Modern Convolutional Object Detectors

Faster R-CNN, R-FCN, SSD

29 September 2017

Presented by: Kevin Liang

Papers Presented

- *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (NIPS 2015)*

  Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun
- *R-FCN: Object Detection via Region-based Fully Convolutional Networks (NIPS 2016)*

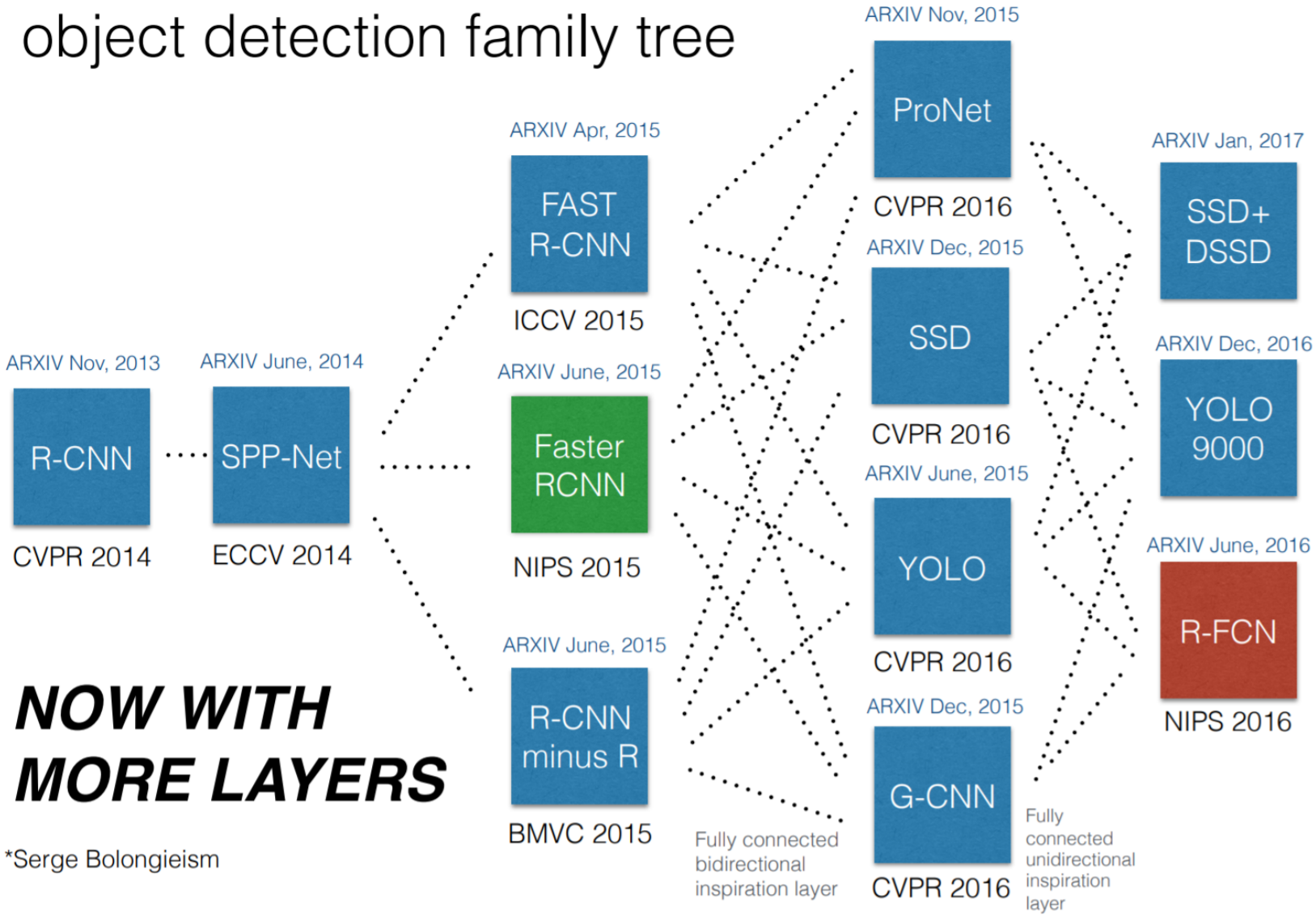
  Jifeng Dai, Yi Li, Kaiming He, Jian Sun
- *SSD: Single Shot MultiBox Detector (ECCV 2016)*

  Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C Berg
- *Speed/accuracy trade-offs for modern convolutional object detectors (CVPR 2017)*

  Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kevin Murphy

Object Detection Models

Some members of the *postdeepluvian** object detection family tree



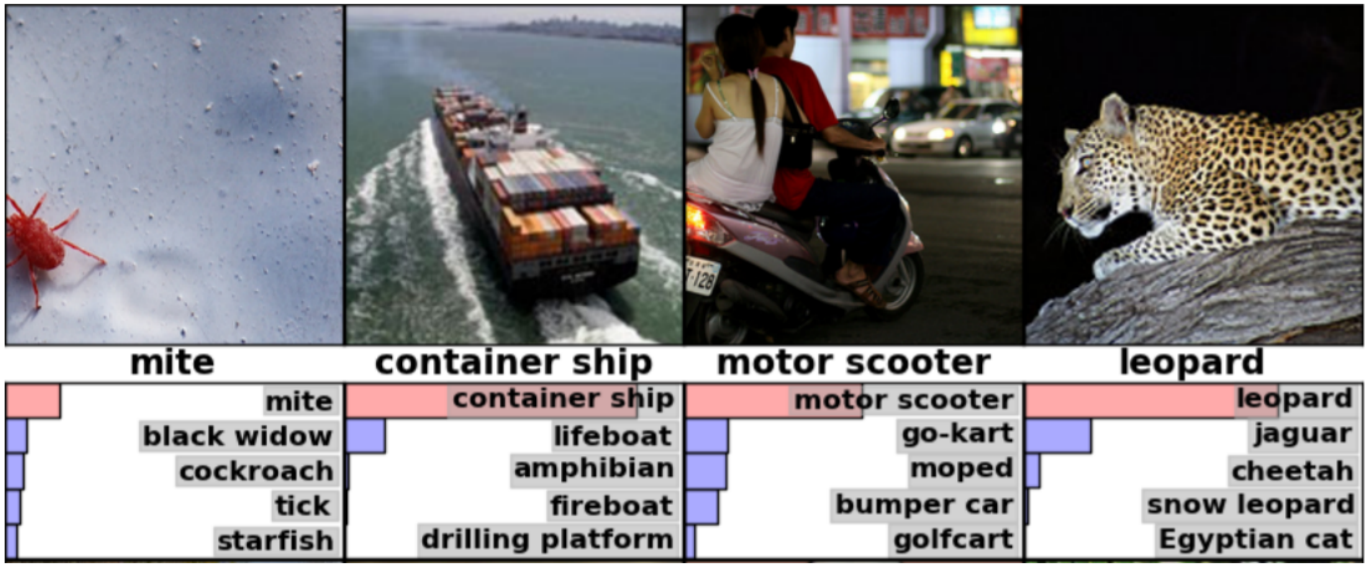
NOW WITH MORE LAYERS

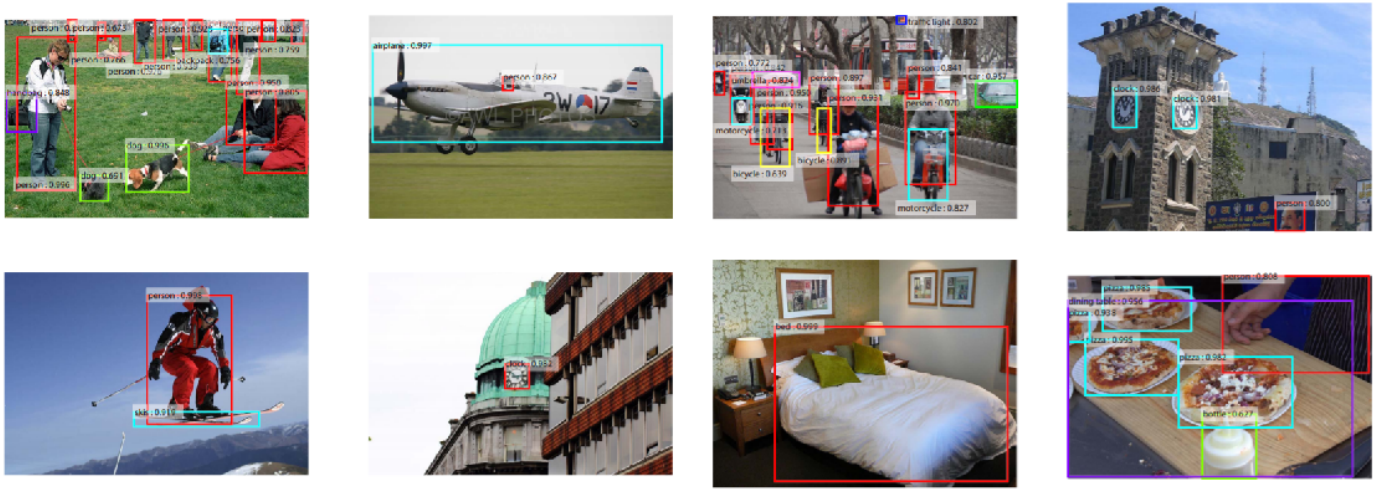
*Serge Bolongieism

Section 1

Background

Classification (top) vs Detection (bottom)

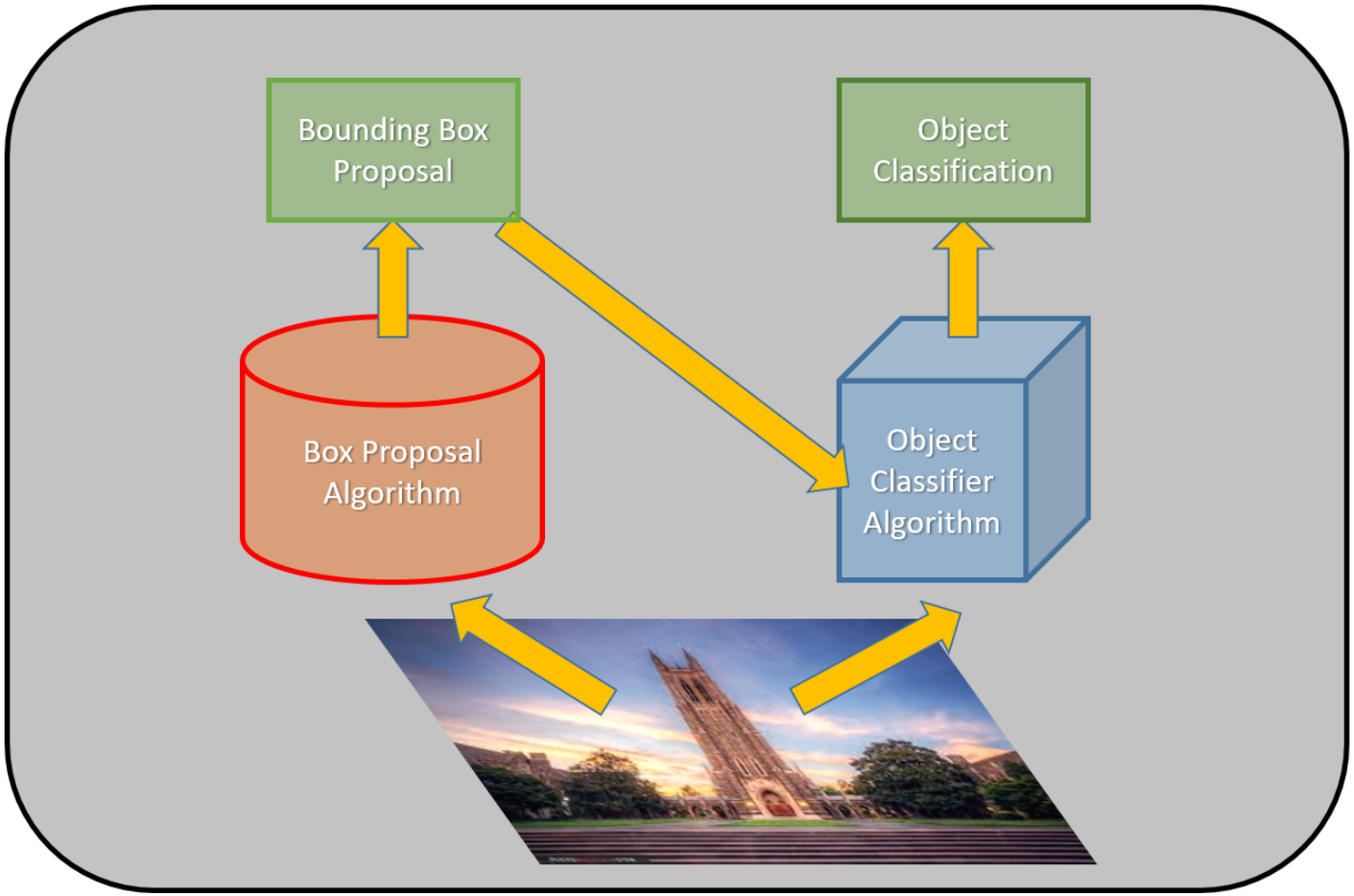




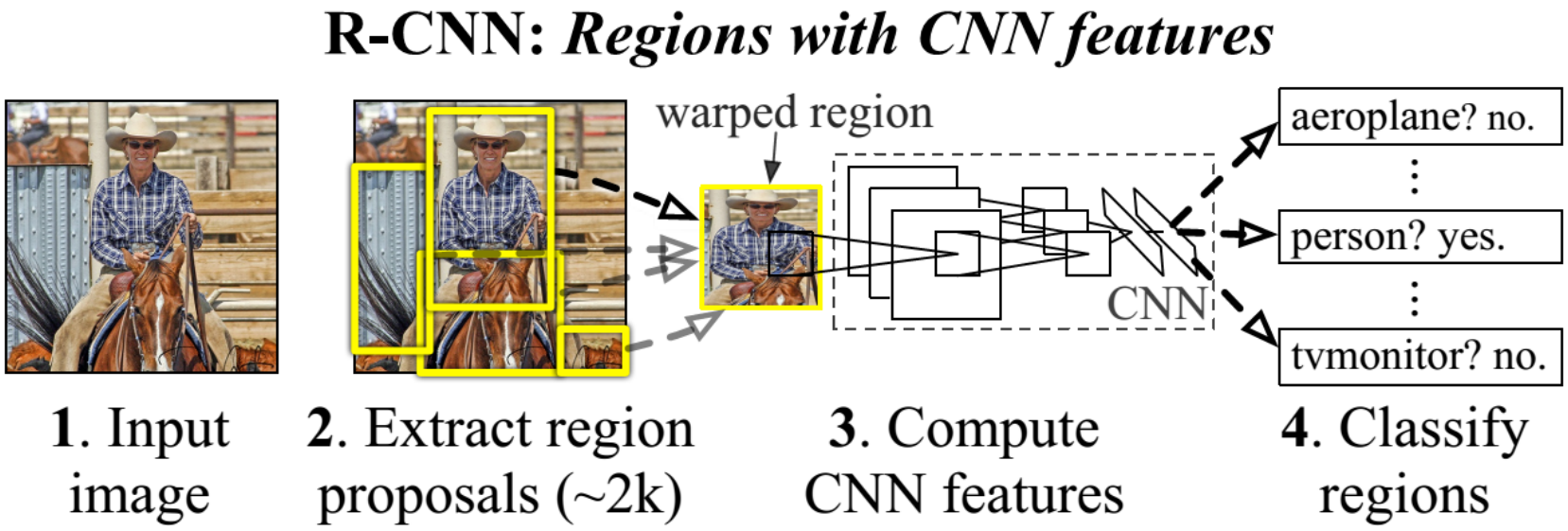
Two-stage Detection Paradigm

Many algorithms break up detection into two components:

- Box Proposal
- Object Classification



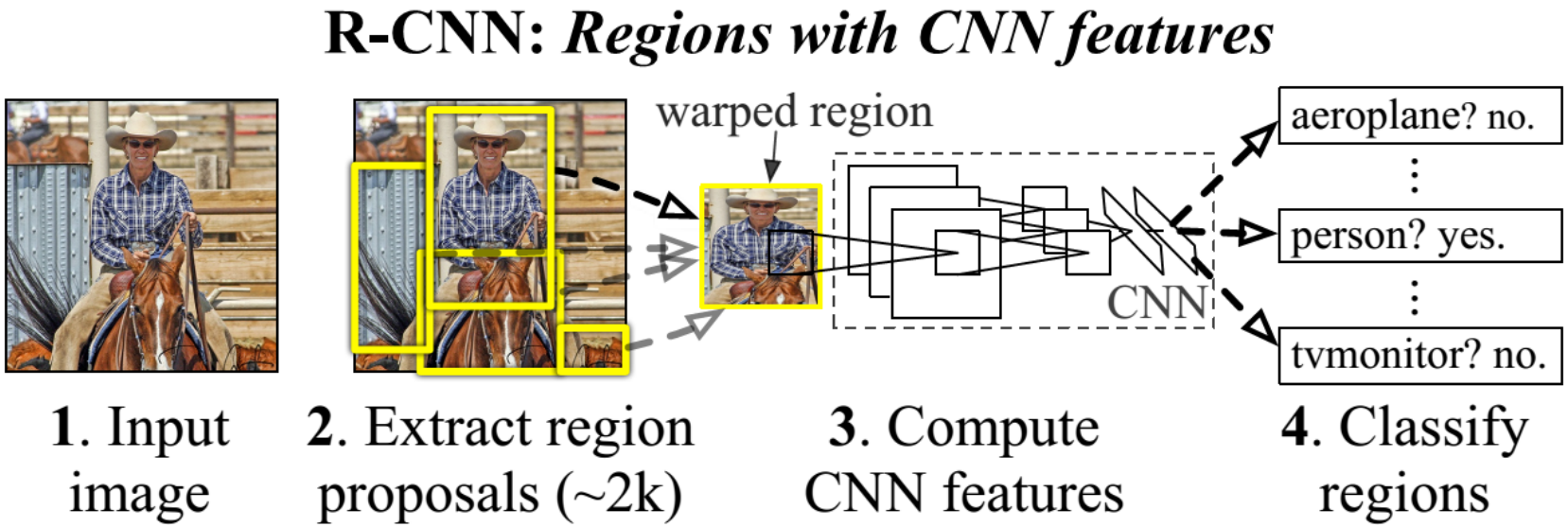
Regions with CNN features (R-CNN): 2013



Pros:

- Demonstrated ImageNet classification + AlexNet learned learned features also applicable for detection
- Significant jump in mAP performance on PASCAL VOC over state-of-the-art

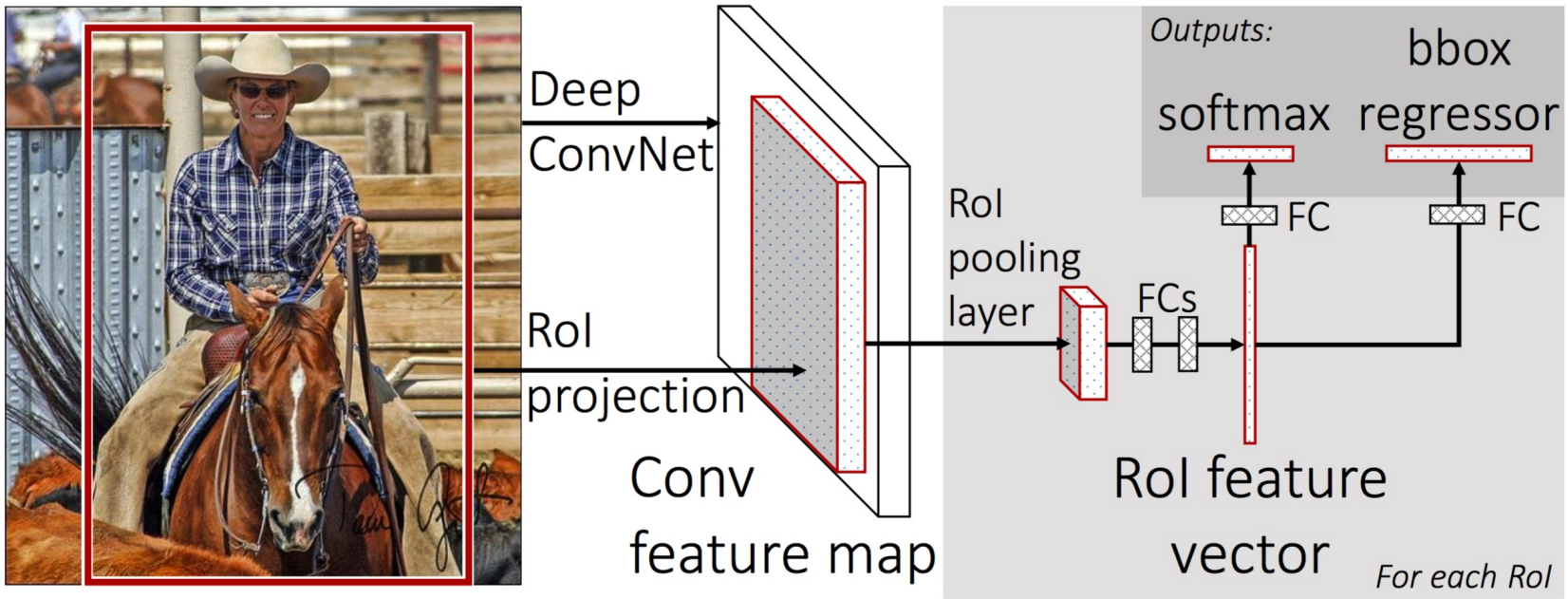
Regions with CNN features (R-CNN) (continued)



Cons:

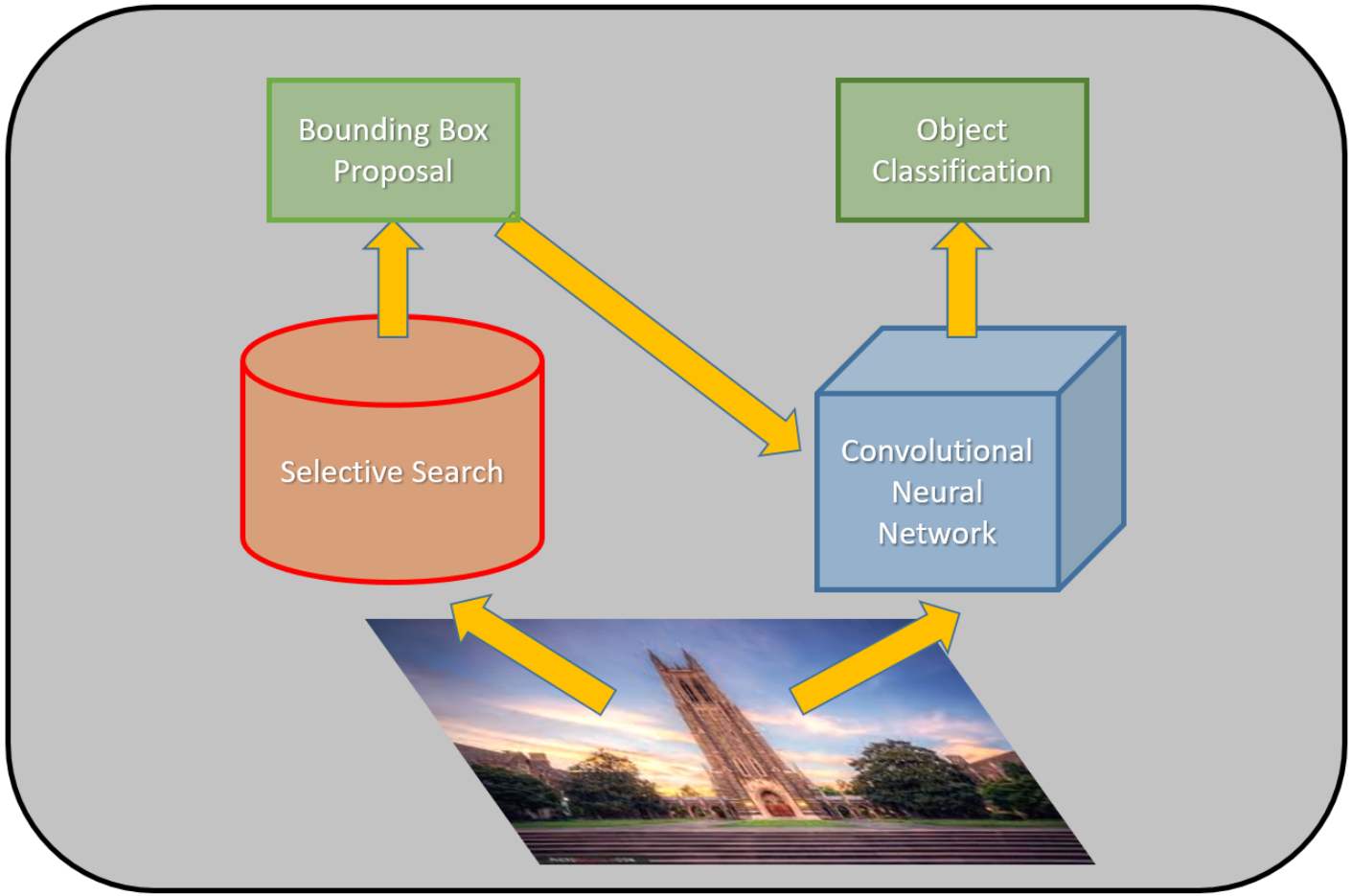
- Proposals generated by external algorithm (Selective Search)
- SVM and bounding box regressors trained separately; saving features to train these is memory intensive
- Patches extracted from raw image; overlapping proposals lead to repeated CNN computation for classification
- Detection takes ~47 s/image (GPU)

Fast R-CNN



- Single pass of entire image through CNN; patches extracted from convolutional feature maps
- Final classification and bounding box regression trained jointly with CNN
- Test time detection is 213x as fast as R-CNN
- *Proposals still generated by external algorithm*

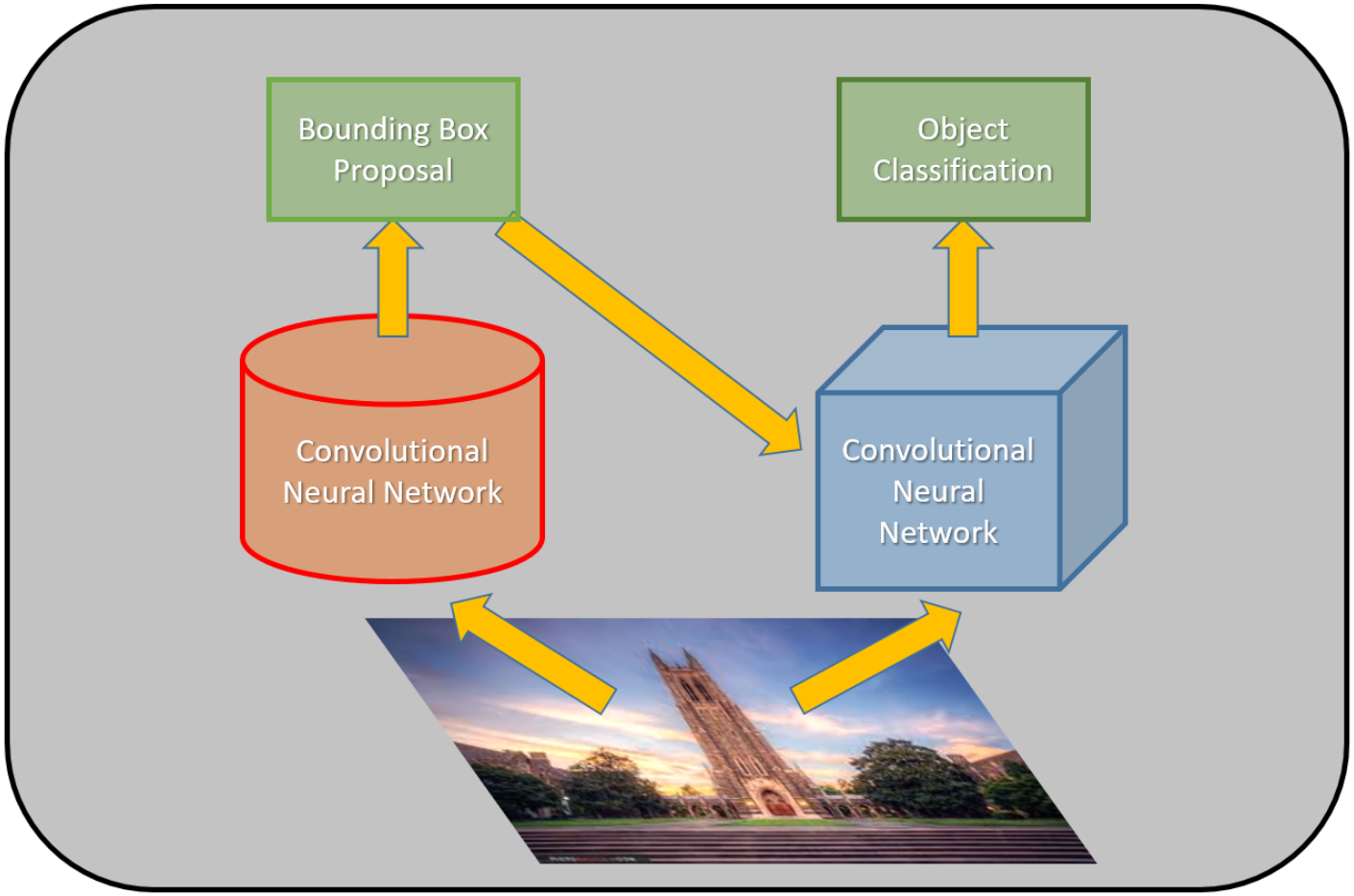
Two-stage Detection Paradigm - Revisited



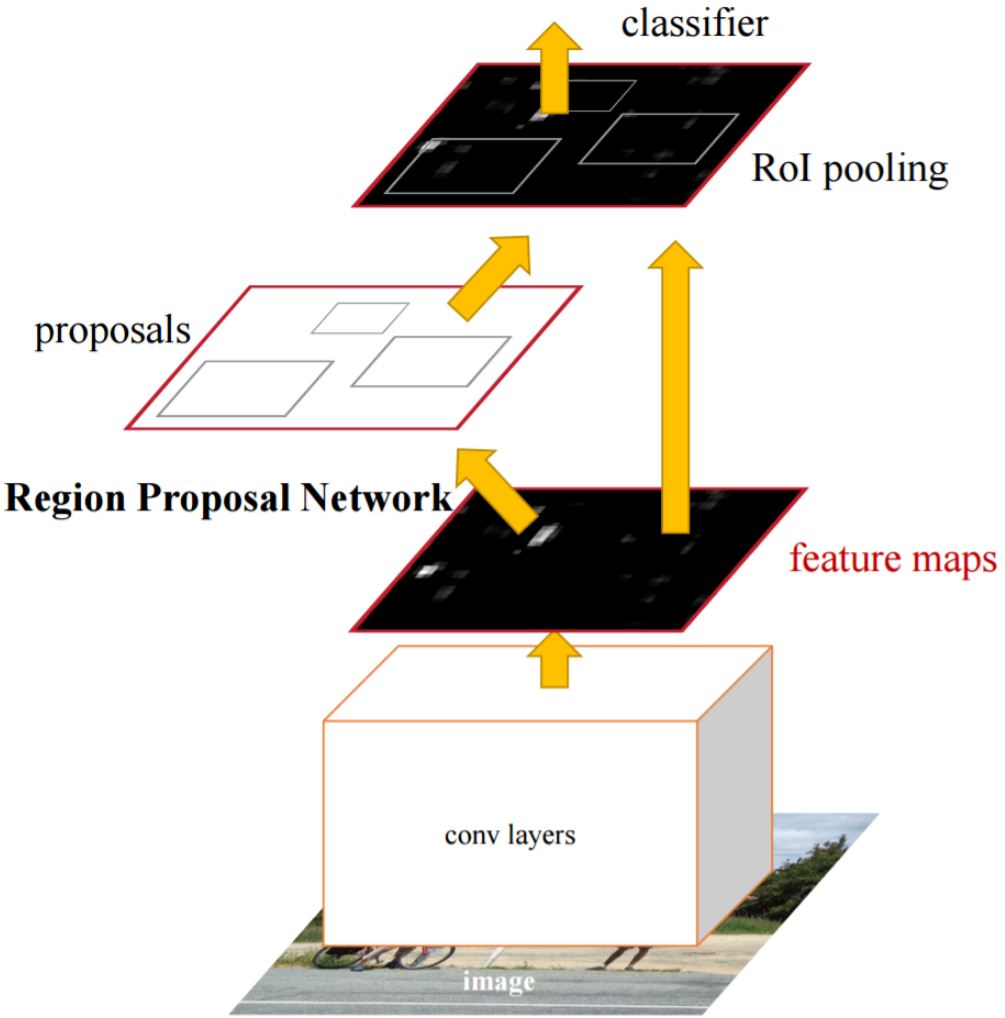
Section 2

Faster R-CNN

Two-stage Detection Paradigm - Revisited



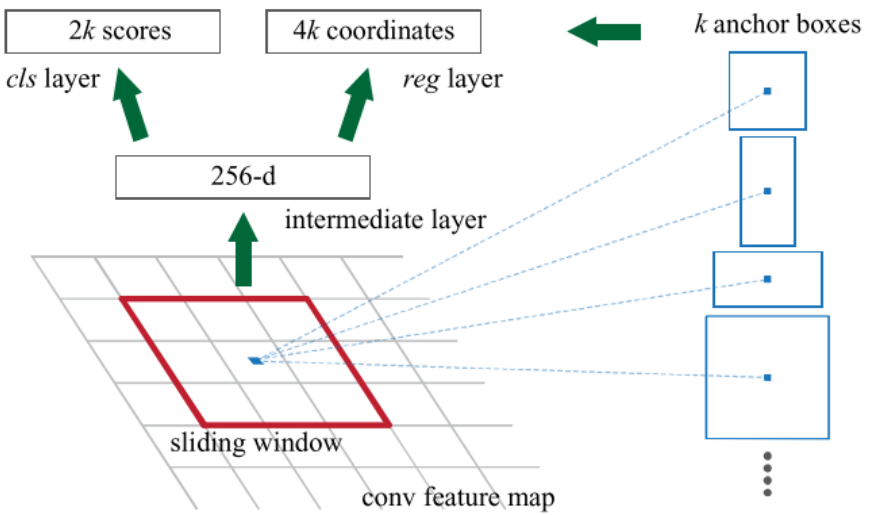
Faster R-CNN: Model Architecture

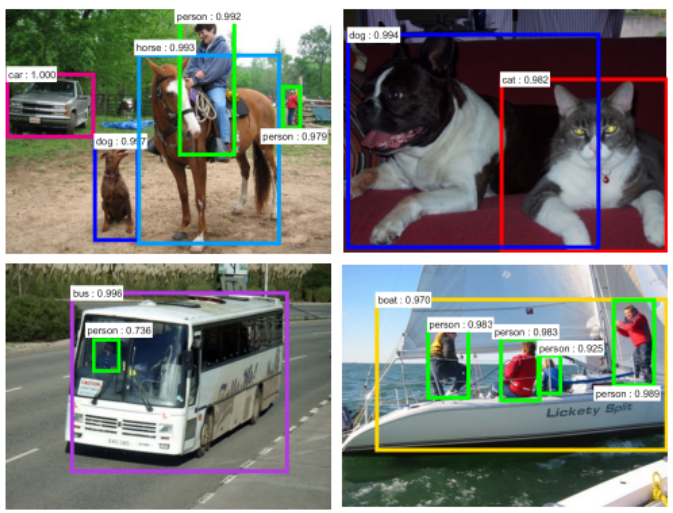


- Box proposal and classification share convolutional computation
- Patches extracted from convolutional feature map
- Detection $\sim$5 fps

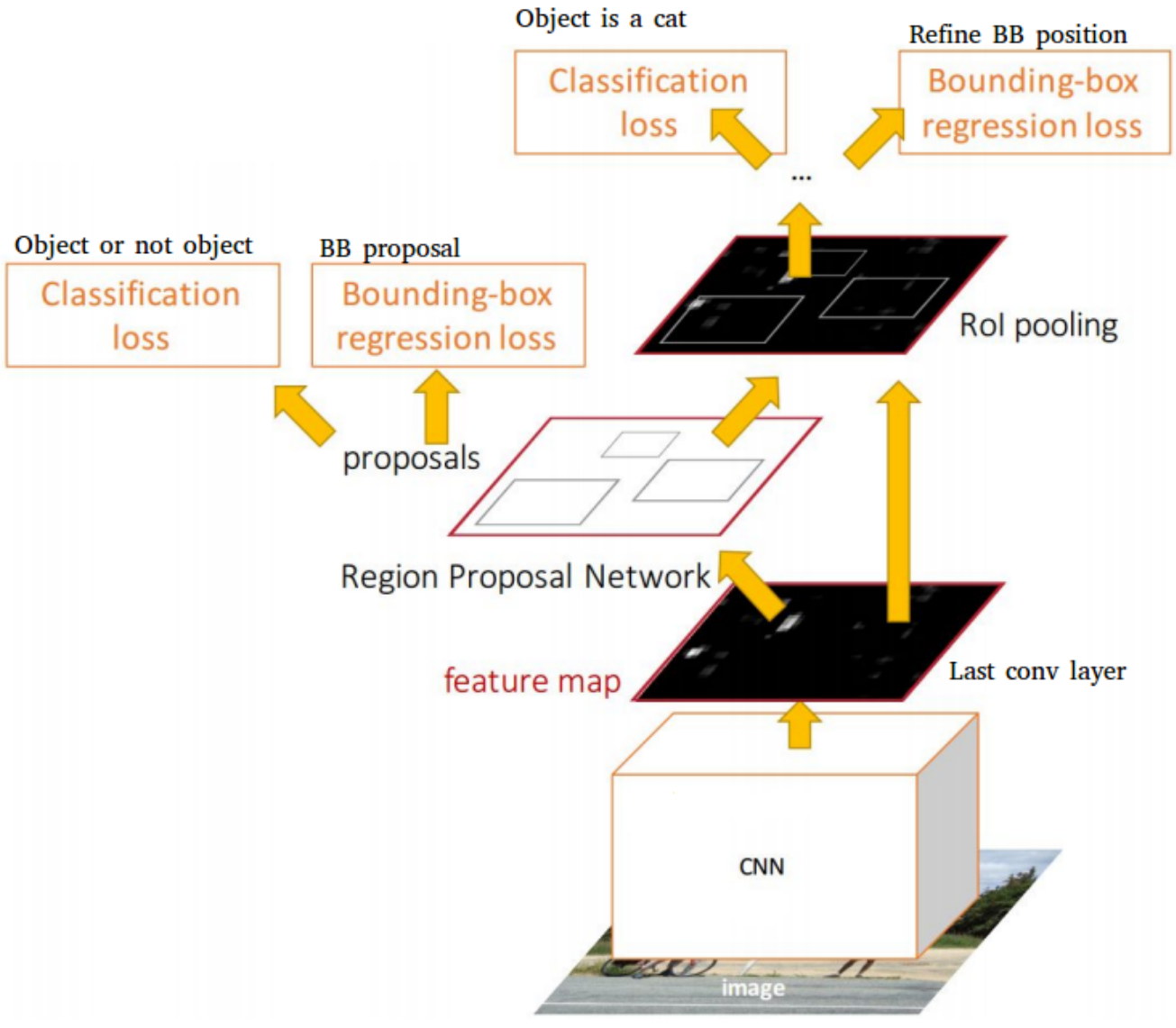
Faster R-CNN: Region Proposal Network

- Sliding window passed over final convolutional feature maps
- Box proposals are parameterized relative to fixed size reference boxes (termed "anchors")
  - Multiple scales and and aspect ratios (eg 1x1, 2x1, 1x2, 2x2, etc.)





Faster R-CNN: Training



Faster R-CNN: Loss

RPN Loss:

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

Bounding box parameterization:

$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a) \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a \\ t_w^* &= \log(w^*/w_a), & t_h^* &= \log(h^*/h_a) \end{aligned}$$

Localization loss:

$$L_{reg}(t, t^*) = \sum_{j \in x,y,w,h} \text{smooth}_{L1}(t_j - t_j^*),$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if} \quad |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Faster R-CNN: ResNet

ResNet weights:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Section 3

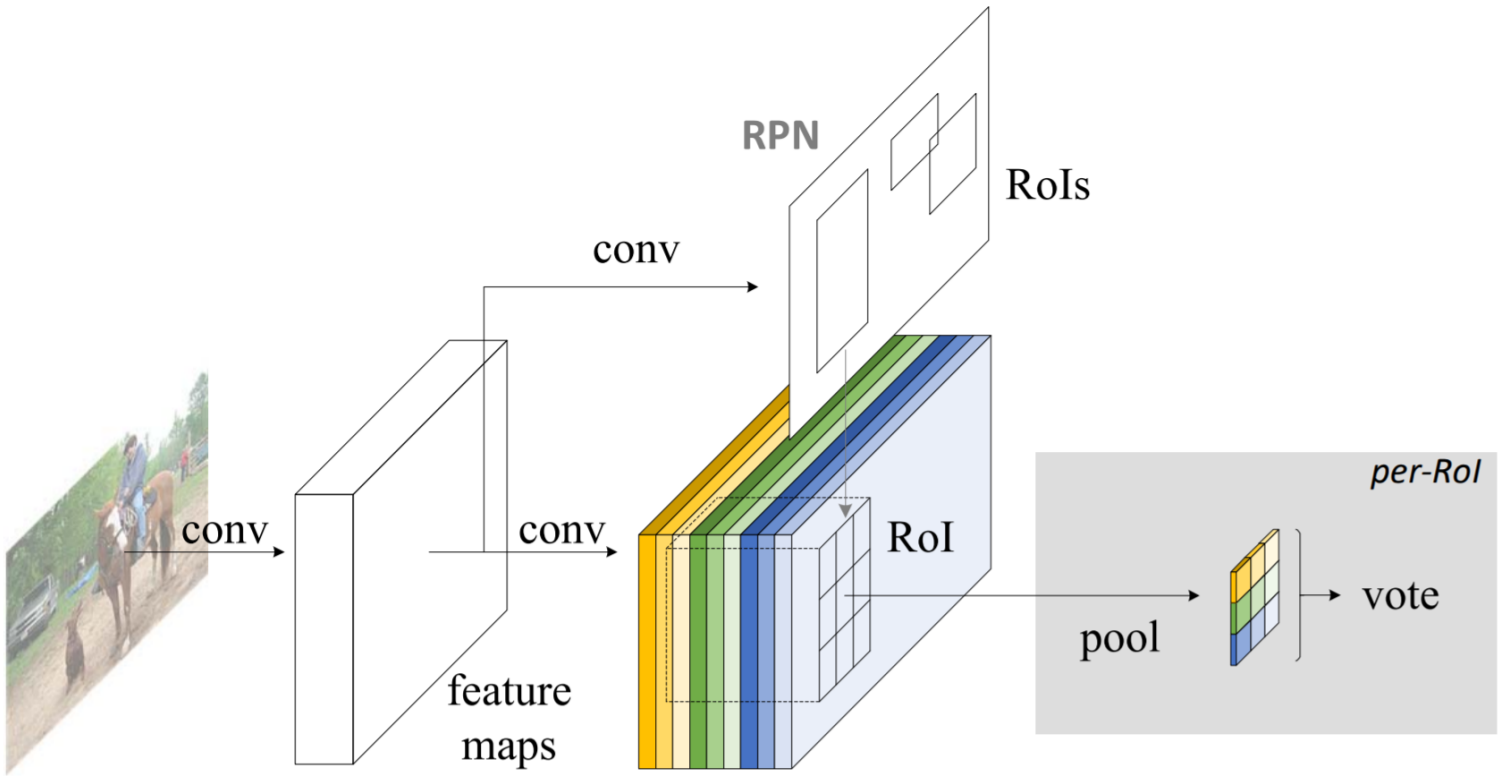
R-FCN

Region-based Fully Convolutional Networks: Inspiration

- Fast and Faster R-CNN save time by sharing computation of repeated convolutional features for object classification and region proposals, respectively
- However, Faster R-CNN still contains several unshared fully connected layers that must be computed for each of hundreds of proposals
- Dilemma of opposing needs for translation invariance (classification) and translation variance (localization)
  $\Rightarrow$ *Leads to unnatural insertion of region proposals between convolutional layers*

	R-CNN [7]	Faster R-CNN [19, 9]	R-FCN [ours]
depth of shared convolutional subnetwork	0	91	101
depth of RoI-wise subnetwork	101	10	**0**

Region-based Fully Convolutional Networks: Model Architecture



Region-based Fully Convolutional Networks: Visualization

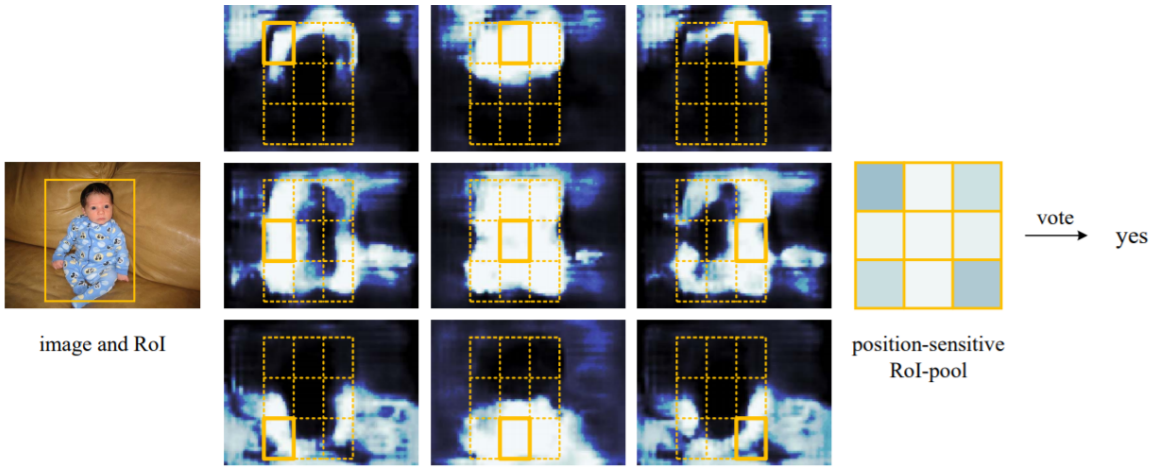
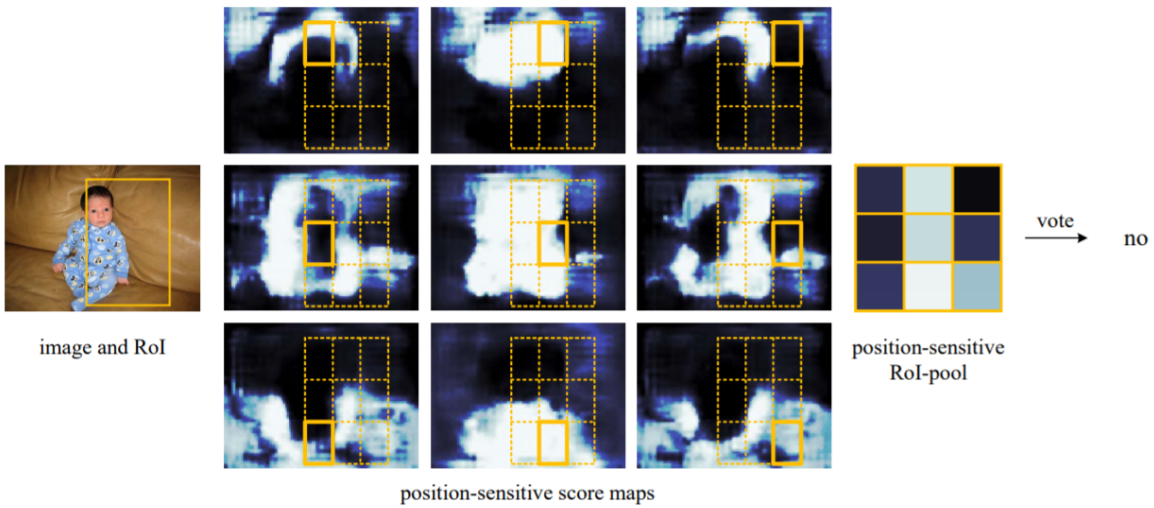


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.



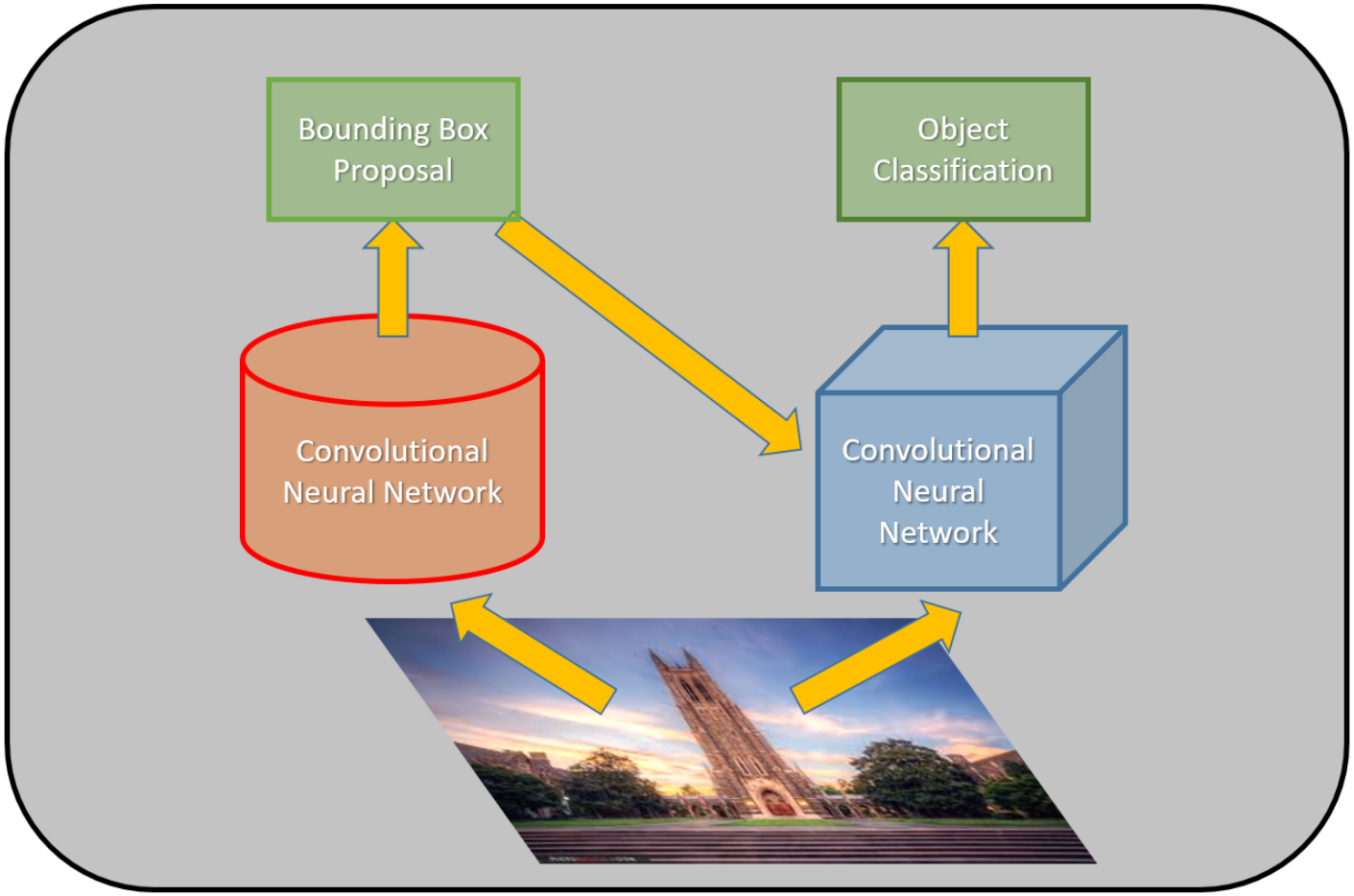
Region-based Fully Convolutional Networks: Benefits and Contributions

- 2.5-20x faster than Faster R-CNN
- Comparable accuracy
- Easy Online Hard Example Mining (OHEM)
- Use of atrous convolutions

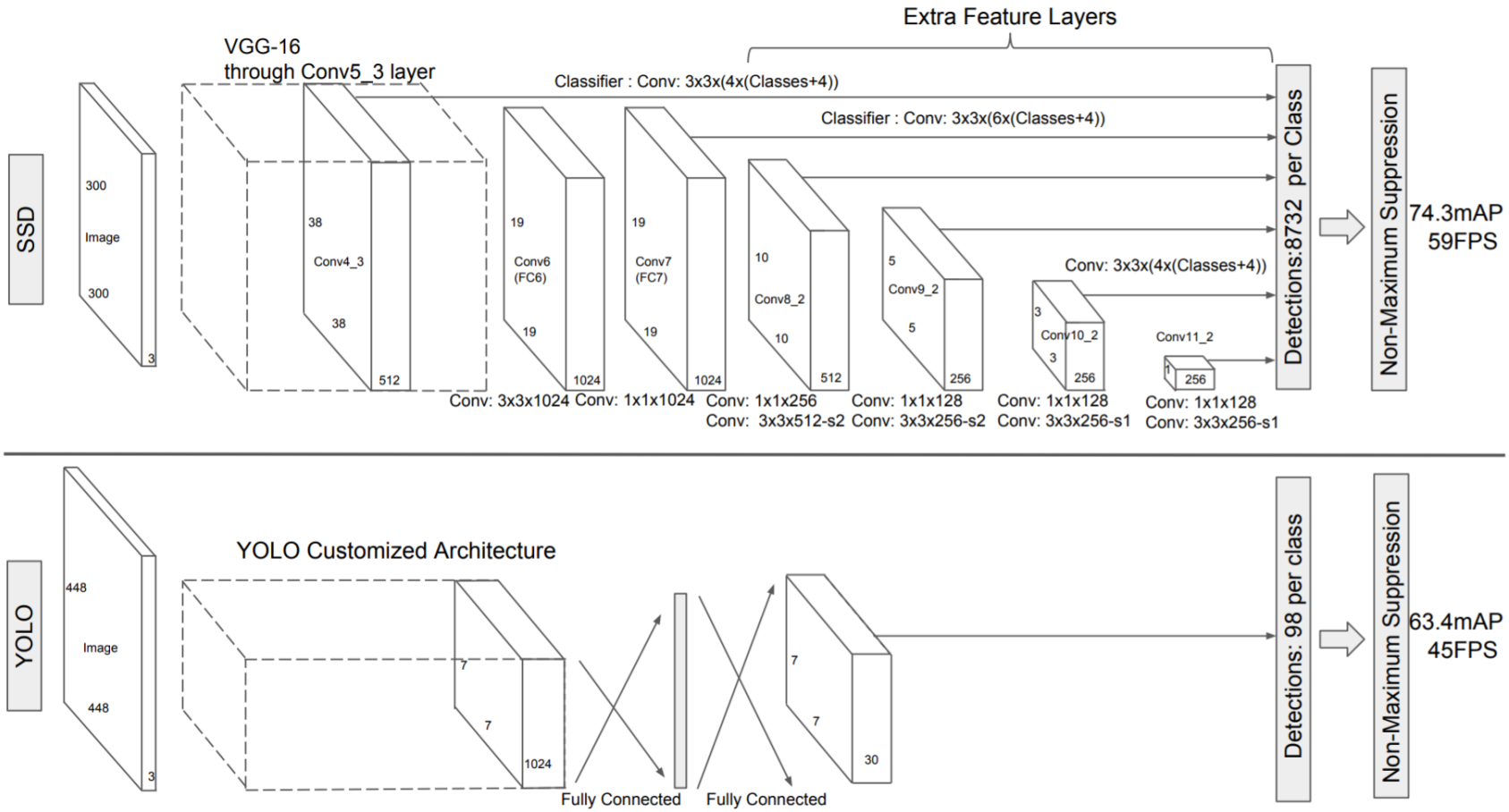
Section 4

SSD

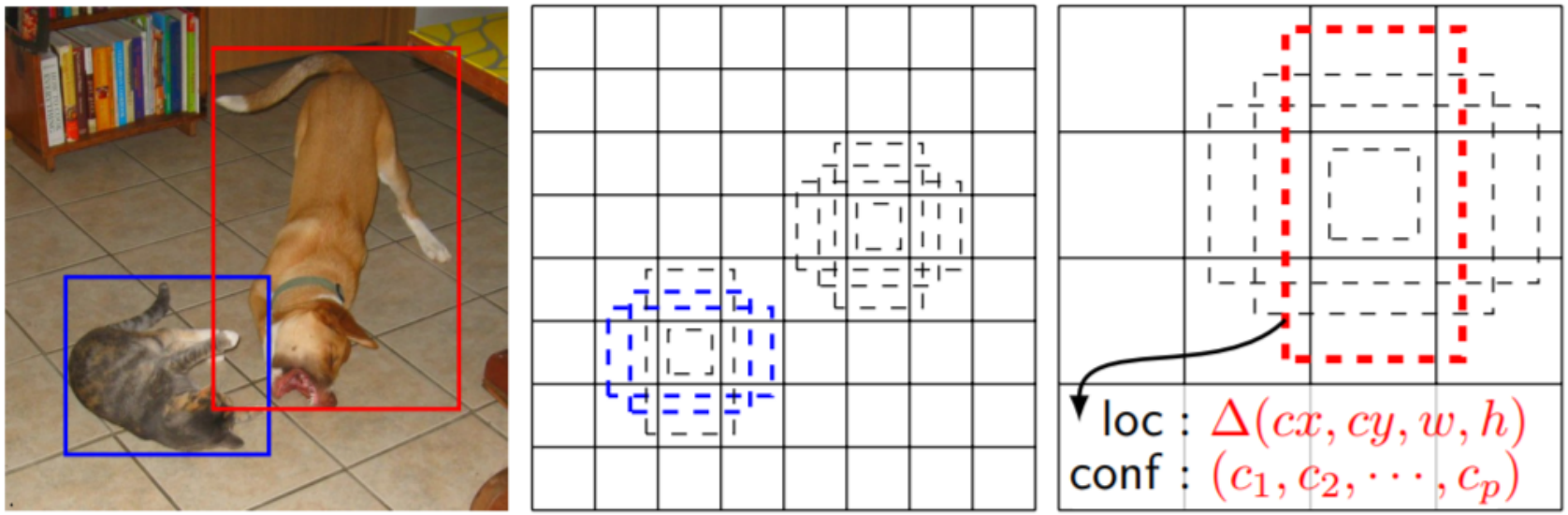
Two-stage Detection Paradigm - Revisited Again



Single Shot Multibox Detector: Model Architecture



Single Shot Multibox Detector: Default boxes

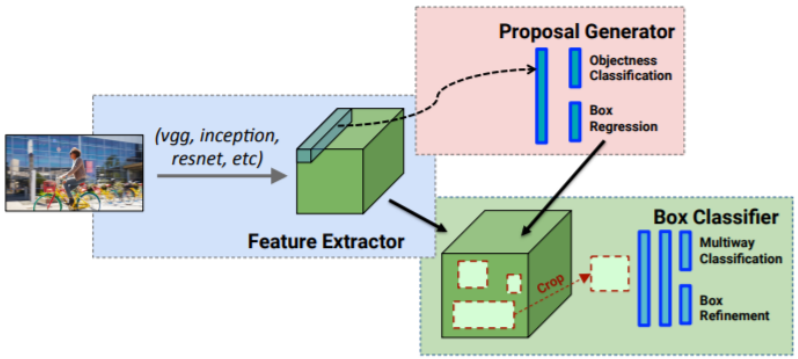


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

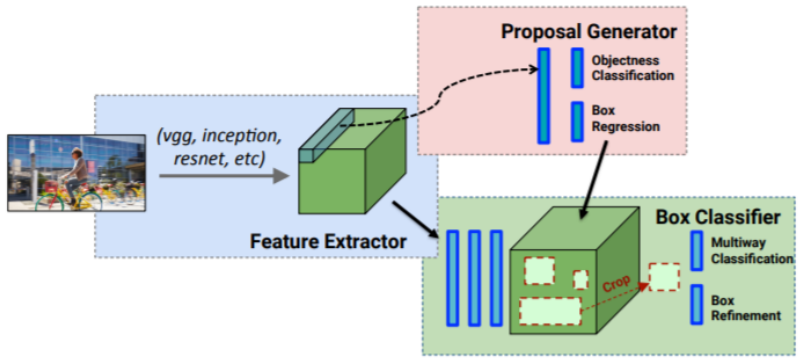
Section 5

Speed/Accuracy Comparison

CNNs and Meta Architectures Summary

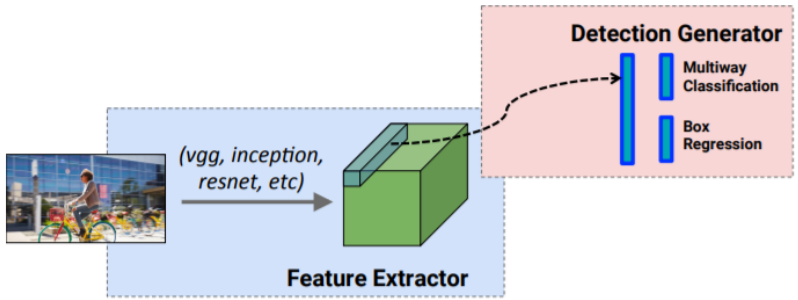


Faster RCNN



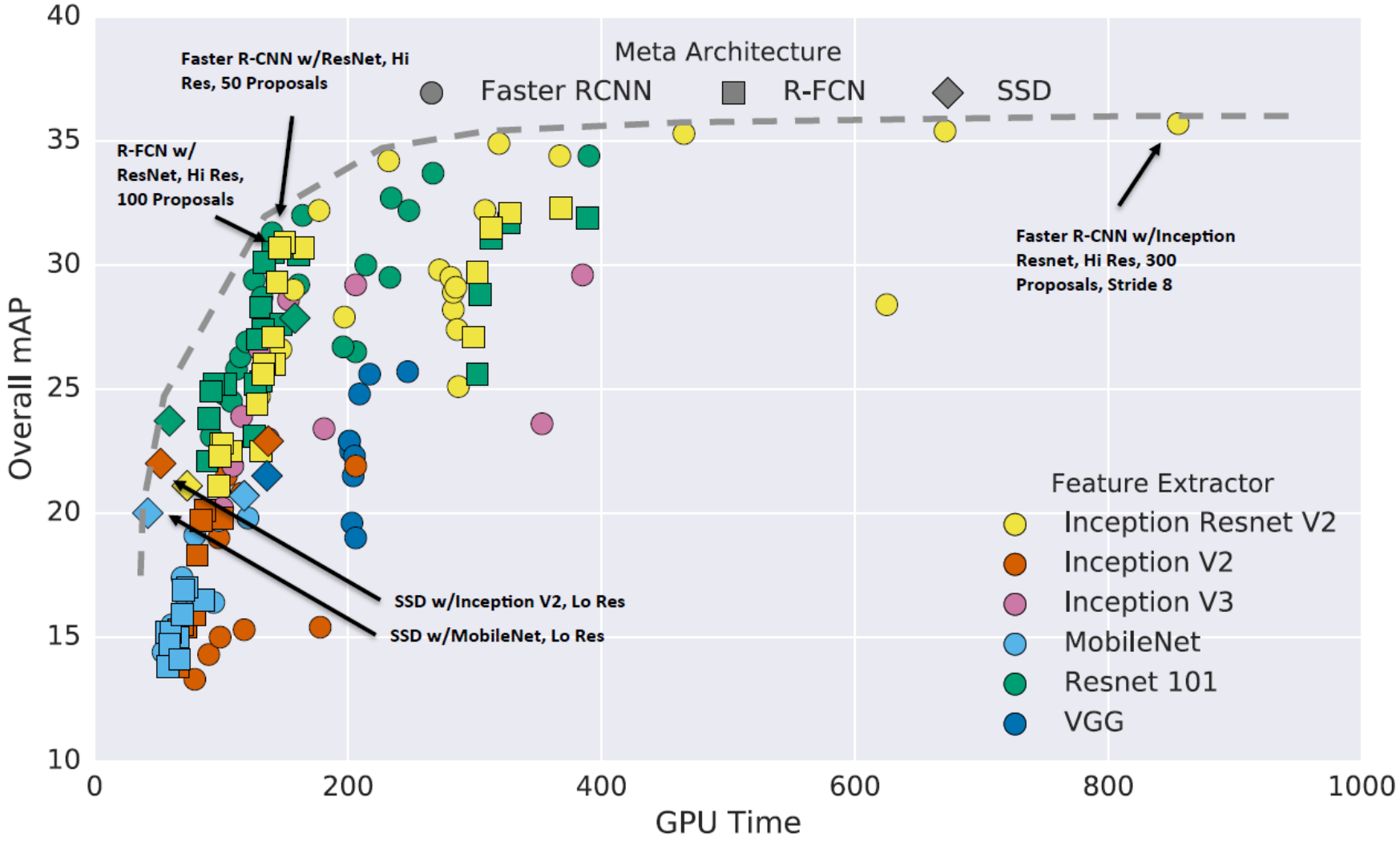
R-FCN

Model	Top-1 accuracy	Num. Params.
VGG-16	71.0	14,714,688
MobileNet	71.1	3,191,072
Inception V2	73.9	10,173,112
ResNet-101	76.4	42,605,504
Inception V3	78.0	21,802,784
Inception Resnet V2	80.4	54,336,736

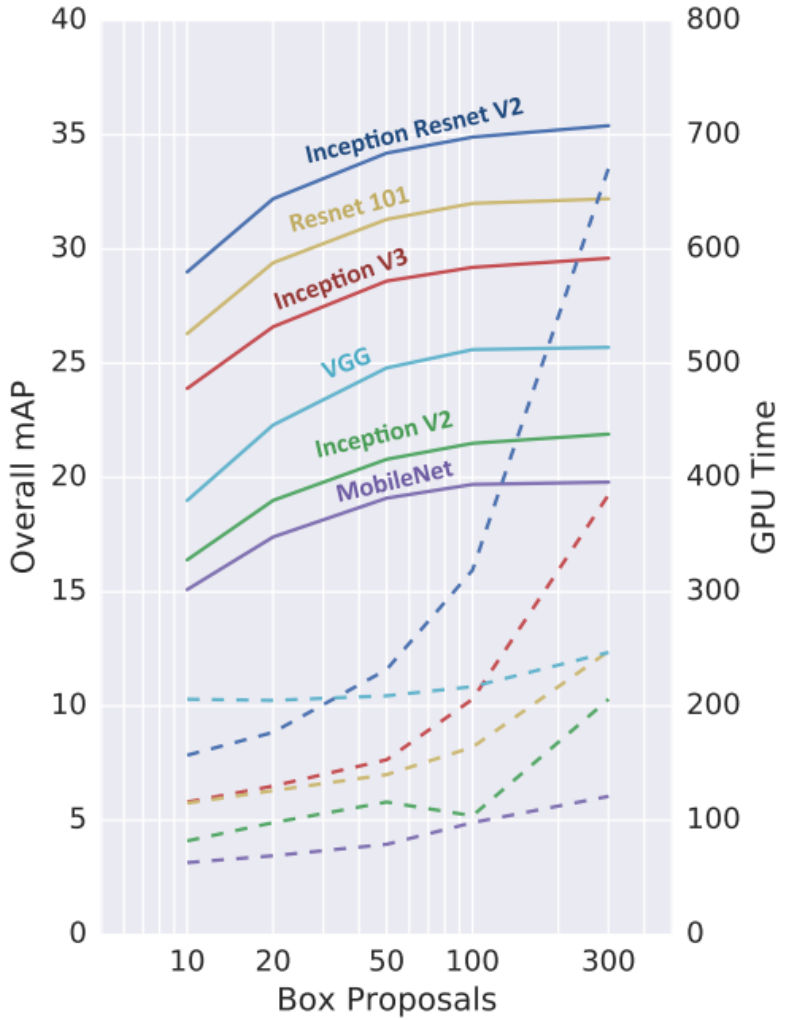


SSD

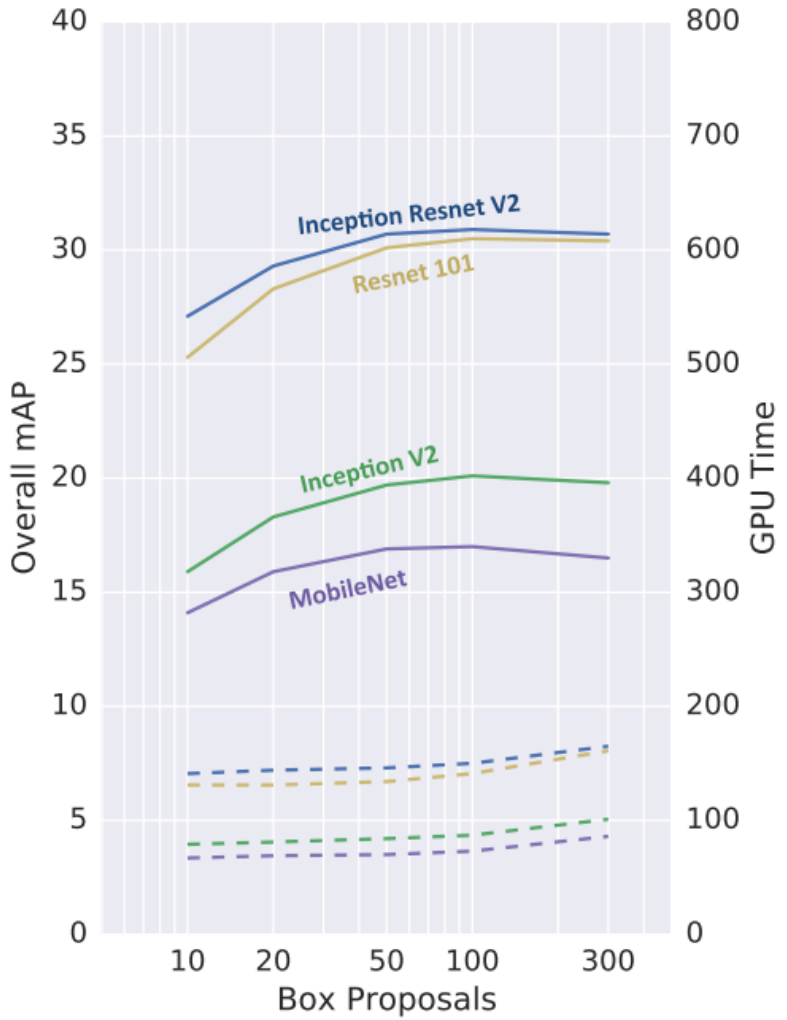
Accuracy vs Speed



Accuracy/Speed vs Box proposals



(a) FRCNN



(b) RFCN

Conclusions

- Found a speed-accuracy performance "frontier"
- For two-stage set-ups, more box proposals leads to better performance, but at the cost of speed and significantly diminishing returns.
- General correlation of convolutional feature extractor performance on ImageNet classification and performance as part of an object detector.
- Higher resolution images lead to higher quality localization, but at the cost of speed and memory. A similar trade-off exists with respect to convolutional feature extractor output stride (atrous convolutions).
- Won 2016 MS COCO object detection challenge by ensembling these implementations.

Open-Sourced TensorFlow Implementations

This repository Search Pull requests Issues Marketplace Explore

tensorflow / models

Watch 1,567 Star 21,402 Fork 9,715

Code Issues 358 Pull requests 120 Projects 0 Wiki Insights

Branch: master models / research / object_detection / Create new file Upload files Find file History

Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.