Object Detection with CNN

Roberto Paredes

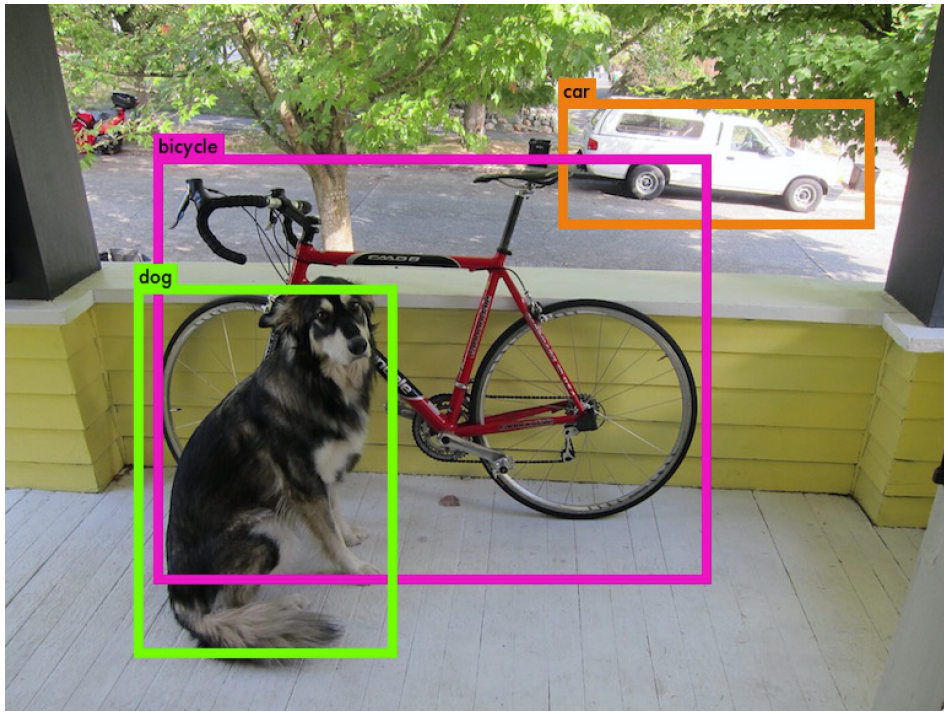
Centro de Investigación
Pattern Recognition and Human Language Technologies
Universidad Politécnica de Valencia

Index

- Introduction
- Region CNN
- Fast R-CNN
- Faster R-CNN

Introduction

- Object Detection is an important task in many computer vision problems
- State of the art was based on Fisher Vectors, SIFT, spatial pyramid...
- It started to claim attention to DL community



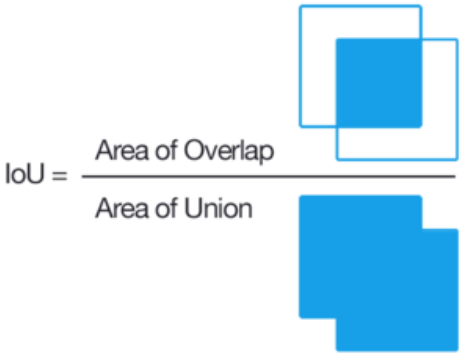
Introduction - Datasets

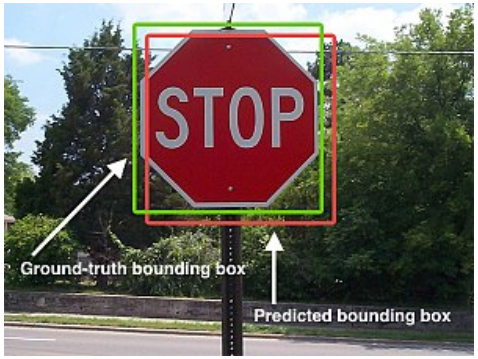
- Very important datasets and contests:
  - The PASCAL Visual Object Classes (VOC)
  - ImageNet Object Detection Challenge
  - Common Objects in Context (COCO)
    `http://cocodataset.org`

Introduction - Metrics

- Intersection over Union (Jackard Index):

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



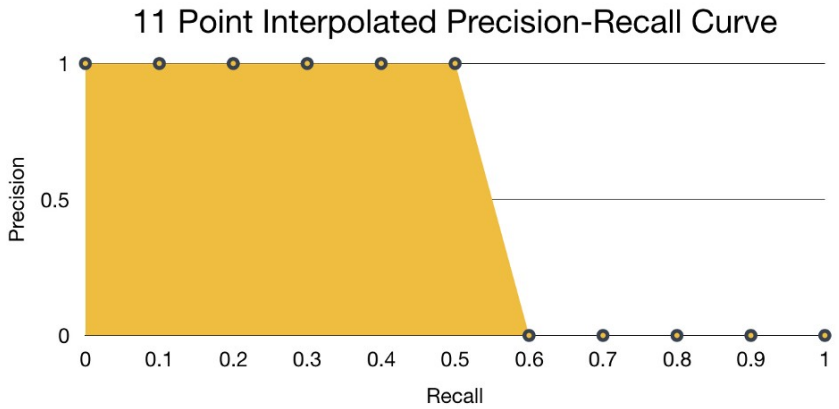


Introduction - Metrics

- Average Precision (AP)
- Need to meassure Precision and Recall for a given IoU threshold

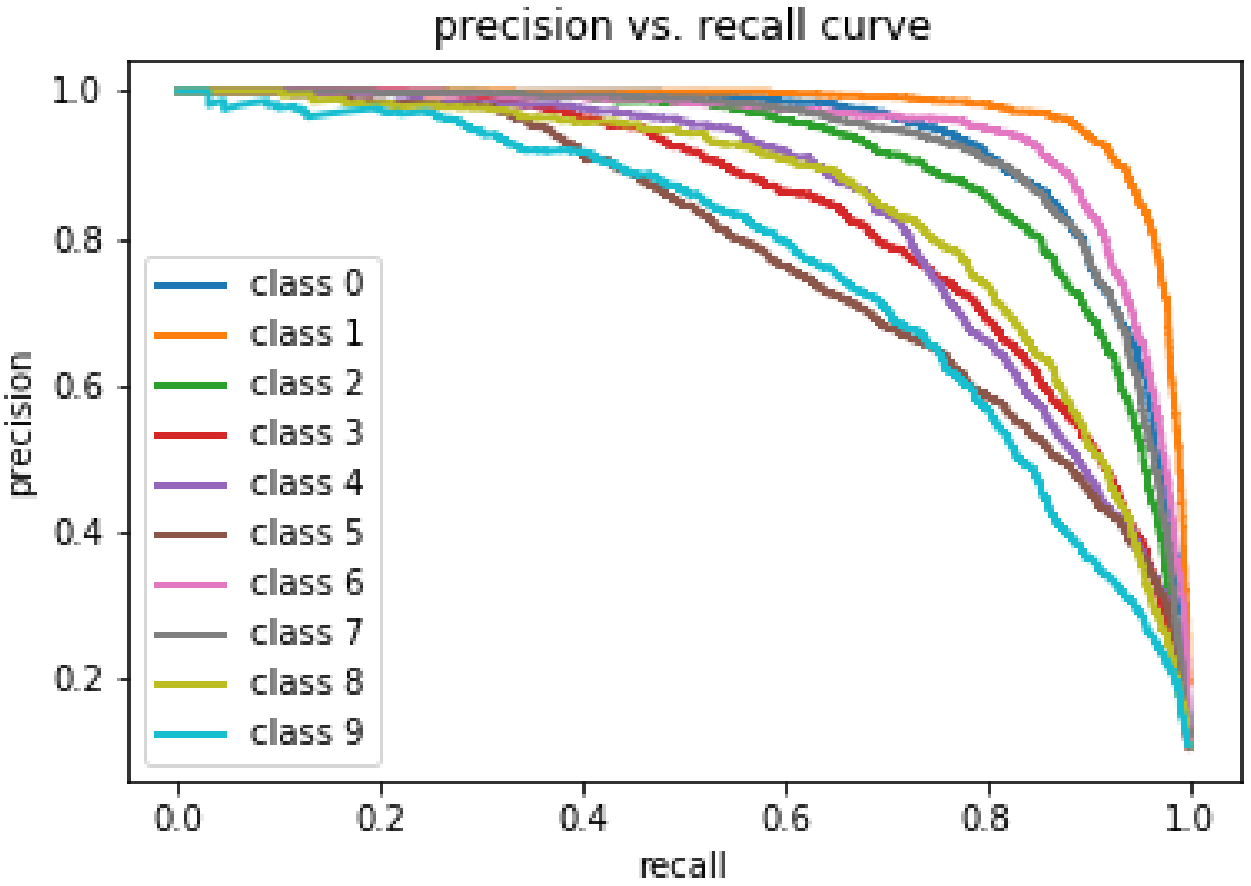
$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

- AP is the area under the Precision-Recall curve for different IoU thresholds



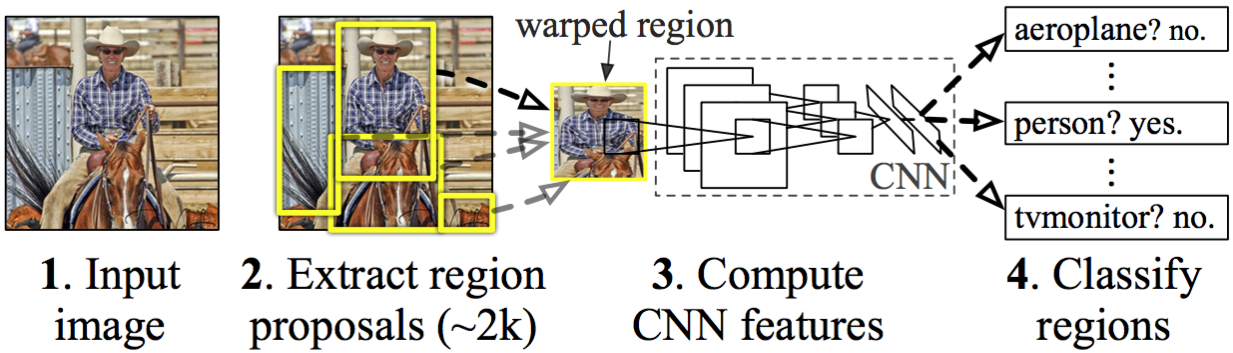
Introduction - Metrics

- Mean Average Precision (mAP)
- Is the mean AP for the different classes of objects to detect



Region CNN

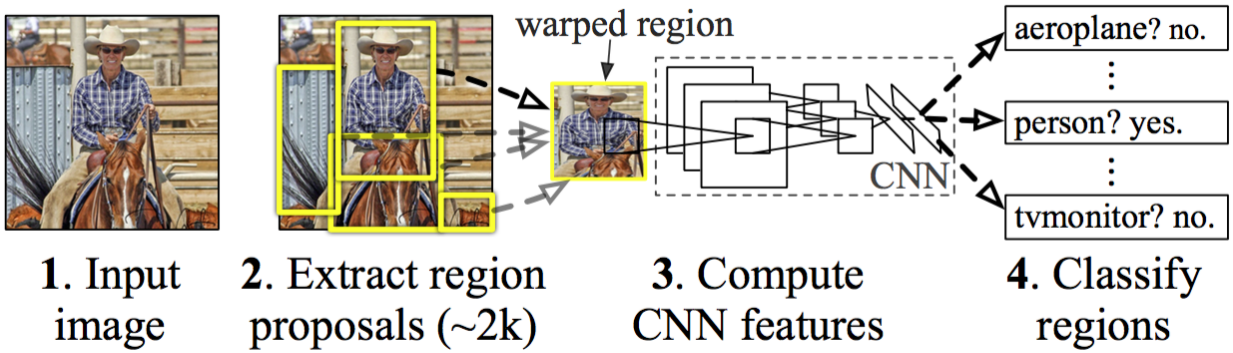
- How to apply Convolutional Networks?



- R-CNN achieves a mean average precision (mAP) of 53.7%
- State of the art was 35.1% **using the same region proposals**

Region CNN

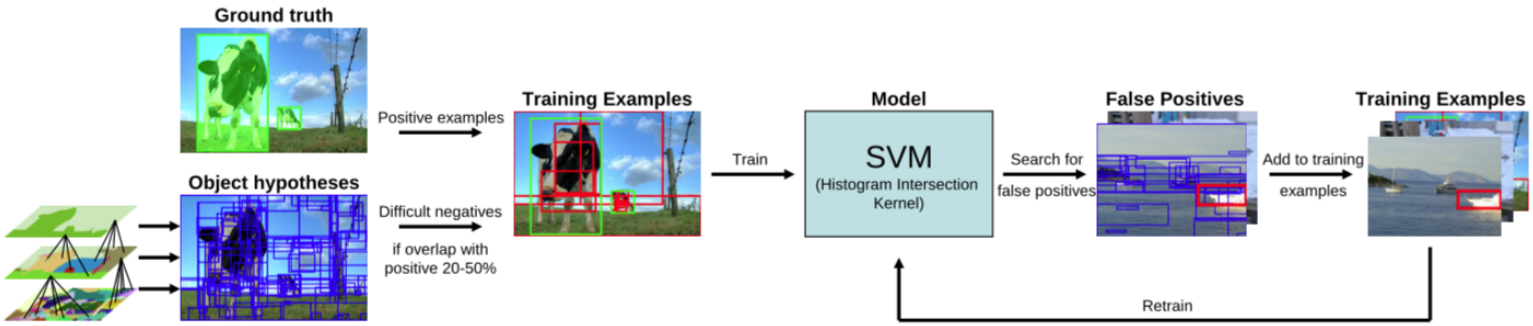
- Extract regions (≈ 2000). Image resizing to fit CNN
- Represent each region with a fixed-size vector. FC layer from Alexnet



- Classify with a Linear SVM

Region CNN - Region proposal

- A crucial point is how to extract regions
- Selective Search for Object Recognition



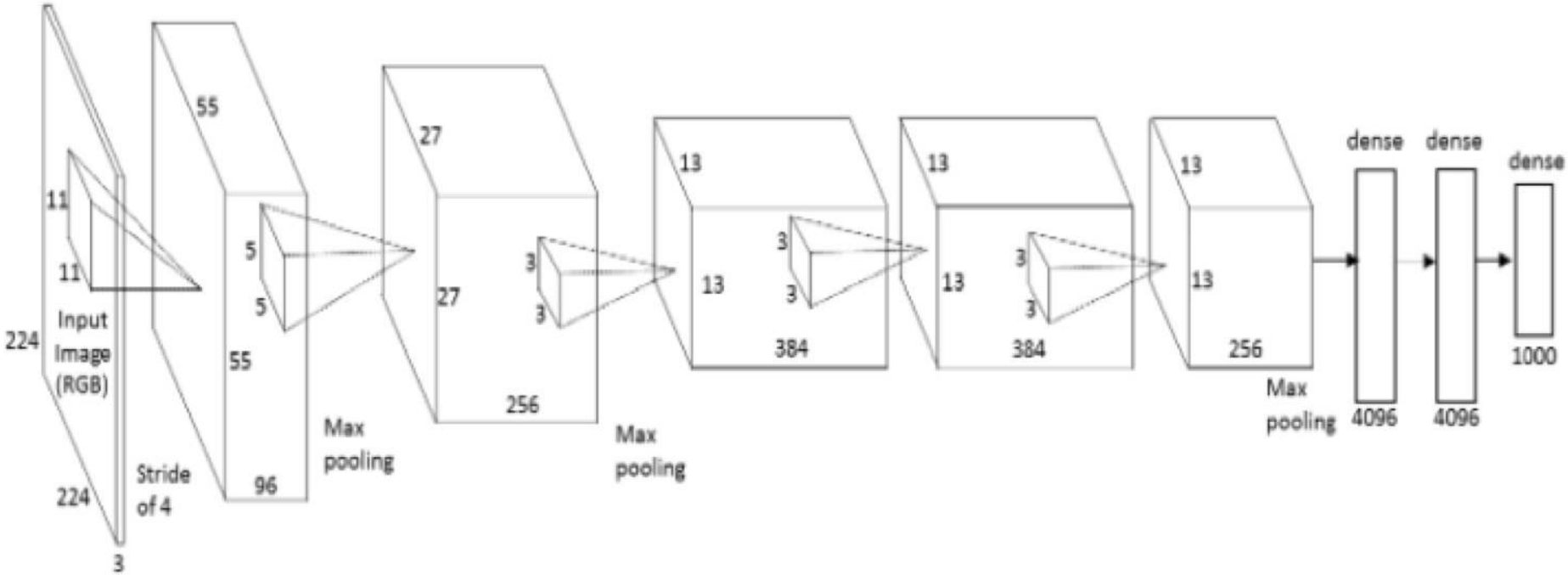
https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf

Region CNN - Training

- Training set:
  `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/examples/index.html`
- A set of images
- A list of pairs:
  - Bounding boxes
  - Class labels (20 classes)

Region CNN - Training

- Labeled data is scarce:
- CNN from a Pre-training (AlexNet) over Imagenet



Region CNN - Training

- Domain-specific fine-tuning on a small dataset (PASCAL):
  - 21 softmax classification layer (for the 20 VOC classes plus background)
  - Positives samples are region proposals with IoU$\geq$ 0.5
  - mini-batch of size 128
    - 32 positive windows (over all classes)
    - 96 background windows
  - SGD learning-rate=1/10 learning rate of pre-training

Region CNN - Training

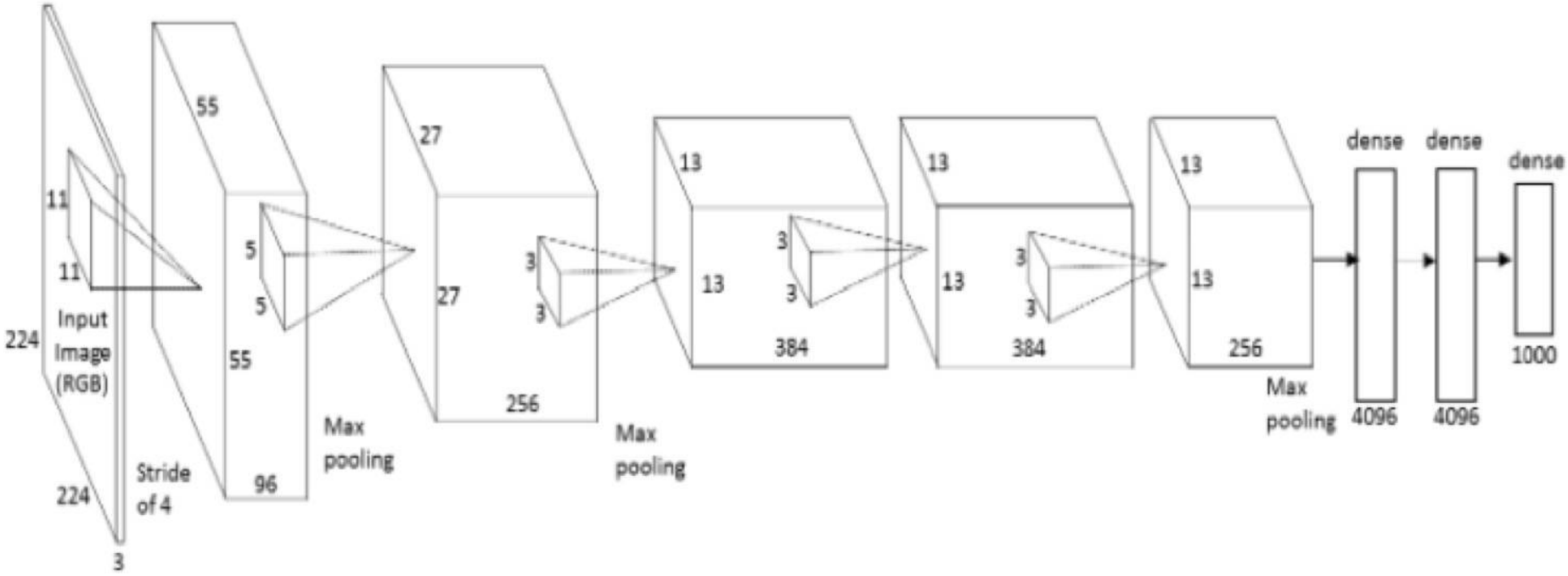
- Binary classifier (Linear SVM):
  - Region with object should be labeled as positive
  - Region without object should be labeled as negative
  - Region with partial object?
    - IoU below 0.3 is considered negative
    - critical parameter, e.g. 0.5 reduces mAP in 5 points
  - Linear SVM is trained with these labels

Region CNN - Test

- Extract regions
- Each region is mean-subtracted
- Warp each region (plus some extra context p=16 pixels) to fit CNN size (227x227 RGB image)
- Each image goes through five convolutional layers and two FC
- Last FC layer is used as representation 4096 features (in front of 360k)

Region CNN - Test

- Alexnet FC as feature vector

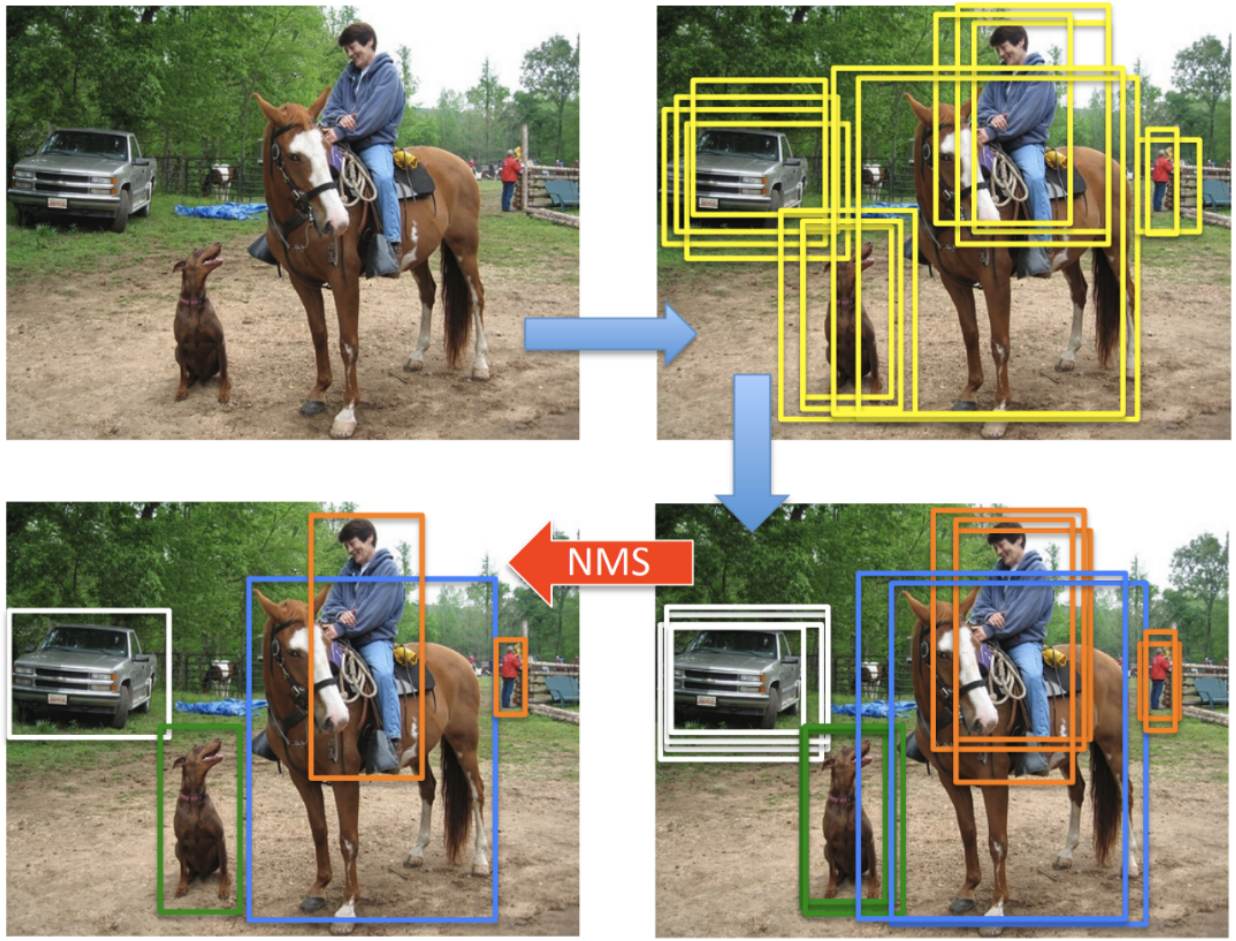


Region CNN - Test

- Linear SVM over these 4096 feature vector
- Matrix computations for the weights, 4096×21 matrix
- Each region gets a score
- Greedy non-maximum suppression (**for each class independently**)

Region CNN - Test

- NMS: non-maximum suppression to combine detections



Region CNN - Test Summary

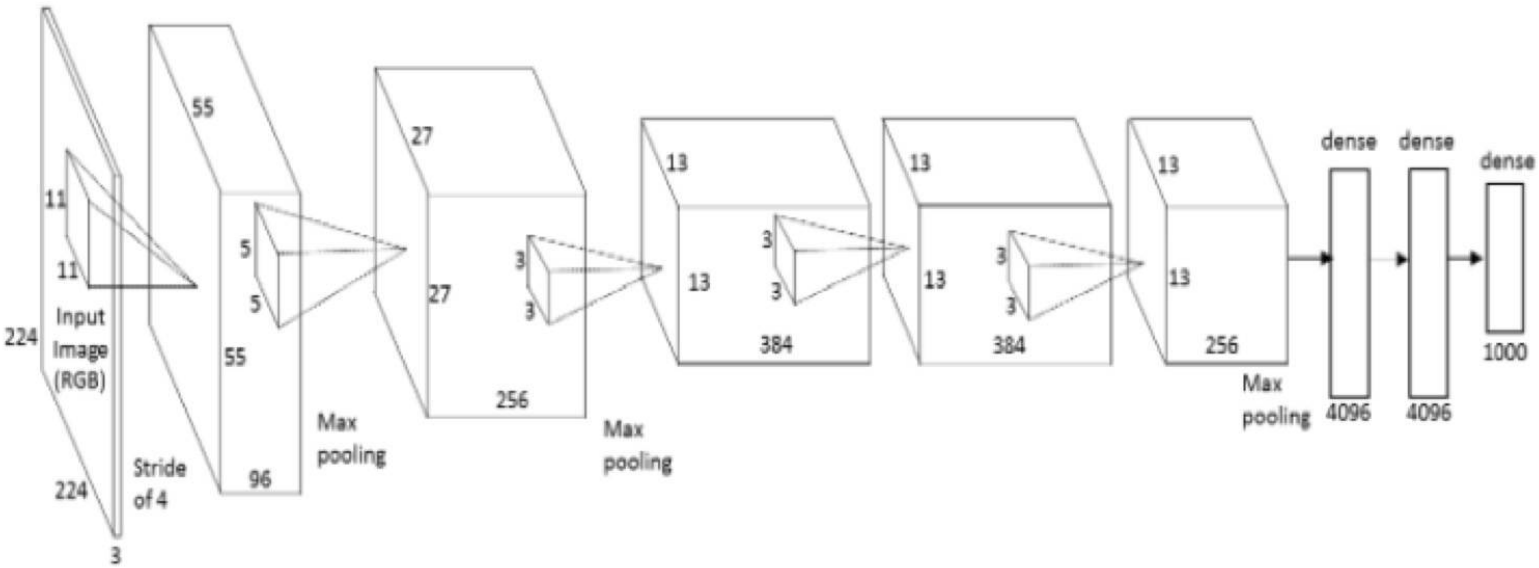
- Extract regions
- Each region is mean-subtracted and resized to 227x227 RGB image
- Each image goes through five convolutional layers and two FC
- Last FC layer is used as representation 4096 features (in front of 360k)
- Linear SVM over this vector
- NMS for merging regions
- Approx 1 minute per image

However, what we need?

`https://www.youtube.com/watch?v=VOC3huqHrss`

Region CNN - Options

- No finetunning
- Layer to extract features:
  - $Pool_5$ (approx. 6% of parameters)
  - FC_6
  - FC_7



Region CNN - Results

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT $pool_5$	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2

* FT, finetunning

Region CNN - Analysis

- Where come the errors from?
  - more of the errors result from poor localization
  - low IoU with ground-truth
  - features are discriminative (no significant confusion among classes)
- Better bounding boxes of object detected
- However comes from region proposal

Region CNN - Solution

- train a linear regression model to predict a new detection window
- from the $Pool_5$ features for a selective search region proposal
- *Object detection with discriminative trained part based models*:
  - Using functions that map a feature vector to the upper-left (x_1, y_1) and and lower-right (x_2, y_2), corners of the bounding box.

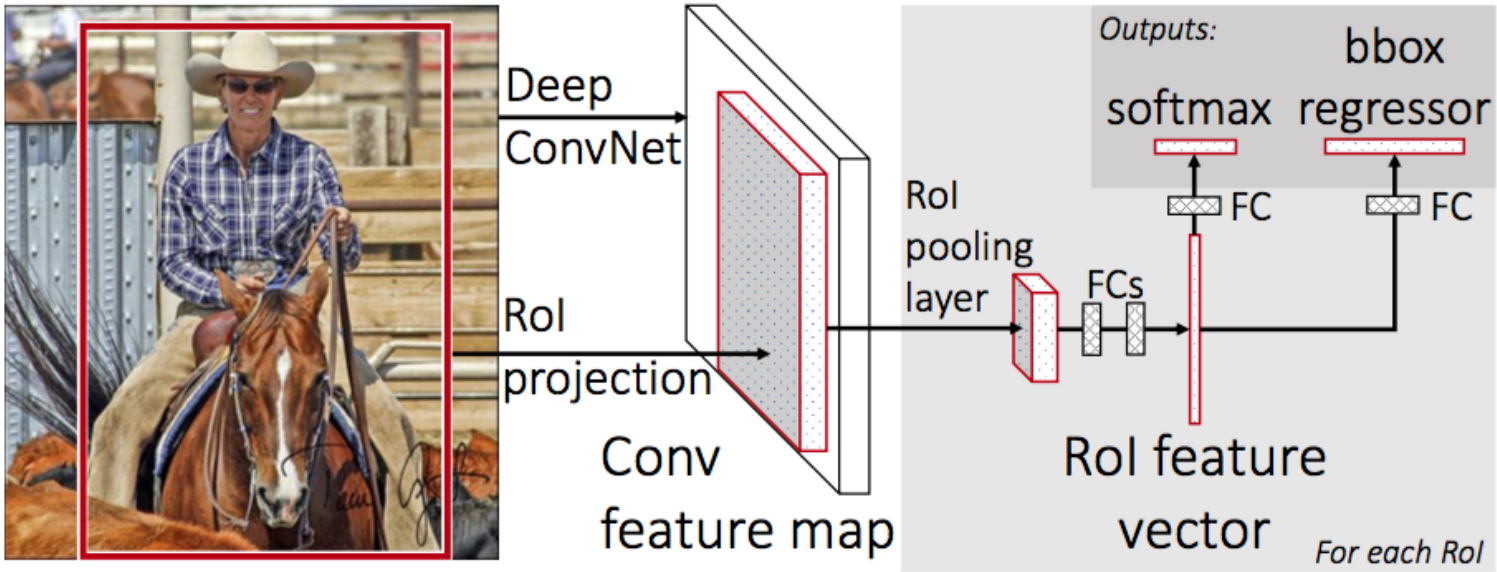
Region CNN - Results

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT $pool_5$	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc_7 BB	**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**	**64.8**	**58.5**
DPM v5 [17]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [25]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [27]	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

* FT, finetunning
* BB, Bounding Box regression

Fast R-CNN

- A **unique** CNN forward for the same image
- Regions re-use this same forward



Fast R-CNN

- Pre-trained VGG16 is used as CNN model
- Regions of interest (ROI) are projected up to the last CNN map

Fast R-CNN

- Last maxpool is removed
- A ROI pooling layer is then apply to obtain a **fixed** feature vector length (7x7x512)
- To this end the projected roi is split into a grid of sub-windows
- Maxpooling is apply to each sub-window

Fast R-CNN - Training

- ROIs of images are properly presented to save computations
- Two output layers:
  - Softmax over $K + 1$ classes
  - Bonding box regression (scale-invariant translation and log-space height/width shift):

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

- Data augmentation to achieve scale-invariant object detection

Fast R-CNN - Test

- A unique CNN forward through Conv+MaxPool
- Thousands of forward through FC
- SVD compression to speedup FC forward

Fast R-CNN - Results

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	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
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	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**

VOC 2007

Fast R-CNN - Results

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

VOC 2010

Fast R-CNN - Results

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	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
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	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**

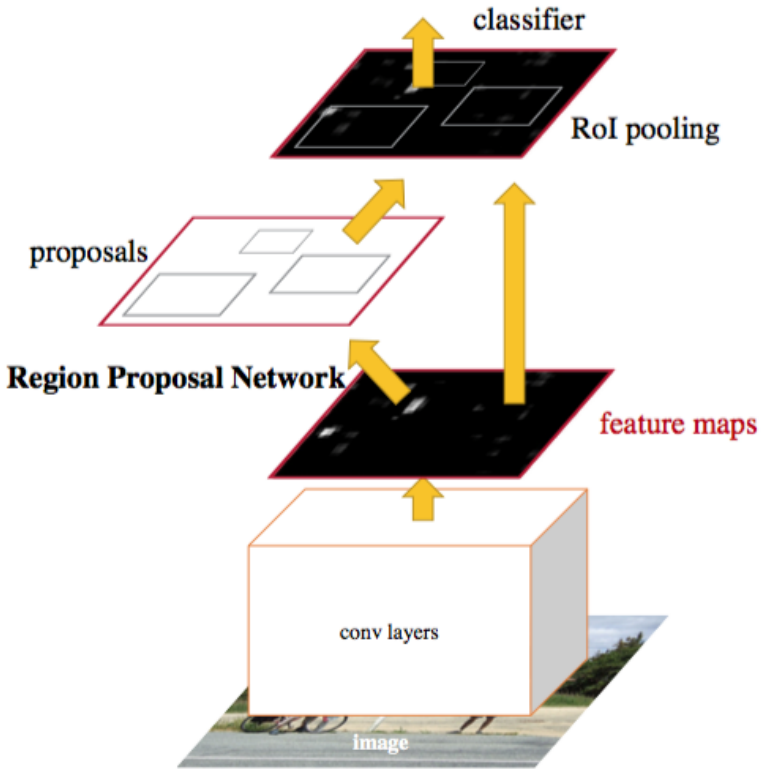
VOC 2012

Fast R-CNN - Results

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Faster R-CNN

- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- An unified network



Faster R-CNN

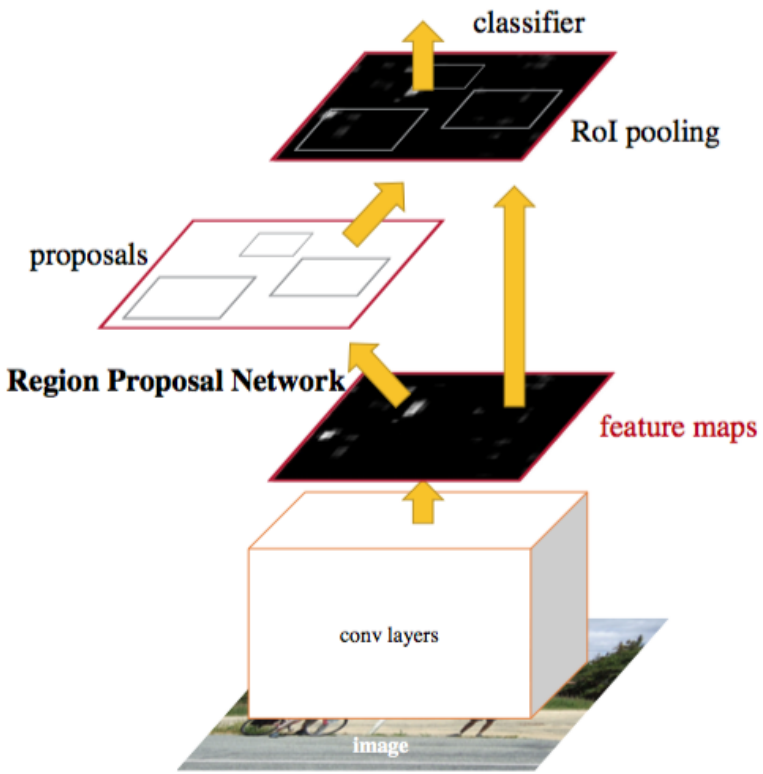
- Region proposal methods typically rely on inexpensive features and economical inference schemes
- Region proposals are the test-time computational bottleneck in state-of-the-art detection systems:
  - Selective Search takes 2 seconds per image in a CPU implementation
  - EdgeBoxes provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image
- Region proposal step still consumes as much running time as the detection network

Faster R-CNN

- Idea: the convolutional feature maps used by region-based detectors, like Fast R- CNN, can also be used for generating region proposals
- Solution: Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks
- The RPN is thus a kind of fully convolutional network (FCN)
- The RPN can be trained end-to-end specifically for the task for generating detection proposals
- Finally RPN and Fast R-CNN are combined

Faster R-CNN

- Faster R-CNN two modules:
  1. A deep fully convolutional network that proposes regions (RPN)
  2. A Fast-RCNN over the proposed regions
- An unified framework where the RPN module tells the Fast R-CNN module where to look

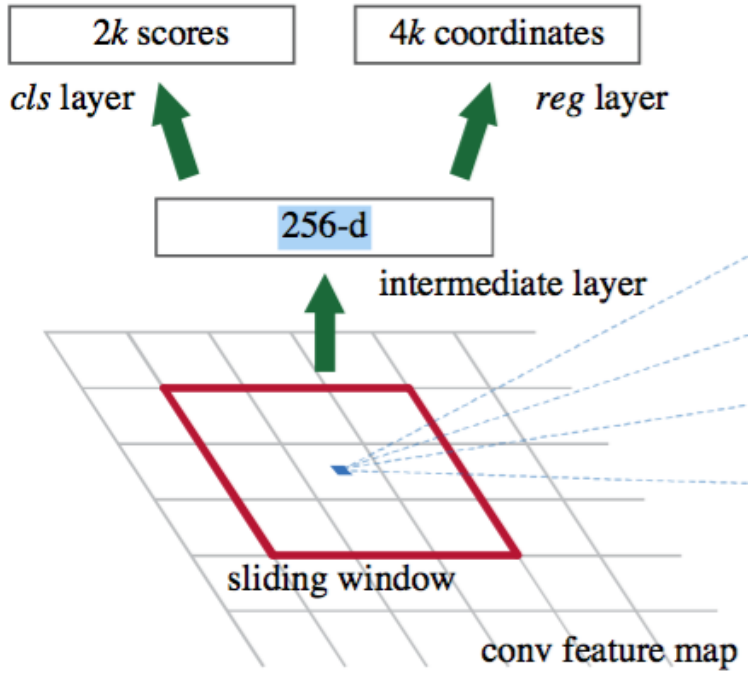


Faster R-CNN

- The RPN Network:
  - takes an image as input and outputs a set of rectangular object proposals
  - an objectness score also is provided
  - shared conv layers with the Faster R-CNN
- How it works:
  - Sliding a small network over the convolutional feature map output
  - $n \times n$ sliding window
  - Each sliding window is mapped to a lower-dimensional feature (256 or 512)
  - This feature is fed into two FC, a box-regression layer and a box-classification layer

Faster R-CNN

- Sliding window 3×3
- FC layers



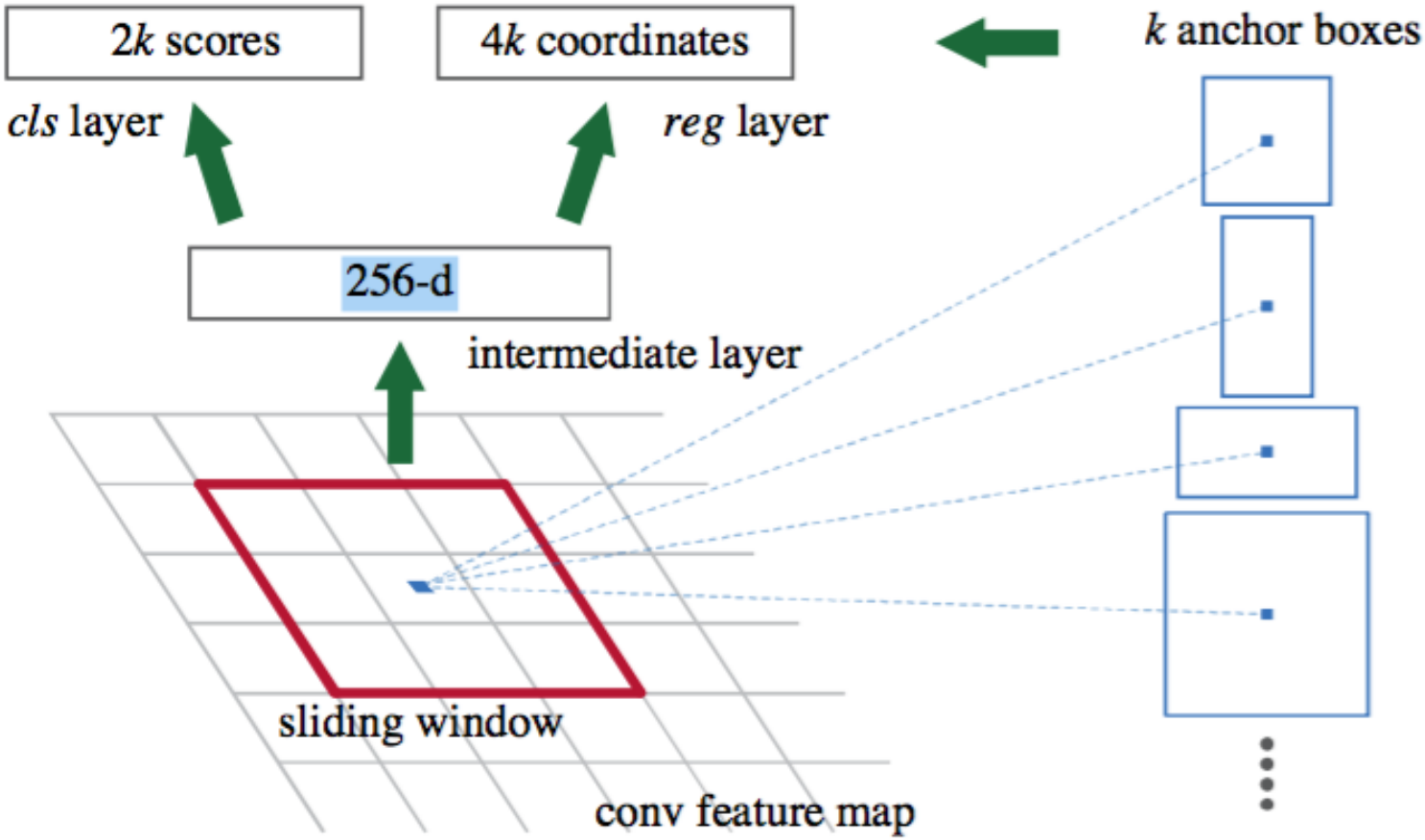
- For simplicity the authors implement the cls layer as a two-class softmax layer. Alternatively, one may use logistic regression to produce k scores

Faster R-CNN

- The RPN is thus a kind of fully convolutional network (FCN):
  - The FC layers are shared among all the positions
  - A $n \times n$ convolutional layer
  - Two 1×1 convolutional layers
- At each sliding-window location, multiple region proposals are predicted
- k different region proposals, *anchors*

Faster R-CNN

- For each region proposal two different targets
  - Box regression, anchor fitting
  - Objectness score



Faster R-CNN - Training

- Training RPN:
  - Objectness loss
  - Regression box loss (only for positive samples)
- Training both, RPN and Fast R-CNN:
  - Alternating training (iterative alternating process)
  - Approximate joint training (joint RPN and Fast R-CNN)
  - Non-approximate joint training (using a RoI pooling layer that is differentiable)
  - Step Alternating Training. (not share (Both) - share cnn (RPN) - share cnn (Faster R-CNN))

Faster R-CNN - Results

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**