on higher resolution input.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Many more versions…

- Yolo
- Yolov2
- Yolov3
- Yolov4
- Yolov5
- Yolov6
- Yolov7
- YoloX
- YoloS
- YoloF
- YoloR
- …

Most influential Googgle scholar papers 2021

Google Scholar reveals its most influential papers for 2021

Early clinical observations of COVID-19 and its mortality risk factors among the most cited output, while a five-year-old AI paper continues to command attention.

24 August 2021

Bec Crew

Wei Liu et al.Ã Â Ã Â European Conference on Computer Vision (2016)

Examples of using SSD, an object-detection algorithm described in a highly cited artificial intelligence paper.

COVID-19-related papers have eclipsed artificial intelligence research in the

Related articles

Google Scholar reveals its most influential papers for 2020

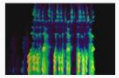
13 July 2020

Bec Crew

Google Scholar reveals its most influential papers for 2019

2 August 2019

Bec Crew



The race to the top among the world's leaders in artificial intelligence

10 December 2020

Neil Savage

COVID-19 research update: How many pandemic papers have been published?

28 August 2020

Nature Index

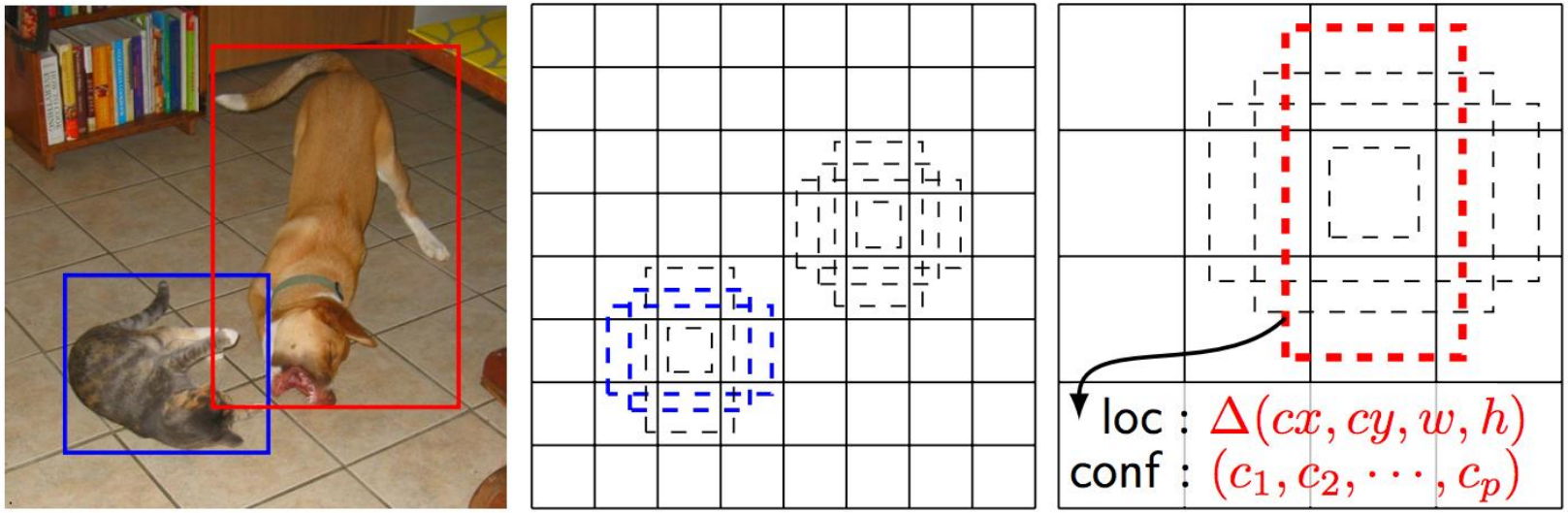
Latest supplement

Nature Index 2021 Canada

Key to Canada's strength as a leading science nation is its high-quality research, informed by its vast natural resources. But the country is losing ground in areas where it held an early lead.

Access free ▸

SSD: Single Shot Detector



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

Idea: Similar to YOLO, but denser grid map, multiscale grid maps.

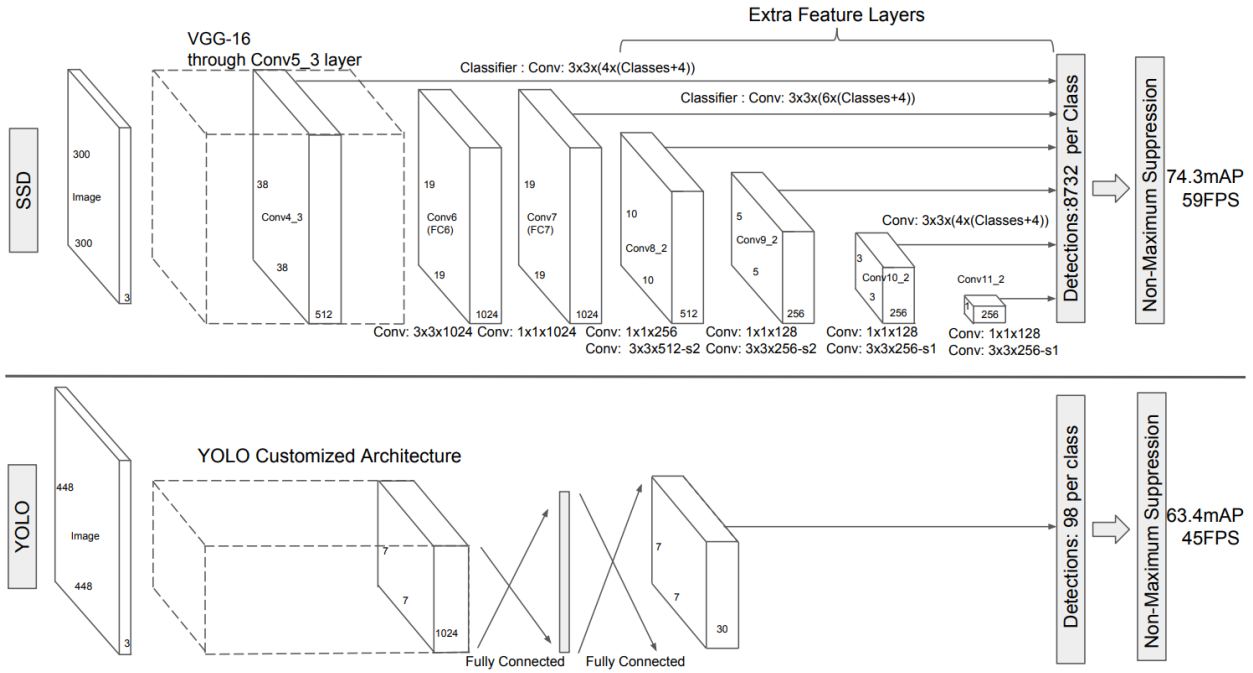
+ Data augmentation
+ Hard negative mining

Liu et al. ECCV 2016

2016: Single Shot Multibox detector

The early network layers are based on a standard architecture (VGG) used for high quality image classification

- **add convolutional feature layers to the end of the truncated base network**, which decreases in size progressively and
- allow predictions of **detections at multiple scales**.



Liu, Wei, et al. "Ssd: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, 2016.

Multi-scale feature maps for detection

In a convolutional fashion, evaluate a small set (e.g. 4) of **default boxes** of different aspect ratios at each location in **several feature maps with different scales**

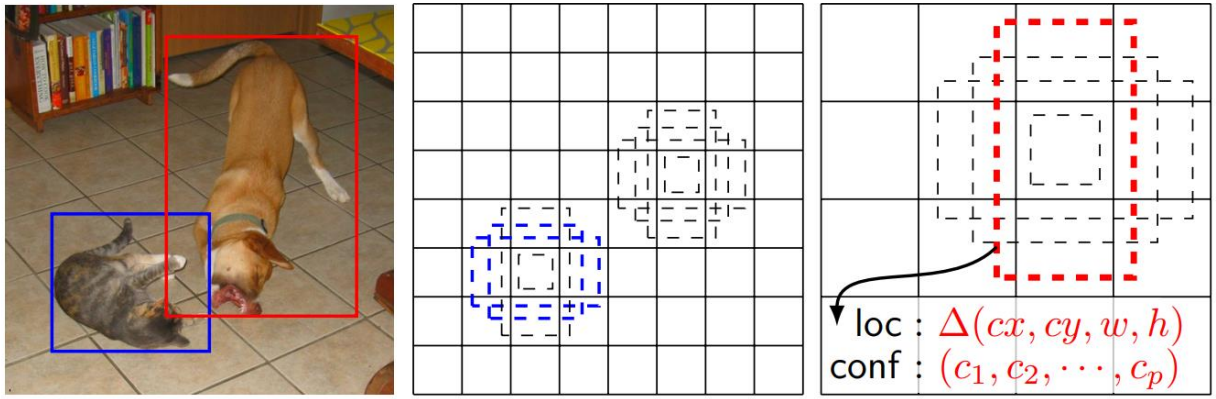
For each default box, predict both the shape offsets and the confidences for all object categories

This results in a total of **(c + 4)k filters** (k default boxes) that are applied around each location in the feature map, yielding **(c + 4)kmn** outputs for a m × n feature map

Multi-scale feature maps for detection

Default boxes are similar to the anchor boxes used in Faster R-CNN, but **applied to several feature maps of different resolutions**.

Allowing different default box shapes in several feature maps let us efficiently discretize the space of possible output box shapes.



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

Matching strategy

Key difference between training SSD and training a typical detector that uses region proposals, is that **ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs**:

- need to determine which default boxes correspond to a ground truth detection and train the network accordingly.

Some version of this is also required for training in YOLO and for the region proposal stage of Faster R-CNN.

Matching strategy

- **Matched each ground truth box to the default box with the best Jaccard overlap,**
  - then match default boxes to any ground truth with Jaccard overlap higher than a threshold (0.5).

This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping default boxes rather than requiring it to pick only the one with maximum overlap.

The **overall objective loss function** is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Hard Negative Mining in Object Detection

- **Imbalance Issue**: Excessive negative predictions compared to actual objects.
- **SSD's Approach**: Sorts negatives by confidence loss and selects those with the highest loss.
- **Balanced Ratio**: Maintains a maximum 3:1 ratio of negatives to positives.
- **Outcome**: Enhances training speed and stability.

Cons

SSD is very **sensitive to the bounding box size**.

- In other words, it has much worse performance on smaller objects than bigger objects.

- This is not surprising because those small objects may not even have any information at the very top layers

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Table 7: **Results on Pascal VOC2007 `test`.** SSD300 is the only real-time detection method that can achieve above 70% mAP. By using a larger input image, SSD512 outperforms all methods on accuracy while maintaining a close to real-time speed.

Comparison on VOC'2007

PASCAL VOC 2007 and 2012 data sets consist of 20 categories. The evaluation terms are AP in each single category and mAP across all the 20 categories.

COMPARATIVE RESULTS ON VOC 2007 TEST SET (%)

Methods	Trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN (Alex) [15]	07	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	68.6	58.5
R-CNN(VGG16) [15]	07	73.4	77.0	63.4	45.4	44.6	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	35.6	66.8	67.2	70.4	71.1	66.0
SPP-net(ZF) [64]	07	68.5	71.7	58.7	41.9	42.5	67.7	72.1	73.8	34.7	67.0	63.4	66.0	72.5	71.3	58.9	32.8	60.9	56.1	67.9	68.8	60.9
GCNN [70]	07	68.3	77.3	68.5	52.4	38.6	78.5	79.5	81.0	47.1	73.6	64.5	77.2	80.5	75.8	66.6	34.3	65.2	64.4	75.6	66.4	66.8
Bayes [85]	07	74.1	83.2	67.0	50.8	51.6	76.2	81.4	77.2	48.1	78.9	65.6	77.3	78.4	75.1	70.1	41.4	69.6	60.8	70.2	73.7	68.5
Fast R-CNN [16]	07+12	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4	70.0
SDP+CRC [34]	07	76.1	79.4	68.2	52.6	46.0	78.4	78.4	81.0	46.7	73.5	65.3	78.6	81.0	76.7	77.3	39.0	65.1	67.2	77.5	70.3	68.9
SubCNN [60]	07	70.2	80.5	69.5	60.3	47.9	79.0	78.7	84.2	48.5	73.9	63.0	82.7	80.6	76.0	70.2	38.2	62.4	67.7	77.7	60.5	68.5
StuffNet30 [100]	07	72.6	81.7	70.6	60.5	53.0	81.5	83.7	83.9	52.2	78.9	70.7	85.0	85.7	77.0	78.7	42.2	73.6	69.2	79.2	73.8	72.7
NOC [114]	07+12	76.3	81.4	74.4	61.7	60.8	84.7	78.2	82.9	53.0	79.2	69.2	83.2	83.2	78.5	68.0	45.0	71.6	76.7	82.2	75.7	73.3
MR-CNN&S-CNN [105]	07+12	80.3	84.1	78.5	70.8	**68.5**	88.0	85.9	87.8	60.3	85.2	73.7	87.2	86.5	85.0	76.4	48.5	76.3	75.5	85.0	81.0	78.2
HyperNet [101]	07+12	77.4	83.3	75.0	69.1	62.4	83.1	87.4	87.4	57.1	79.8	71.4	85.1	85.1	80.0	79.1	51.2	79.1	75.7	80.9	76.5	76.3
MS-GR [104]	07+12	80.0	81.0	77.4	72.1	64.3	88.2	88.1	88.4	64.4	85.4	73.1	87.3	87.4	85.1	79.6	50.1	78.4	79.5	86.9	75.5	78.6
OHEM+Fast R-CNN [113]	07+12	80.6	85.7	79.8	69.9	60.8	88.3	87.9	89.6	59.7	85.1	76.5	87.1	87.3	82.4	78.8	53.7	80.5	78.7	84.5	80.7	78.9
ION [95]	07+12+S	80.2	85.2	78.8	70.9	62.6	86.6	86.9	**89.8**	61.7	86.9	76.5	**88.4**	87.5	83.4	80.5	52.4	78.1	77.2	86.9	**83.5**	79.2
Faster R-CNN [17]	07	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6	69.9
Faster R-CNN [17]	07+12	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6	73.2
Faster R-CNN [17]	07+12+COCO	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9	78.8
[illegible]	07+12+COCO	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	87.5	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9	79.6
SSD512 [7]	07+12+COCO	**86.6**	**88.3**	**82.4**	76.0	66.3	**88.6**	**88.9**	89.1	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	81.2	**81.6**

* '07': VOC2007 trainval, '07+12': union of VOC2007 and VOC2012 trainval, '07+12+COCO': trained on COCO trainval35k at first and then fine-tuned on 07+12. The S in ION '07+12+S' denotes SBD segmentation labels.

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), 3212-3232.

Comparison on VOC'2012

Methods	Trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN(Alex) [15]	12	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1	53.3
R-CNN(VGG16) [15]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
Bayes [85]	12	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2	66.4
Fast R-CNN [65]	07++12	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	68.4
SutffNet30 [100]	12	83.0	76.9	71.2	51.6	50.1	76.4	75.7	87.8	48.3	74.8	55.7	85.7	81.2	80.3	79.5	44.2	71.8	61.0	78.5	65.4	70.0
NOC [114]	07+12	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1	68.8
MR-CNN&S-CNN [105]	07++12	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0	73.9
HyperNet [101]	07++12	84.2	78.5	73.6	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7	71.4
OHEM+Fast R-CNN [113]	07++12+coco	90.1	87.4	79.9	65.8	66.3	86.1	85.0	92.9	62.4	83.4	69.5	90.6	88.9	88.9	83.6	59.0	82.0	74.7	88.2	77.3	80.1
ION [95]	07+12+S	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	57.8	82.0	64.7	88.9	86.5	84.7	82.3	51.4	78.2	69.2	85.2	73.5	76.4
Faster R-CNN [17]	07++12	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5	70.4
Faster R-CNN [17]	07++12+coco	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2	75.9
YOLO [18]	07++12	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8	57.9
YOLO+Fast R-CNN [18]	07++12	83.4	78.5	73.5	55.8	43.4	79.1	73.1	89.4	49.4	75.5	57.0	87.5	80.9	81.0	74.7	41.8	71.5	68.5	82.1	67.2	70.7
YOLOv2 [72]	07++12+coco	88.8	87.0	77.8	64.9	51.8	85.2	79.3	93.1	64.4	81.4	**70.2**	91.3	88.1	87.2	81.0	57.7	78.1	71.0	88.5	76.8	78.2
SSD300 [71]	07++12+coco	91.0	86.0	78.1	65.0	55.4	84.9	84.0	93.4	62.1	83.6	67.3	91.3	88.9	88.6	85.6	54.7	83.8	77.3	88.3	76.5	79.3
SSD512 [71]	07++12+coco	91.4	88.6	82.6	71.4	63.1	**87.4**	88.1	93.9	66.9	86.6	66.3	92.0	91.7	90.8	88.5	60.9	87.0	75.4	90.2	80.4	82.2
R-FCN (ResNet101) [65]	07++12+coco	**92.3**	**89.9**	**86.7**	**74.7**	**75.2**	86.7	**89.0**	**95.8**	**70.2**	**90.4**	66.5	**95.0**	**93.2**	**92.1**	**91.1**	**71.0**	**89.7**	**76.0**	**92.0**	**83.4**	**85.0**

* '07++12': union of VOC2007 trainval and test and VOC2012 trainval. '07++12+COCO': trained on COCO trainval35k at first then fine-tuned on 07++12.

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), 3212-3232.

Results on VOC'07 and VOC'12

1) If incorporated with a proper way, more **powerful backbone CNN models** can definitely improve the object detection performance (the comparison among R-CNN with AlexNet vs R-CNN with VGG16).

2) With the introduction of the **end-to-end multitask architecture** (FRCN), and RPN (Faster R-CNN), object detection performance is improved gradually and apparently.

3) Due to a large number of trainable parameters, in order to obtain multilevel robust features, **data augmentation is very important** for deep learning-based models (Faster R-CNN).

4) Apart from basic models, there are still many other factors affecting object detection performance,

such as **multiscale and multi-region feature extraction, modified classification networks, additional information from other correlated tasks, multiscale representation, and mining of hard negative samples.**

5) As YOLO is **not skilled in producing object localizations of high IoU**, it obtains a very poor result on VOC 2012.

However, with the complementary information from Fast R-CNN (YOLO+FRCN), anchor boxes, BN, and fine-grained features, the localization errors are corrected (YOLOv2).

Comparison on Microsoft COCO

- Microsoft COCO is composed of 300 000 fully segmented images, in which each image has an average of 7 object instances from a total of 80 categories.
- As there are a lot of less iconic objects with a broad range of scales and a stricter requirement on object localization, this data set is more challenging than PASCAL 2012.
- Object detection performance is evaluated by AP computed under different degrees of IoUs and on different object sizes.

Methods	Trained on	0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [16]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.4	30.1	7.3	32.1	52.0
ION [95]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
NOC+FRCN(VGG16) [114]	train	21.2	41.5	19.7	-	-	-	-	-	-	-	-	-
NOC+FRCN(Google) [114]	train	24.8	44.4	25.2	-	-	-	-	-	-	-	-	-
NOC+FRCN (ResNet101) [114]	train	27.2	48.4	27.6	-	-	-	-	-	-	-	-	-
GBD-Net [109]	train	27.0	45.8	-	-	-	-	-	-	-	-	-	-
OHEM+FRCN [113]	train	22.6	42.5	22.2	5.0	23.7	34.6	-	-	-	-	-	-
OHEM+FRCN* [113]	train	24.4	44.4	24.8	7.1	26.4	37.9	-	-	-	-	-	-
OHEM+FRCN* [113]	trainval	25.5	45.9	26.1	7.4	27.7	38.5	-	-	-	-	-	-
Faster R-CNN [17]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
YOLOv2 [72]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
SSD300 [71]	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512 [71]	trainval35k	26.8	46.5	27.8	9.0	28.9	41.9	24.8	37.5	39.8	14.0	43.5	59.0
R-FCN (ResNet101) [65]	trainval	29.2	51.5	-	10.8	32.8	45.0	-	-	-	-	-	-
R-FCN*(ResNet101) [65]	trainval	29.9	51.9	-	10.4	32.4	43.3	-	-	-	-	-	-
R-FCN**(ResNet101) [65]	trainval	31.5	53.2	-	14.3	35.5	44.2	-	-	-	-	-	-
Multi-path [112]	trainval	33.2	51.9	36.3	13.6	37.2	47.8	29.9	46.0	48.3	23.4	56.0	66.4
FPN (ResNet101) [66]	trainval35k	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Mask (ResNet101+FPN) [67]	trainval35k	38.2	60.3	41.7	20.1	41.1	50.2	-	-	-	-	-	-
Mask (ResNet101+FPN) [67]	trainval35k	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	**51.2**	-	-	-	-	-	-
DSSD513 (ResNet101) [73]	trainval35k	33.2	53.3	35.2	13.0	35.4	51.1	28.9	43.5	46.2	21.8	49.1	66.4
DSSD300 [74]	trainval	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0

* FRCN*: Fast R-CNN with multi-scale training, R-FCN*: R-FCN with multi-scale training, R-FCN**: R-FCN with multi-scale training and testing, Mask: Mask R-CNN.

Conclusions

1) By computing **CNN features on shared feature** maps, test consumption is reduced largely.

- Test time is further reduced with the unified multitask learning (FRCN) and removal of additional region proposal generation stage (Faster R-CNN).

2) It takes **additional test time to extract multiscale** features and contextual information (Mask-RCNN).

3) It takes **more time to train a more complex and deeper** network (ResNet101 against VGG16) and this time consumption can be reduced by adding as many layers into shared fully convolutional layers as possible (FRCN).

4) Regression-based models can usually be **processed in real time** at the cost of a drop in accuracy compared with region proposal-based models.

- Also, region proposal-based models can be modified into real-time systems with the introduction of other tricks, such as Batch-Normalisation and residual connections.

5) Thanks to the usage of **transformers**, DETR has a simpler architecture and does not need post-processing on the outputs.

Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Cham: Springer International Publishing, 2020