ML Engineer 과정

Object Detection

Gunhee Kim

Computer Science and Engineering

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

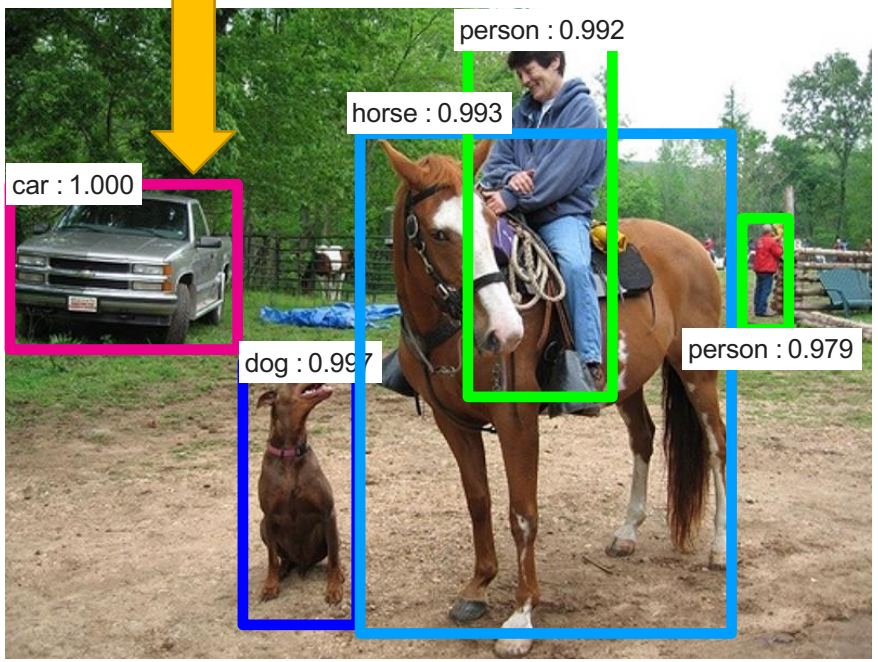
Outline

- Two-stage Models
  - R-CNN (CVPR'14), SPPNet (ECCV'14), Fast R-CNN (ICCV'15), Faster R-CNN (NIPS'15)
- Single-stage Models
- Application: OCR

Object Detection

Localization
Where?

Recognition
What?



Slide Credit: Kaiming He, ICCV 2015 Tutorial

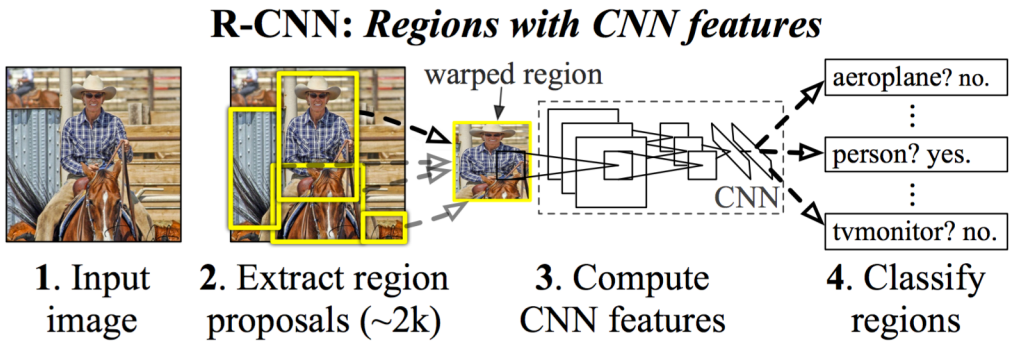
Key Concepts to Remember

Convolutional feature maps

Spatial pooling

RoI (Regions of Interest) pooling

Bounding box regression



R-CNN

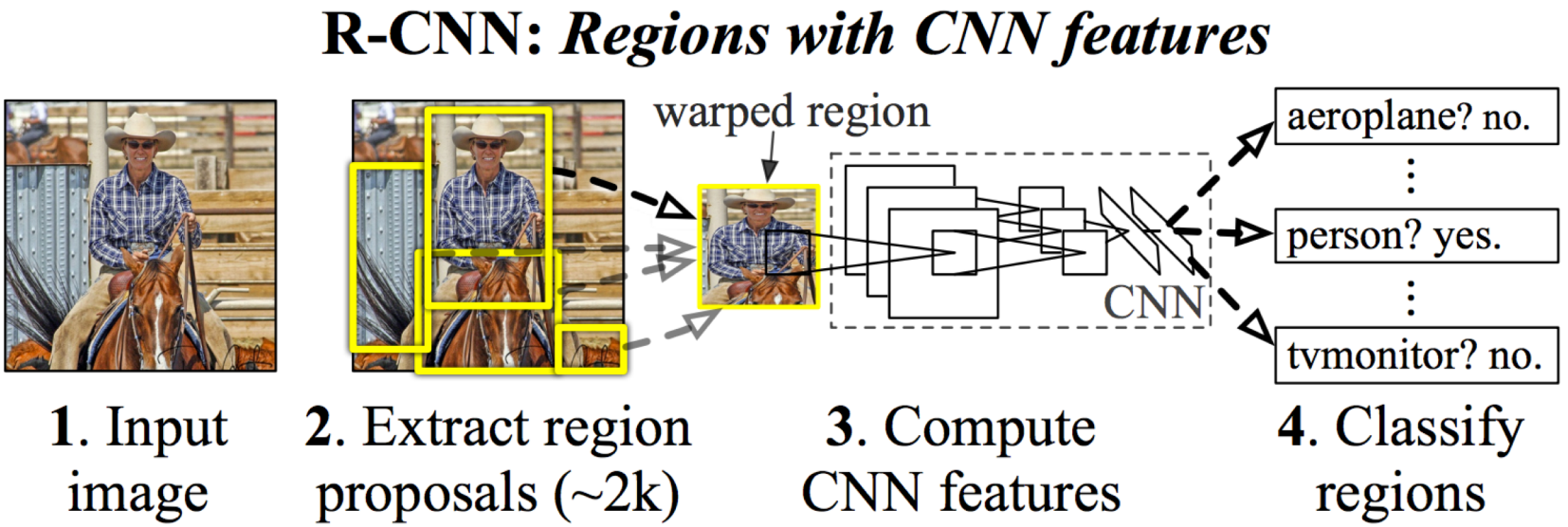
Rich feature hierarchies for accurate object detection and semantic segmentation (CVPR'14)

Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik

https://github.com/rbgirshick/rcnn

R-CNN: We already know this!

For each region proposal (from SS or EdgeBox), compute the CNN feature of that region and classify



Code is at https://github.com/rbgirshick/rcnn

R-CNN: Slow and Naive

Depends on the external region proposal algorithm

Due to image cropping/warping, image resolution can be lost

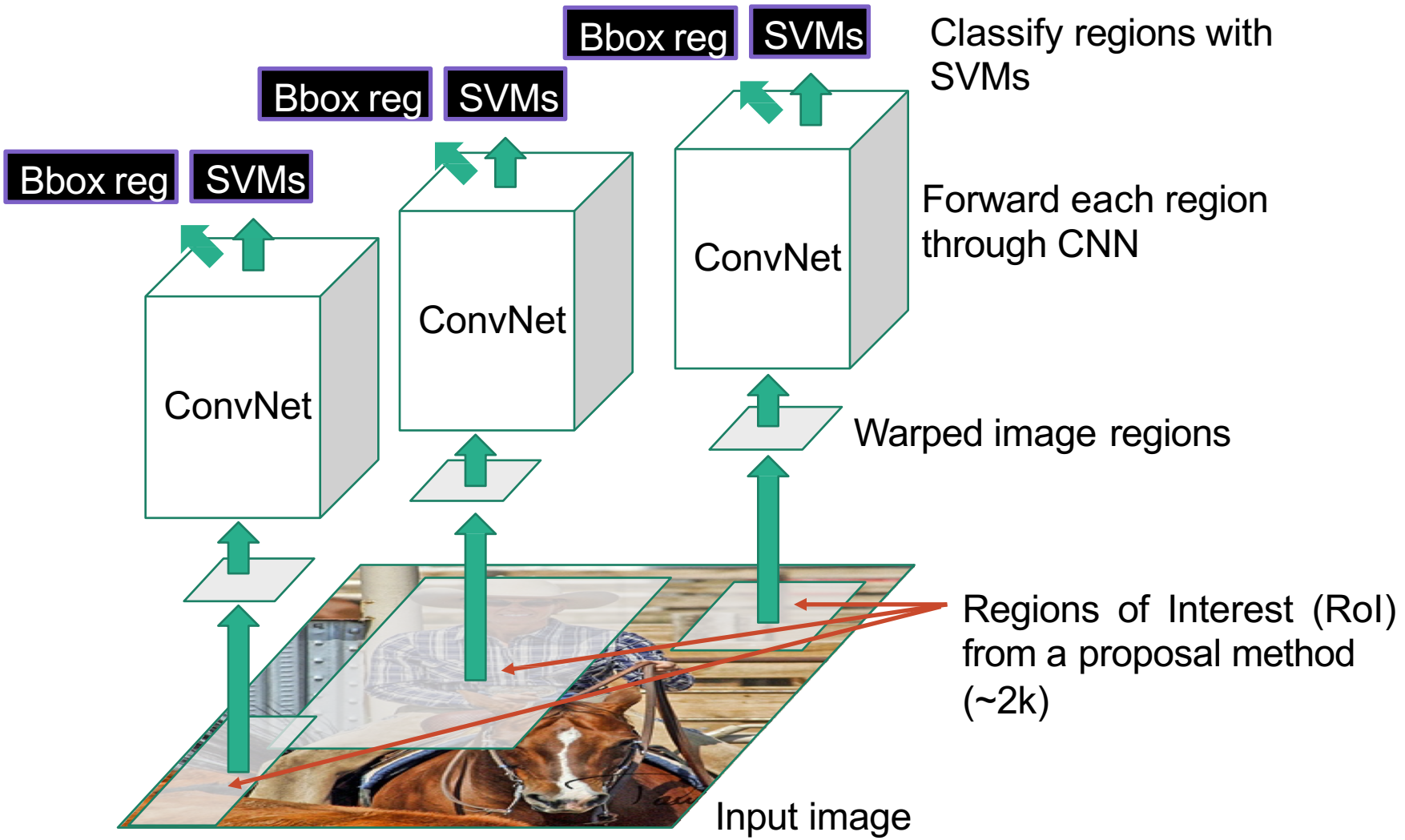
Need to feed-forward multiple times (i.e. proportional to the number of proposals, e.g. ~2K)

- Inference is very slow, about ~10sec / image

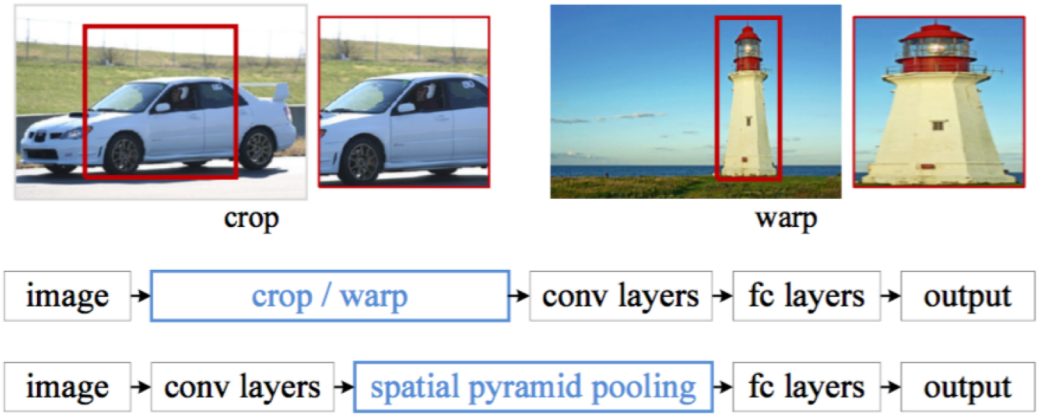
External region proposals are slow, CPU only, and require a complex pipeline (no end-to-end)

- Feature caching requires a lot of disk…
- Training is very slow (~84 hrs)

R-CNN: Slow and Naive



Slide Credit: Ross Girshick, caffe-cvpr-15 tutorial



SPPNET

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition (ECCV'14)

Kaiming, He and Xiangyu, Zhang and Shaoqing, Ren and Jian Sun

https://github.com/ShaoqingRen/SPP_net

Key Problems of Previous Approaches…

1. CNN requires a fixed input image size (e.g. 224x224)
   - Limit both the aspect ratio and the scale of input image
   - Cropping: may not contain the entire object
   - Warping: unwanted geometric distortion

2. R-CNN is time-consuming: repeatedly apply CNNs to each of thousands ROIs

Key Ideas of SPPNet

Use spatial pyramid pooling in the context of CNNs

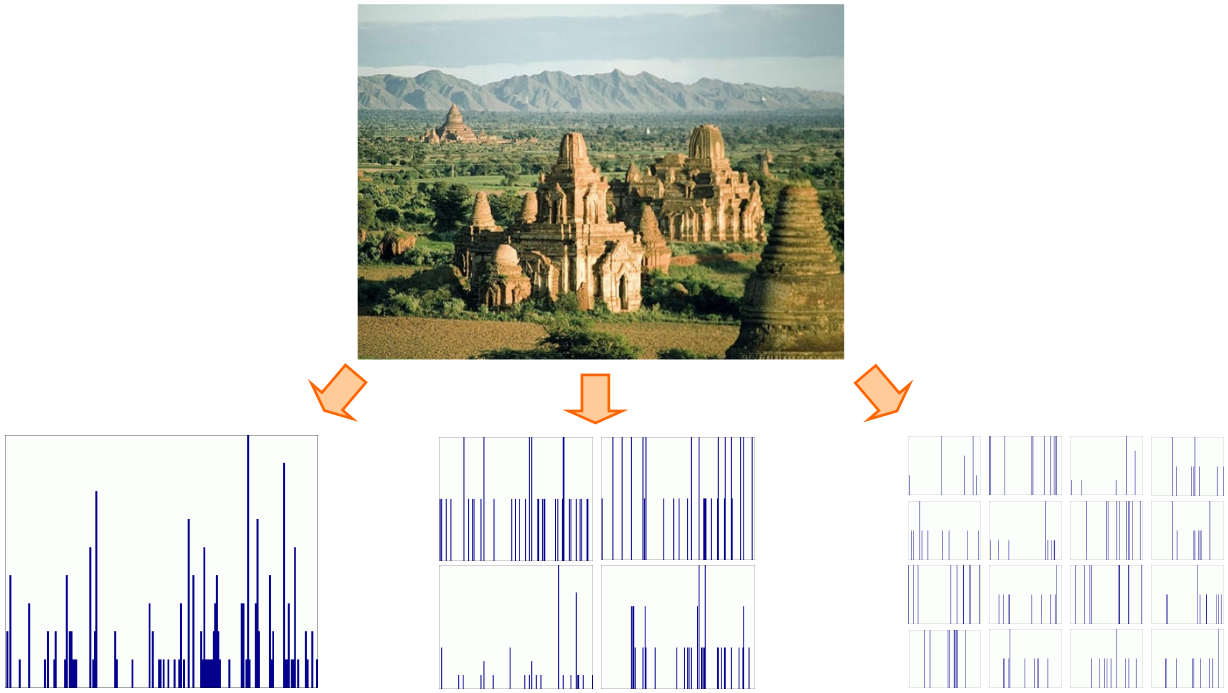
1. Handle arbitrary sized input and multi-resolution images
   - Later FC layer requires fixed-size input
   - SPP can generate a fixed-length output regardless of input sizes
2. Use a **single pass** of CNN on the entire image

Spatial Pyramid

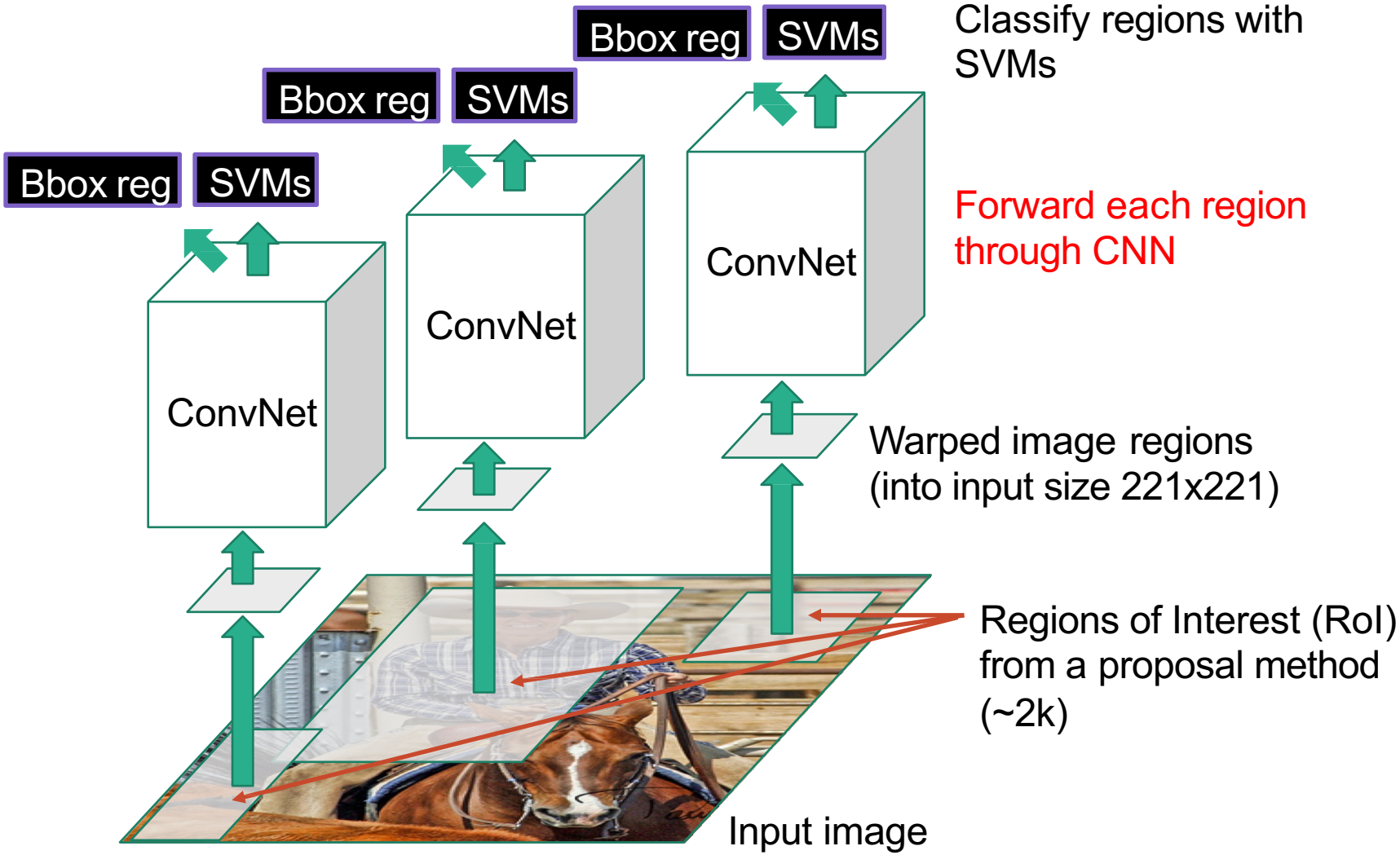
Compute histogram in each spatial bin

- Extension of a bag of features

Locally orderless representation at several levels of resolution

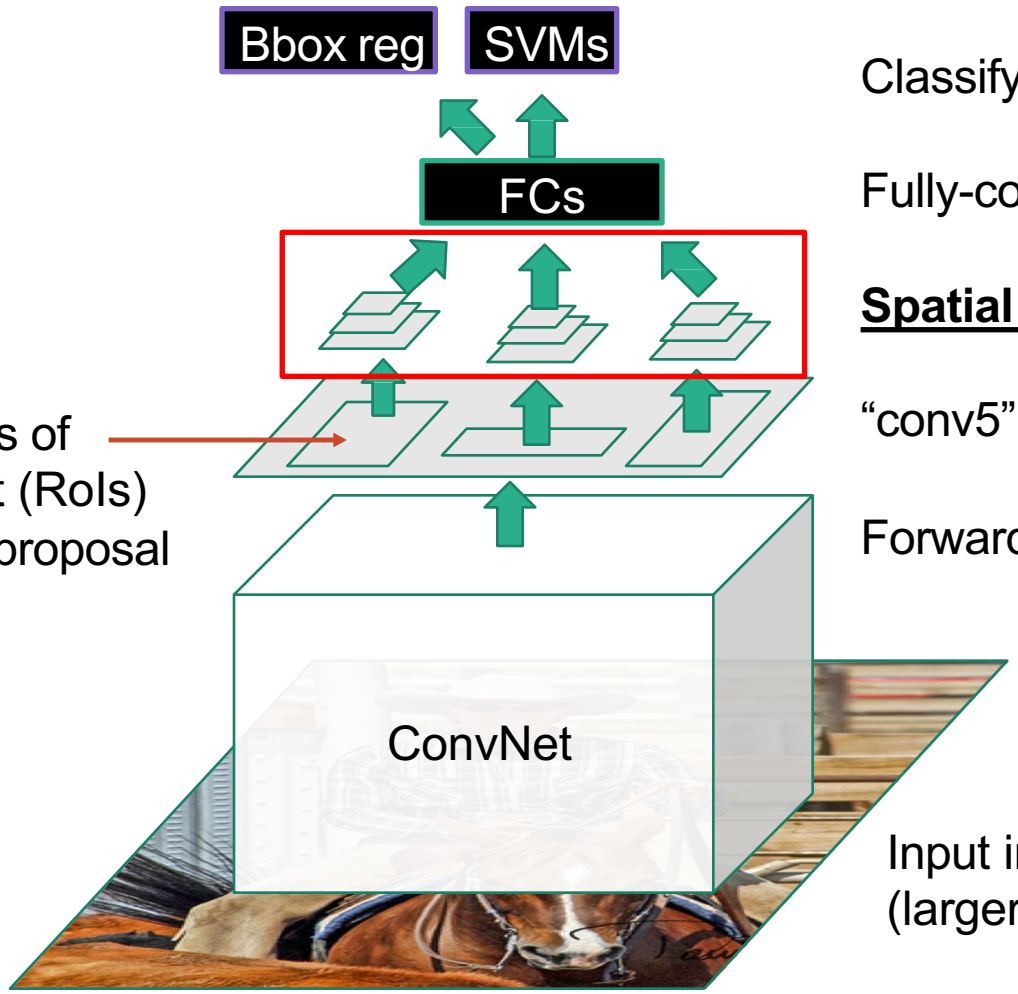


R-CNN



Slide Credit: Ross Girshick, caffe-cvpr-15 tutorial

SPPNet



Apply bounding-box regressors

Classify regions with SVMs

Fully-connected layers

Spatial Pyramid Pooling (SPP) layer

"conv5" feature map of image

Forward whole image through CNN

Input image
(larger than 221x221)

Slide Credit: Ross Girshick, caffe-cvpr-15 tutorial

Spatial Pyramid Pooling Layer

Replace last pooling layer with a SPP layer

- Orthogonal to any CNN designs

Input: conv feature map (e.g. conv5_3 in VGG)

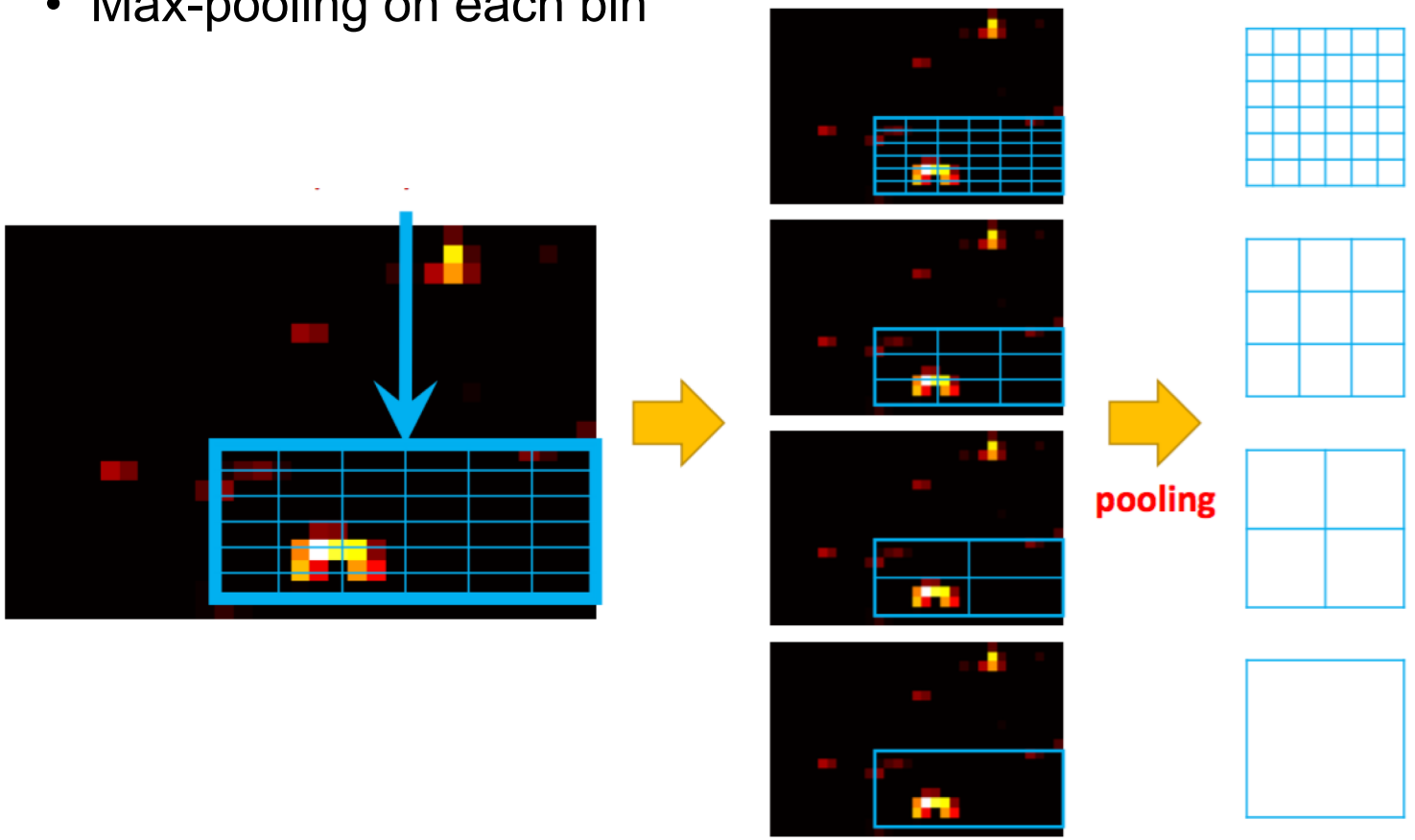
Output: corresponding feature map for a specific region of interest (RoI)

Slide Credit: Kaiming He, ICCV 2015 Tutorial

Spatial Pyramid Pooling Layer

Build a pyramid of feature pools with fixed levels and sizes

- 4-level SP (1x1, 2x2, 3x3, 6x6, totally 50 bins)
- Max-pooling on each bin



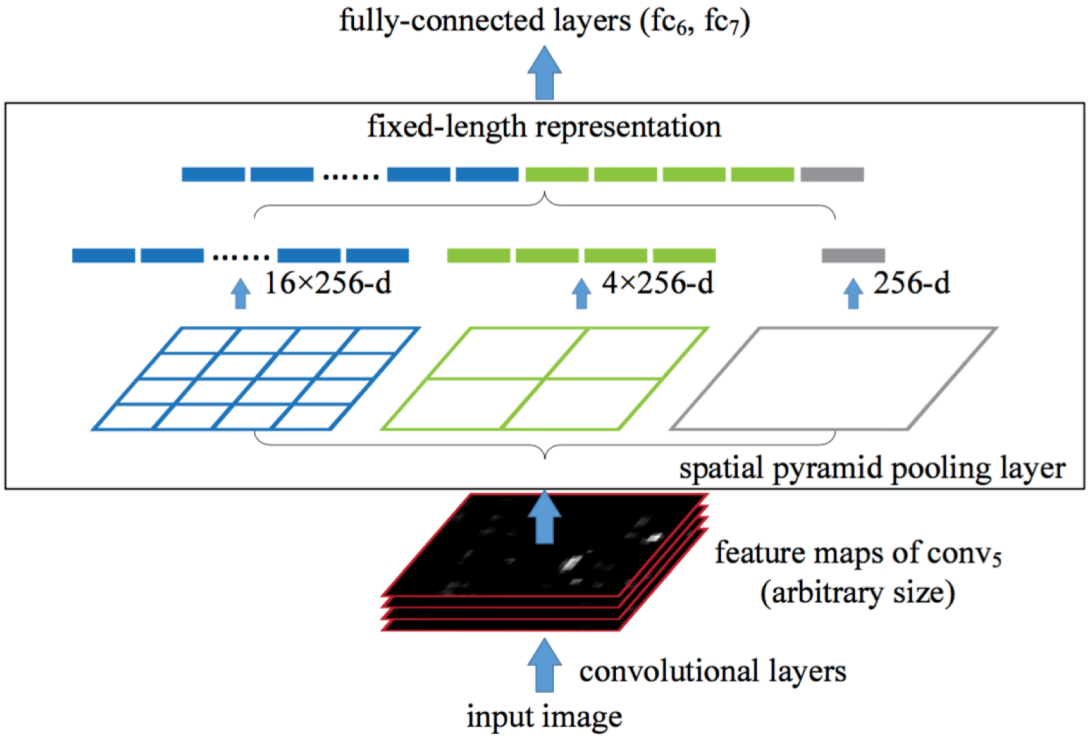
Slide Credit: Kaiming He, ICCV 2015 Tutorial

Spatial Pyramid Pooling Layer

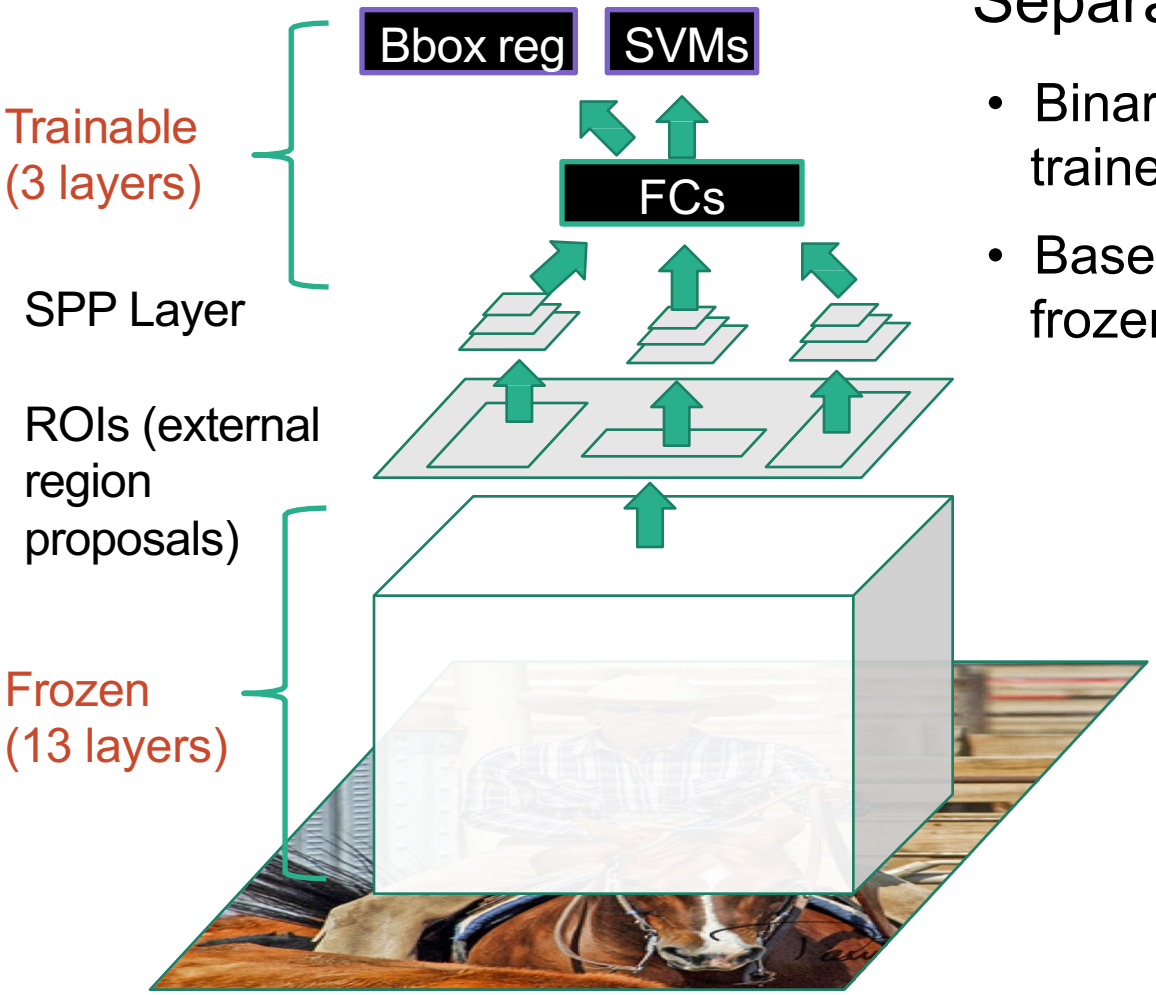
For each pyramid layer, max-pool the responses of each filter, and concatenate them to yield a fixed-length vector

The dimension : (# of conv filters) × (# of total bins)

- e.g. 256 × (1^2 + 2^2 + 3^2 + 6^2) = 256 × 50



SPPNet



Separate fine-tuning

- Binary SVM classifiers are trained on the features
- Base CNN are trained and frozen

Slide Credit: Ross Girshick, caffe-cvpr-15 tutorial

Detection Implementation

Fast mode of selective search

- Generate about 1K candidate windows per image
- Resize the image such that min dim = {480,576,688,864,1200}
- Run CNNs (e.g. ZFNet or VGGNet)

Multi-scale feature extraction

- Choose a single scale such that the scaled window has similar # of pixels to 224x224

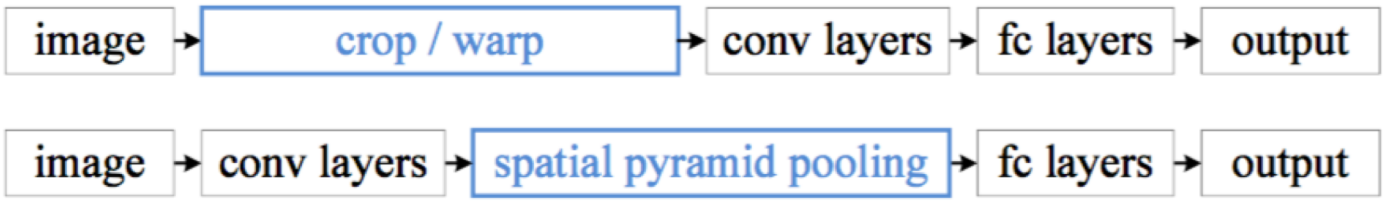
SPPNet: Pros and Cons

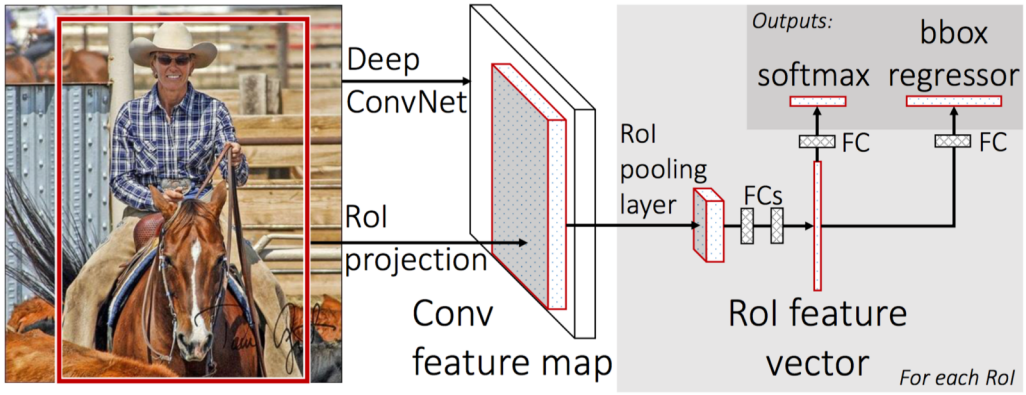
The good

- There is only one CNN feed-forward pass per image; make inference (detection) orders-of-magnitude fast
- FC-layers and output layers are applied regionwise; can take arbitrary sized input

The bad

- SPP Pooling is done in a separate pipeline; training is still slow (like R-CNN)
- Gradient is blocked at the SPP layer; parameters below below SPP layer can't be fine-tuned





FAST R-CNN

Fast R-CNN (ICCV 2015)

Ross Girshick

https://github.com/rbgirshick/fast-rcnn

Fast R-CNN

Share many features with SPPNet

Key Contribution: The RoI Pooling Layer

- A (simplified) single-level SPP layer
- Makes the whole network trainable in a single stage!

Better mAP and fast training/detection!

- vs. R-CNN: 9x faster in training and 213x faster in test
- vs. SPPNet, 3x faster in training and 10x faster in test

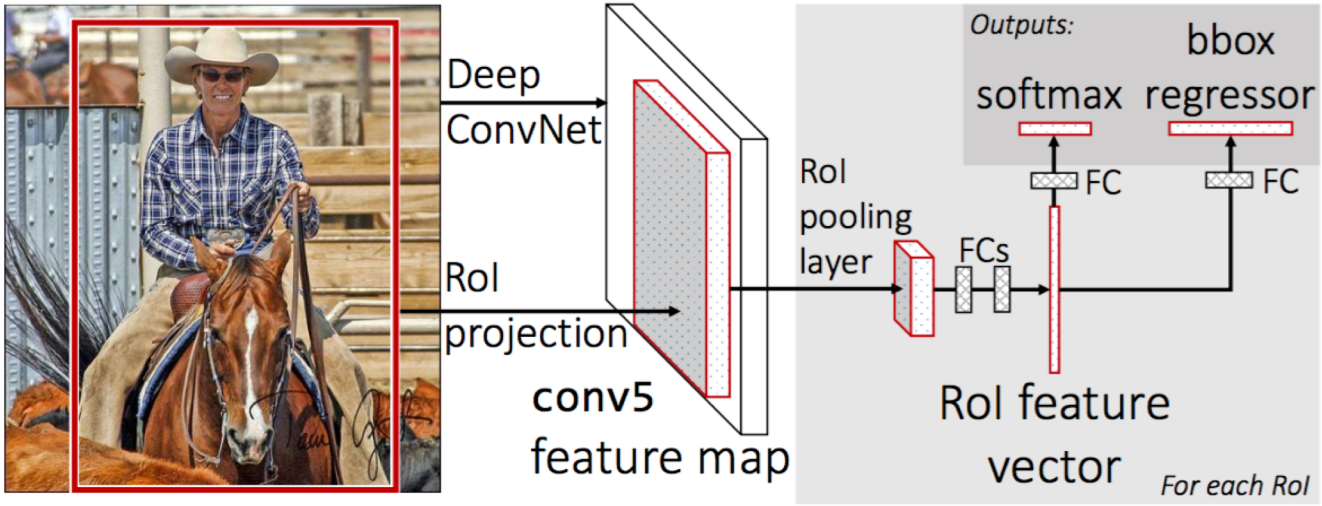
Fast R-CNN Architecture

Input image and multiple RoIs are input into Deep ConvNet

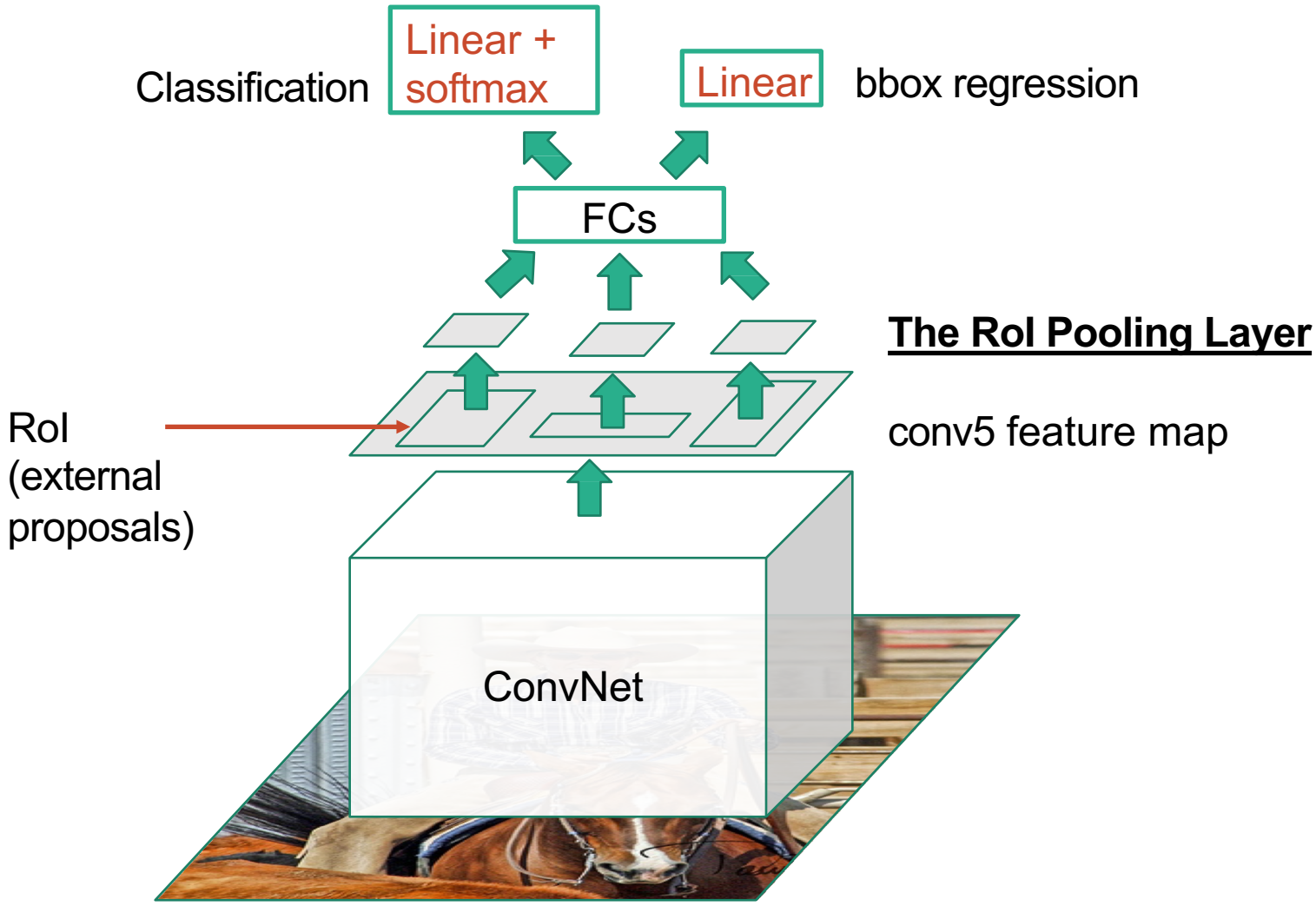
- Each ROI is pooled into a fixed size feature vector

Two output vectors per RoI

- Softmax probability over K+1 classes
- BB coordinates [r,c,h,w] (top-left corner + height/width) for each of K classes



Fast R-CNN



Slide Credit: Ross Girshick, caffe-cvpr-15 tutorial

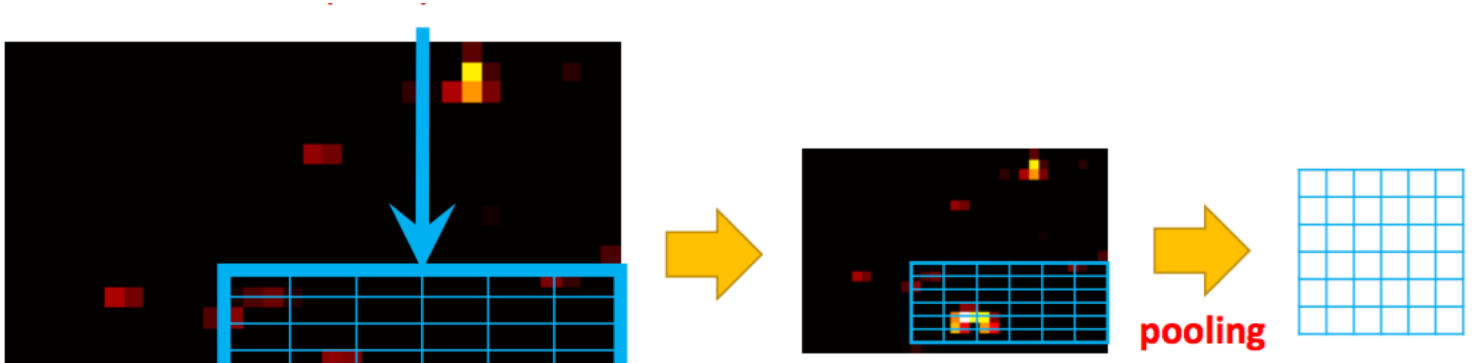
The RoI Pooling Layer

RoI Pooling Layer = SPP layer with 1 pyramid level

- Feature map from deep convnets (e.g. VGG16) perform well **even at a single scale**

Layer input and output

- (Shared) conv5 feature map: [H x W x 512] and R RoIs
- Output: [R x H' x W' x 512] (e.g. H' = W' = 7)



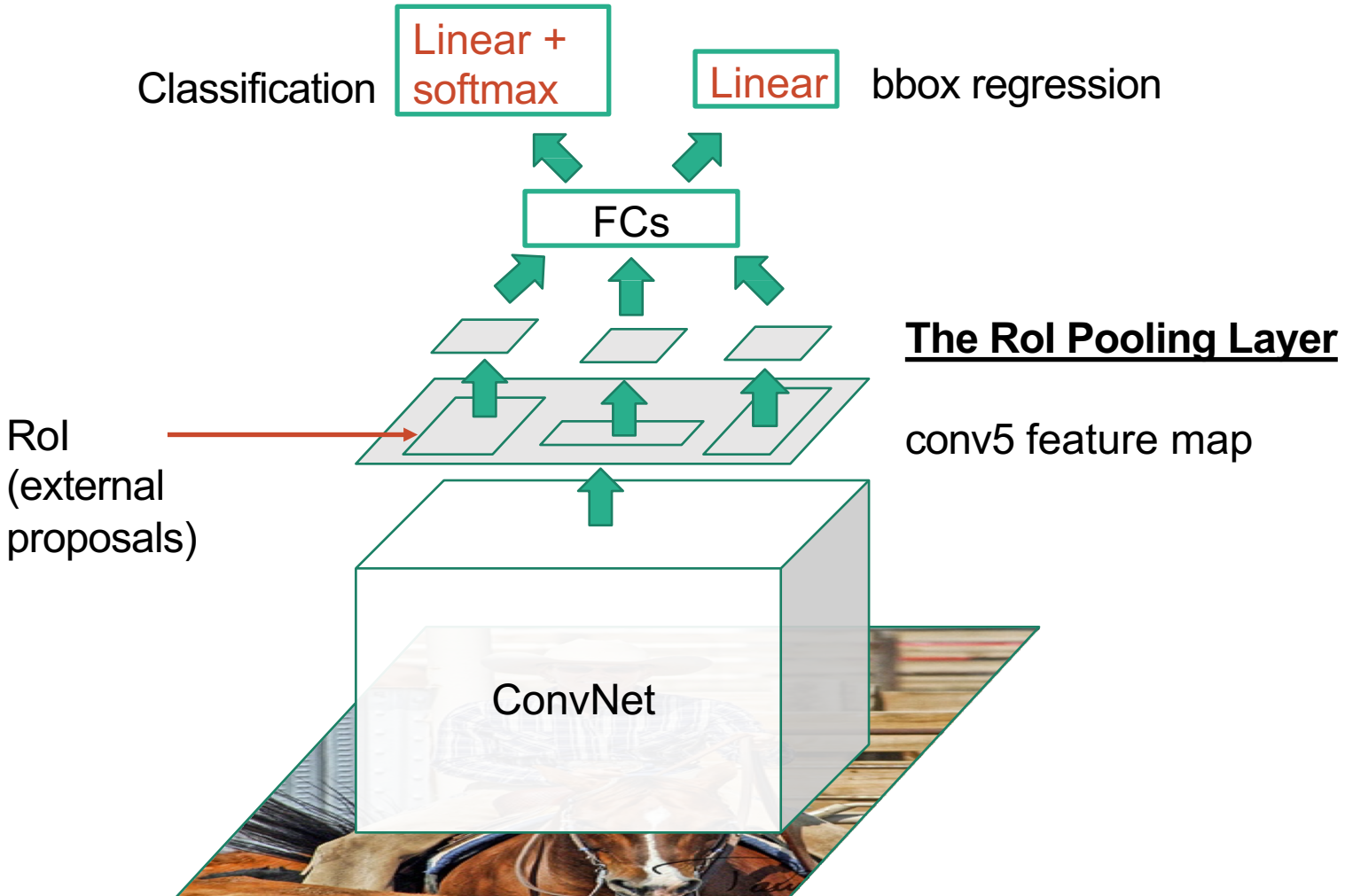
	SPP-net 1-scale	SPP-net 5-scale
$pool_5$	43.0	44.9

Slide Credit: Kaiming He, ICCV 2015 Tutorial

Fast R-CNN: Training

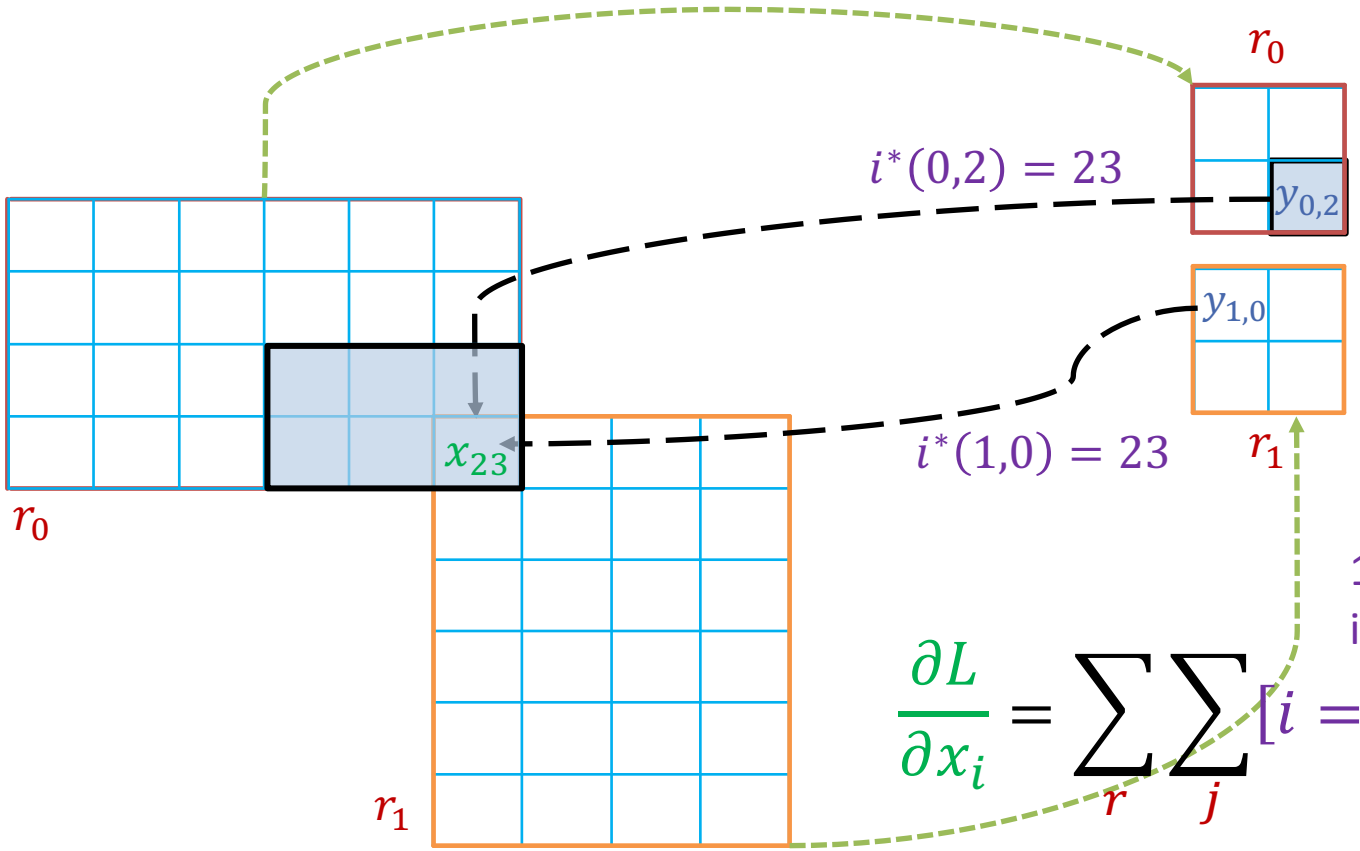
Network could be trained in a single training stage

- The RoI pooling layer operation should be differentiable



1. Backpropagation through RoI Pooling

Almost same as max-pooling (but regions may overlap)



1 if r, j "pooled" input i; 0 o/w

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

Partial for x_i — Over regions r, locations j — Partial from next layer

Slide Credit: Ross Girshick, ICCV 2015 Tutorial

2. Multi-task Loss

Multi-task loss: classification and bbox regression (per ROI)

- The RoI pooling layer operation should be differentiable

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

- $L_{cls}(p, u) = -\log p_u$: classification cross-entropy loss
- $L_{loc}(t, t^*)$: sum of smooth-L1 loss for regression
- u: true class, p: class probability
- t^u: bbox of true class v: predicted bbox,

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

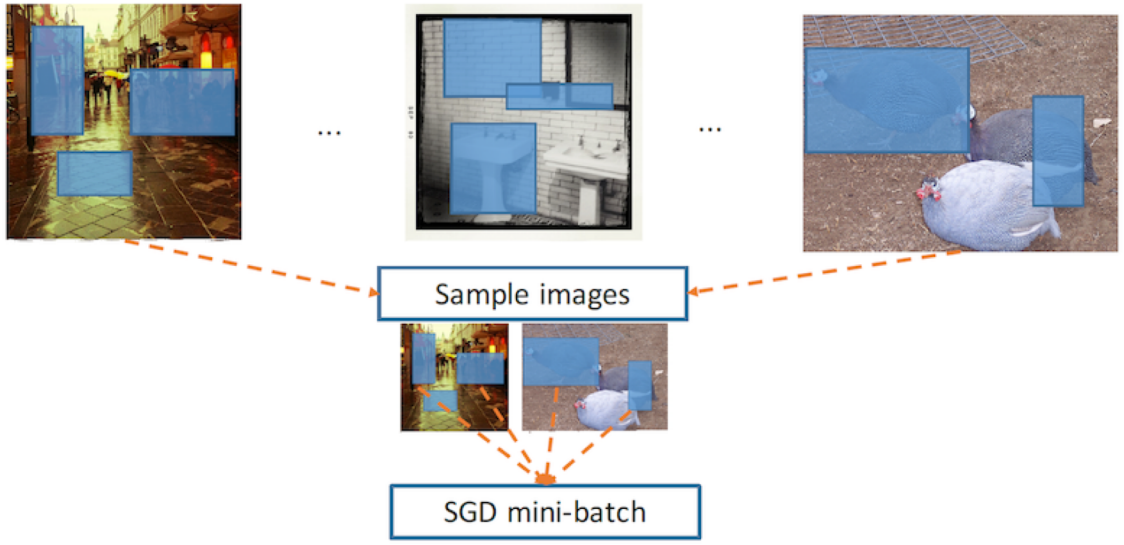
3. Hierarchical Sampling to Build mini-batch

If a mini-batch includes RoIs from many different training images, SGD steps are inefficient

- e.g. What if we sample 128 RoIs uniformly at random?

Hierarchical sampling

- Sample a small number of images (e.g. N=2)
- Then sample many examples from each image (e.g. R/N = 64)



3. Hierarchical Sampling to Build mini-batch

Pros: Share computation between overlapping examples from the same image

Cons: Examples from the sample image may be too correlated

- Not seriously problematic in practice, but significantly reduce computation time

Better training time and testing time with better accuracy

- R-CNN / SPPNet / Fast R-CNN
- Training time: 84 hours / 25.5 hours / 8.75 hours
- VOC07 test mAP: 66.0% / 63.1% / 68.1%
- Test time per image: 47s / 2.3s / 0.32s

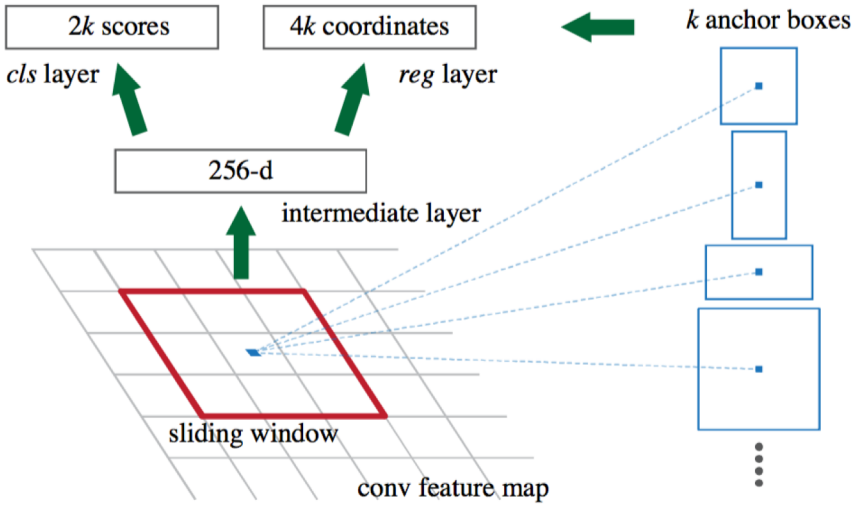
Summary of Fast R-CNN

The good

- Inference is single-forward, and fast (~160x faster than R-CNN)
- Can handle multi-scale input sizes

The bad

- Still depends on the external region proposals (which is eliminated by Faster R-CNN!)



FASTER R-CNN

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

Shaoqing Ren and Kaiming He and Ross Girshick and Jian Sun

https://github.com/rbgirshick/py-faster-rcnn

Faster R-CNN

Eliminates the only drawback of Fast R-CNN:
region proposals are jointly done in a single network

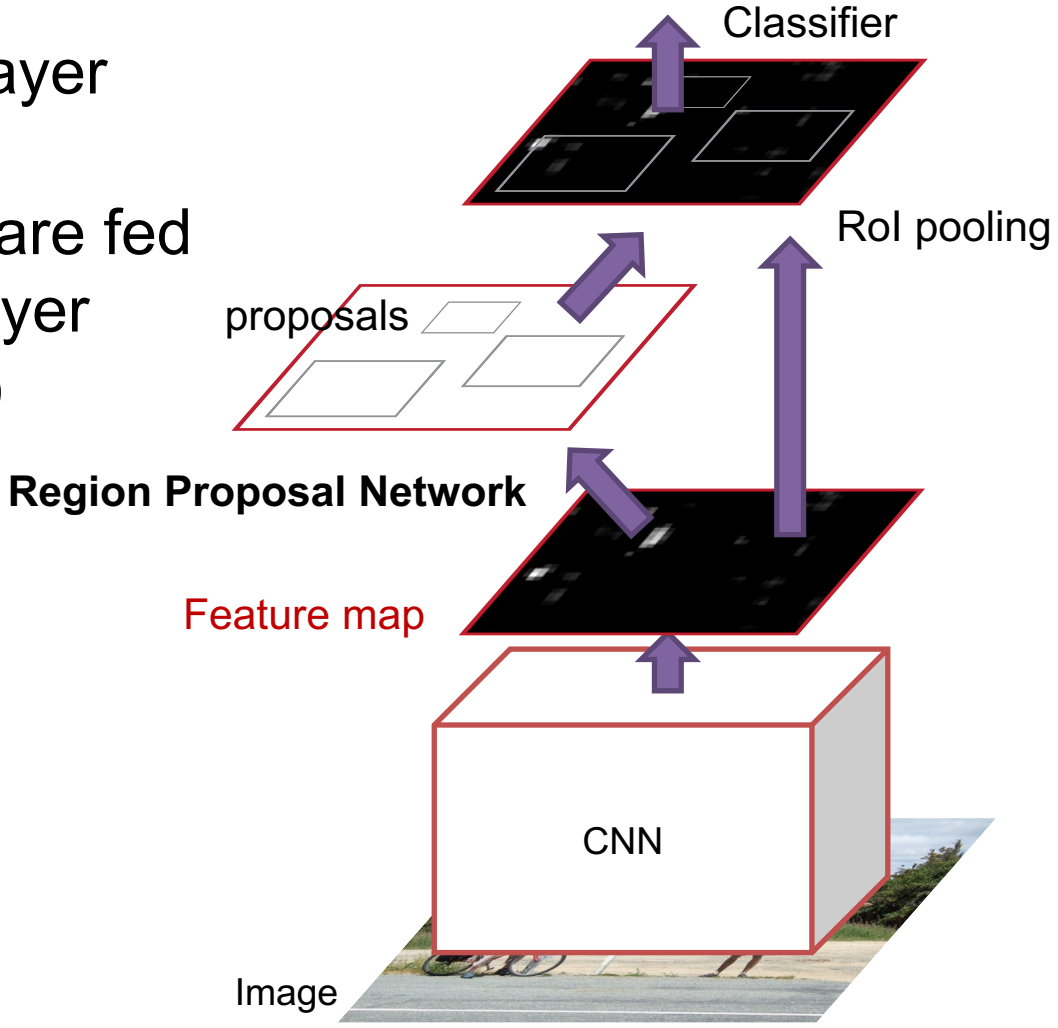
RPN (Region Proposal Network): Let the CNN do region proposals too!!

Faster R-CNN = Fast R-CNN + RPN

Faster R-CNN

Insert a RPN after
the last convolutional layer

The proposed regions are fed
into the RoI Pooling Layer
(same as Fast R-CNN)

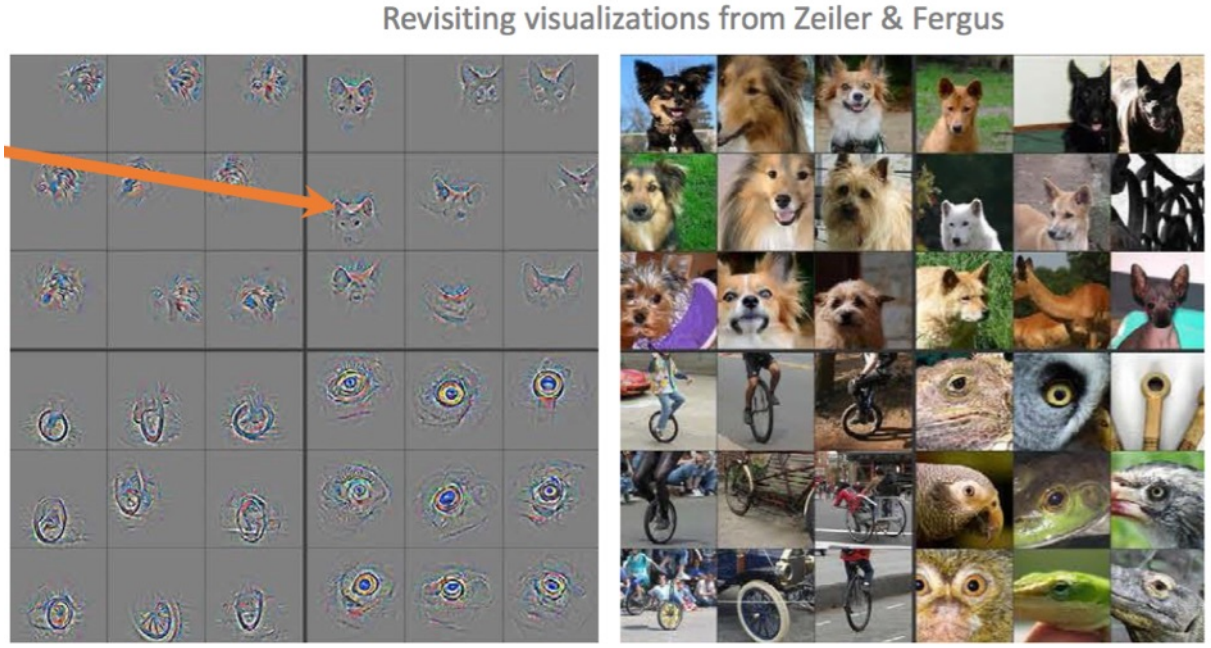


Slide Credit: Ross Girshick, ICCV 2015 Tutorial

Region Proposal from Feature Maps

Intuition from visualizations from (Zeiler & Fergus)

- Convolutional feature map contains an encoded fine-grained localization information!



M. D. Zeiler, and R. Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014.

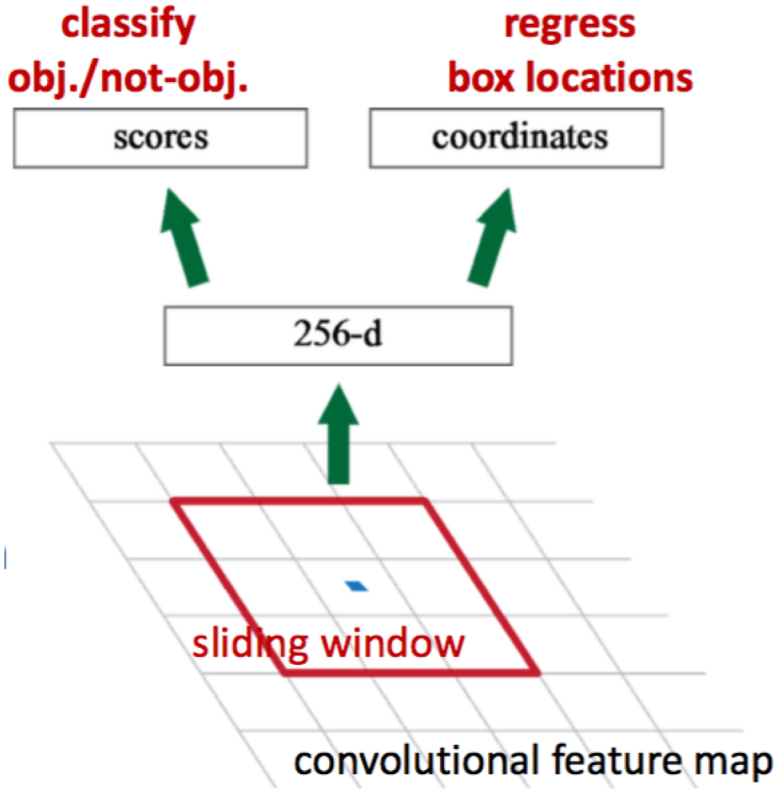
Region Proposal Network

Maintain a sliding-window on the conv feature map, and build a small prediction network for

- Classifying object or not-object
- Bounding-box regressions

Position of sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



Slide Credit: Kaiming He, ICCV 2015 Tutorial

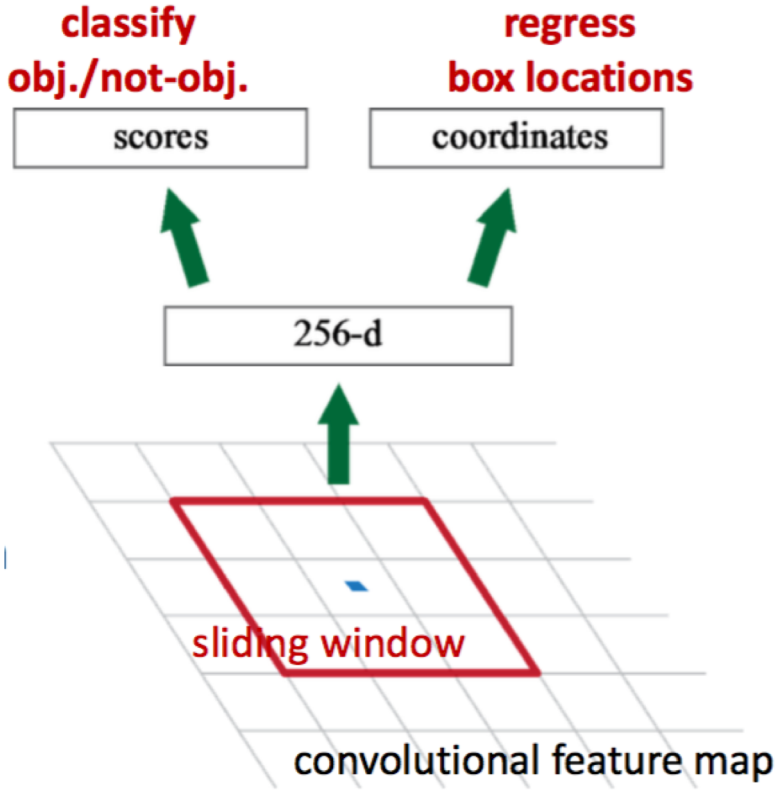
Region Proposal Network

Use ***N*** anchors (pre-defined reference boxes) at each sliding window location

Bounding box regression gives offsets from these anchors

Object scores: regress to 1.0 (positive) or 0.0 (negative)

- Positive: IoU > 0.7
- Negative: IoU < 0.3



Slide Credit: Kaiming He, ICCV 2015 Tutorial

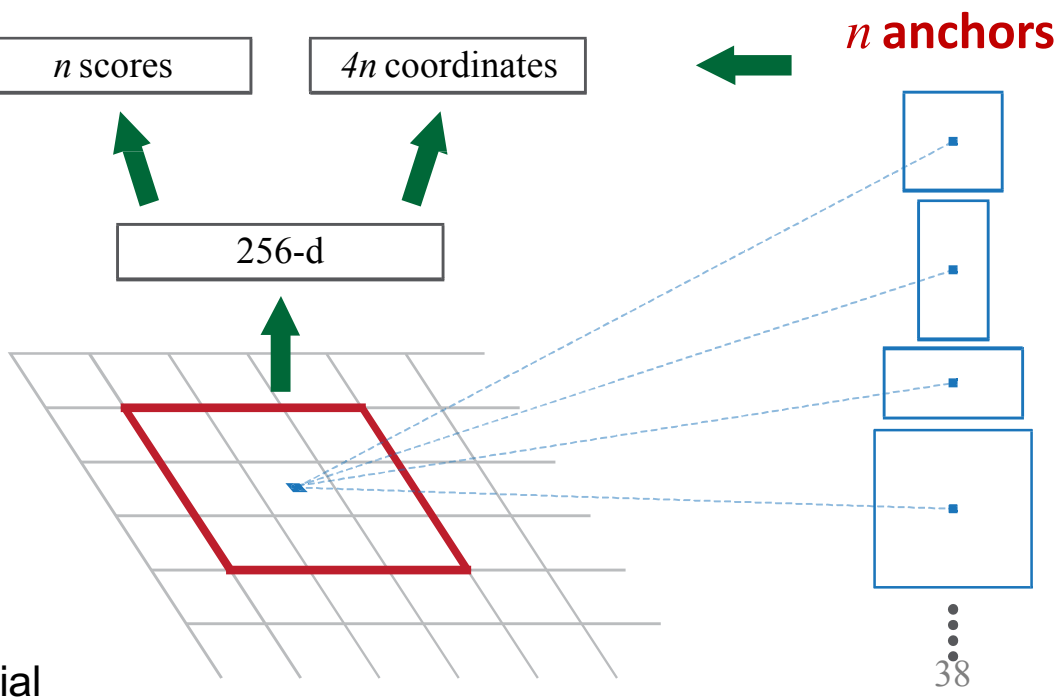
Region Proposal Network

Anchors: multiple anchors are used at each position

- e.g. 3 scales (128, 256, 512) × 3 aspect ratios (2:1, 1:1, 1:2)

Anchors are translation-invariant;
the same set of anchors are used at each window location

The number of region proposals per image:
W × H × n (~2400×9)



Slide Credit: Kaiming He, ICCV 2015 Tutorial

RPN: Training

Two training methods

(#1) Alternating optimization (published in NIPS paper)

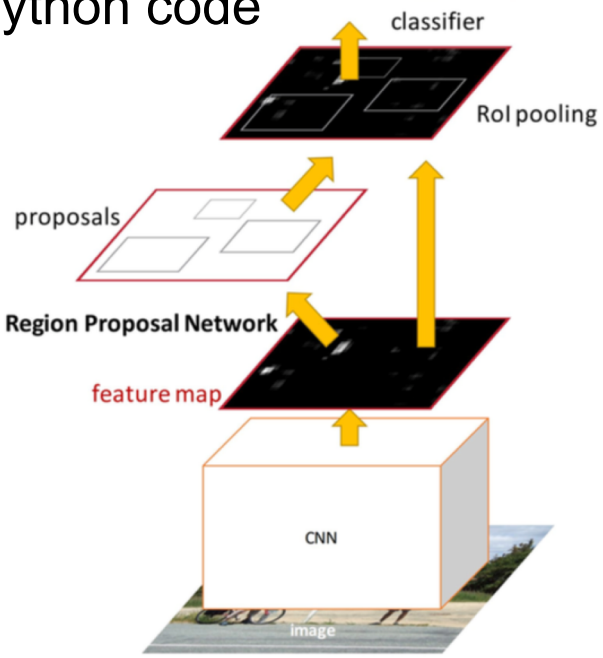
- Train RPN, FRCNN, RPN, FRCNN, and so on … ?!

(#2) Approximate joint optimization (SGD)

- Later updated in PAMI paper and the python code
- Multi-task loss optimization

Goal:

- RPN and Fast-RCNN should share the ‘same convolutional layers’



Slide Credit: Kaiming He, ICCV 2015 Tutorial

Performance!

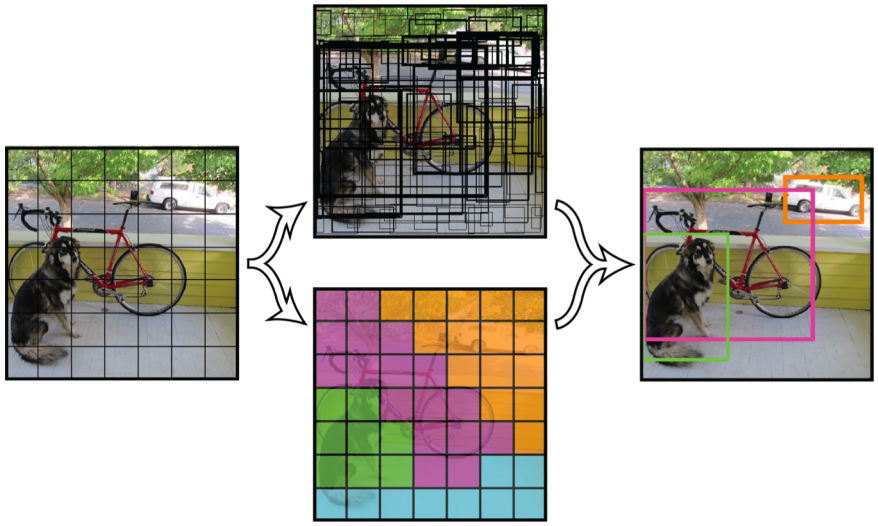
mAP ~ 66.9 (with VOC07) / mAP ~ 73.2 (with VOC07+12)

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Current (2016) state-of-the-art is on the collaboration with RPN (Faster R-CNN) and ResNet-101 !

Outline

- Two-stage Models
  - R-CNN (CVPR'14), SPPNet (ECCV'14), Fast R-CNN (ICCV'15), Faster R-CNN (NIPS'15)
- Single-stage Models
  - YOLO (CVPR' 16), YOLO v2/9000 (CVPR'17), SSD (ECCV' 16), DSSD (ArXiv'17)
- Application: OCR



YOLO

You Only Look Once: Unified, Real-Time Object Detection (CVPR 2016)

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

http://pjreddie.com/darknet/yolo/

YOLO (You Only Look Once)

A single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes

- A full image is processed in one evaluation in an end-to-end manner
- Common post-processing steps such as non-maximum suppression are unnecessary
- Framed as a single regression problem: from image pixels to bbox coordinates and class probabilities

Extremely fast: 45 frames per second on Titan X GPU

- Faster version: 155 frames per second
- Performance is not as good as state-of-the-art detector

YOLO (You Only Look Once)

Divides the input image into a SxS grid (here, S=7)

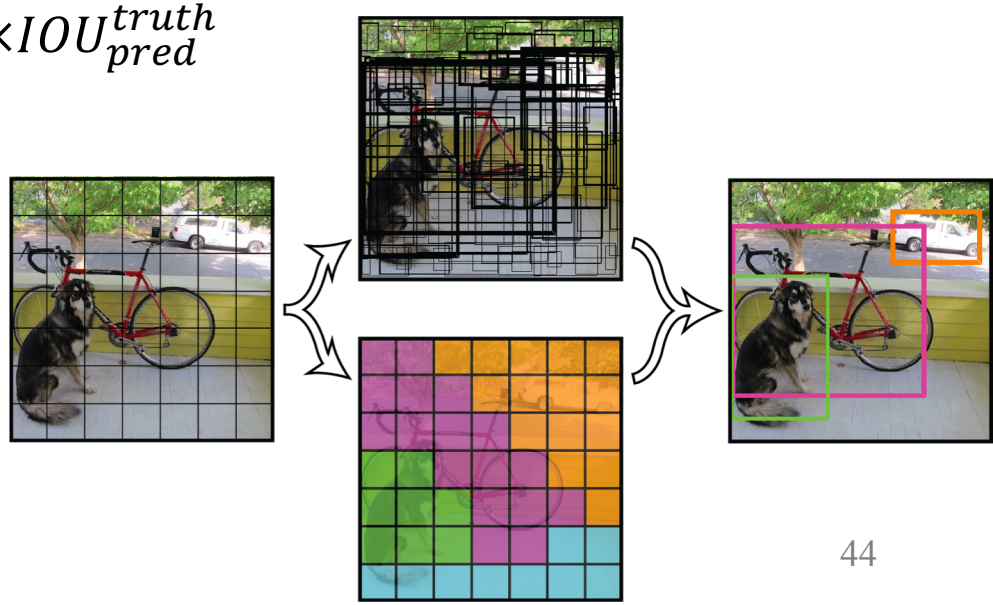
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object

Each grid cell predicts (1) B bboxes, (2) C conditional class probabilities

- Each bbox consists of 5 predictions: x, y, w, h, and confidence
- Confidence = $P(Object) \times IOU_{pred}^{truth}$
- CCP = $P(Class_i | Object)$

Prediction is encoded a SxSx(Bx5+C) tensor

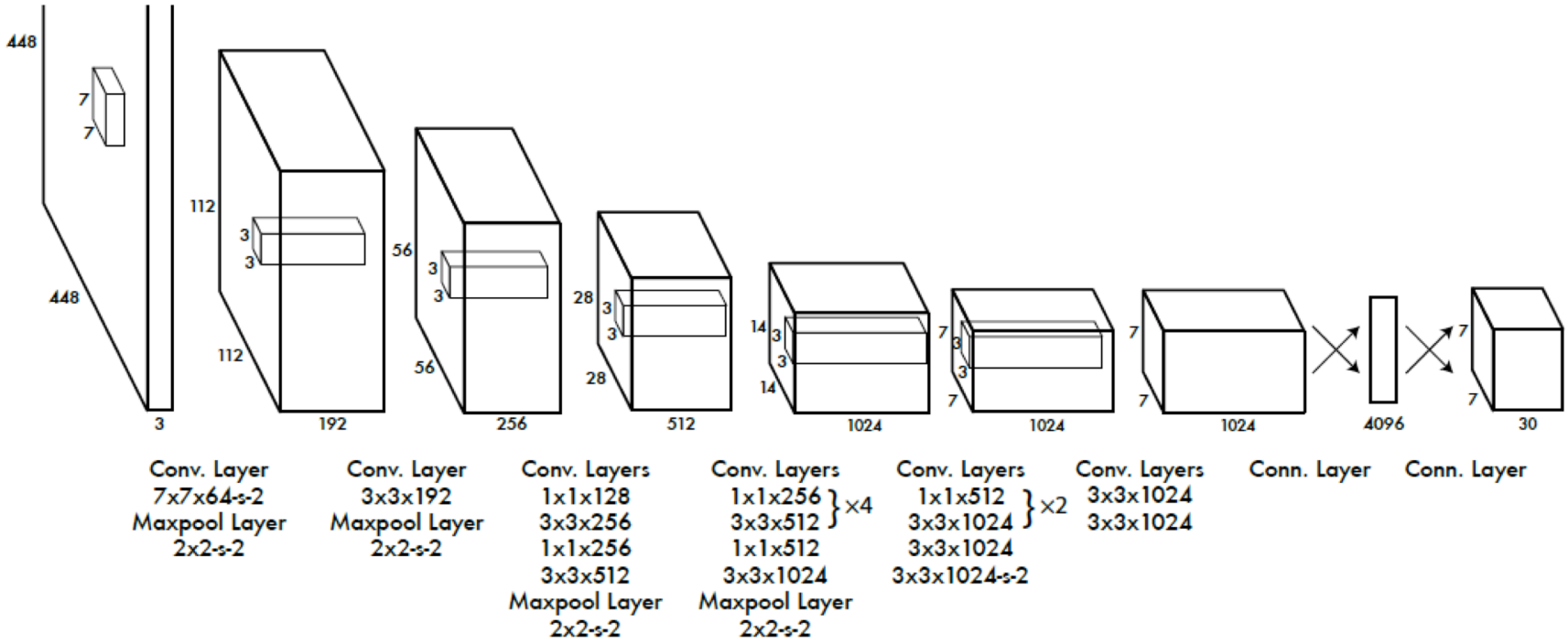
- e.g. PASCAL VOC: 7x7x(2x5+20) = 7x7x30 tensor



Network Design

Based on GoogLeNet model

- 24 conv layers + 2 FC layers
- Faster version: 9 conv layers



Loss Function

Multi-part loss function

j-th bbox predictor in cell i is *responsible* for that prediction

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

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

Coordinate penalties when the prediction is wrong for a true positive

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

Objectness confidence penalty

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

Objectness confidence penalty (for recall)

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

Class prediction penalty

Performance

Fast YOLO is the fastest detector on PASCAL

Significantly better than other real-time systems, but not as good as the state-of-the-art methods

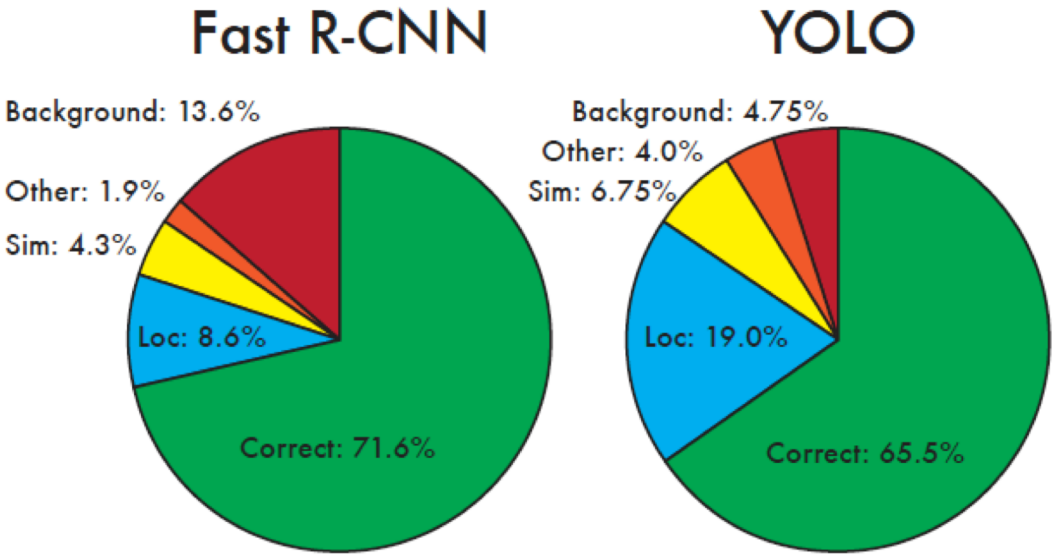
- Faster RCNN with the best accuracy: 7 fps

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

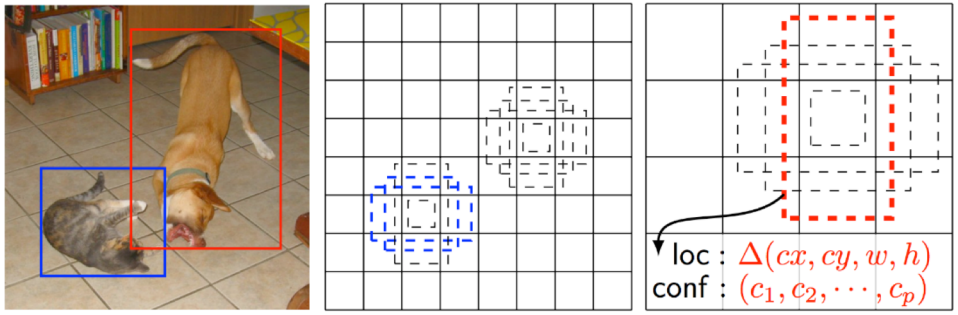
Performance

YOLO struggles to localize object correctly

R-CNN makes many false positives



		mAP	Combined	Gain
	Fast R-CNN	-	71.8	-
Ensemble of Fast R-CNN	Fast R-CNN (2007 data)	**66.9**	72.4	.6
	Fast R-CNN (VGG-M)	59.2	72.4	.6
	Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO + Fast R-CNN	YOLO	63.4	**75.0**	**3.2**



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

SSD

SSD: Single Shot MultiBox Detector (ECCV 2016)

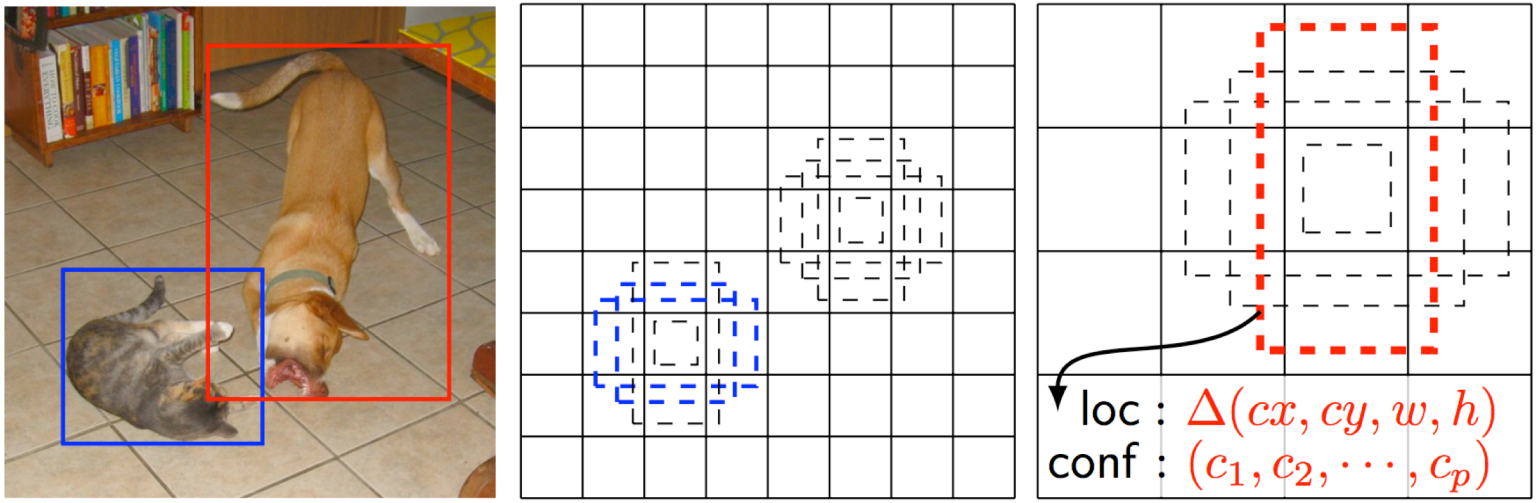
Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg

https://github.com/weiliu89/caffe/tree/ssd

Overview

Single-shot detector: No region proposal like YOLO

- Faster (but not as expected)
- Predict category scores and box offsets for a fixed set of default bounding boxes
- Multi-scale feature maps (c.f. single: Overleaf and YOLO)

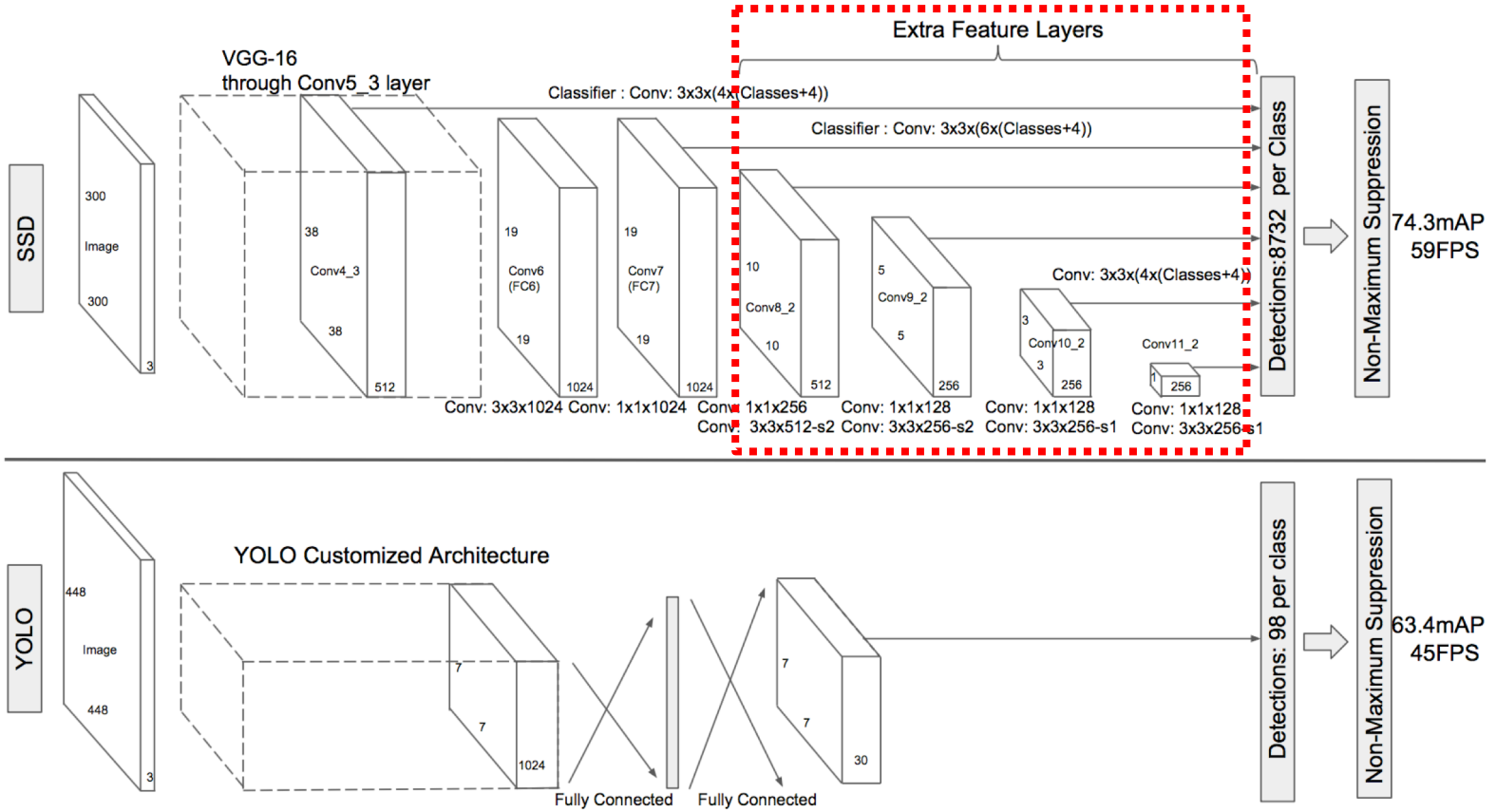


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Multi-scale Feature Maps for Detection

Add CONV feature layers to the end of the base network (e.g. VGG-16)

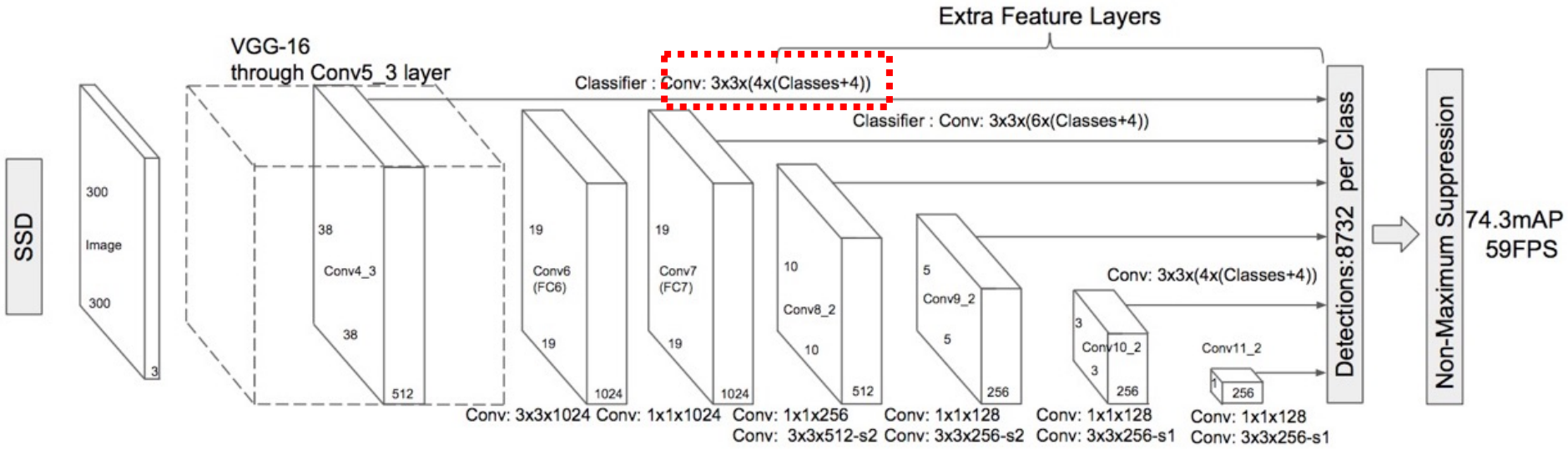
- These layers decrease in size progressively and allow predictions of detections at multiple scales



Convolutional Predictors for Detection

For a feature layer of size $m \times n$ with p channel, apply a $3 \times 3 \times p$ small kernel to each cell

- For each of k boxes at every cell, it produces c class scores and the 4 offsets relative to the default box (i.e. anchor)
- That is, $(c + 4)k$ filters are applied
- $(c + 4)kmn$ outputs for a $m \times n$ feature map



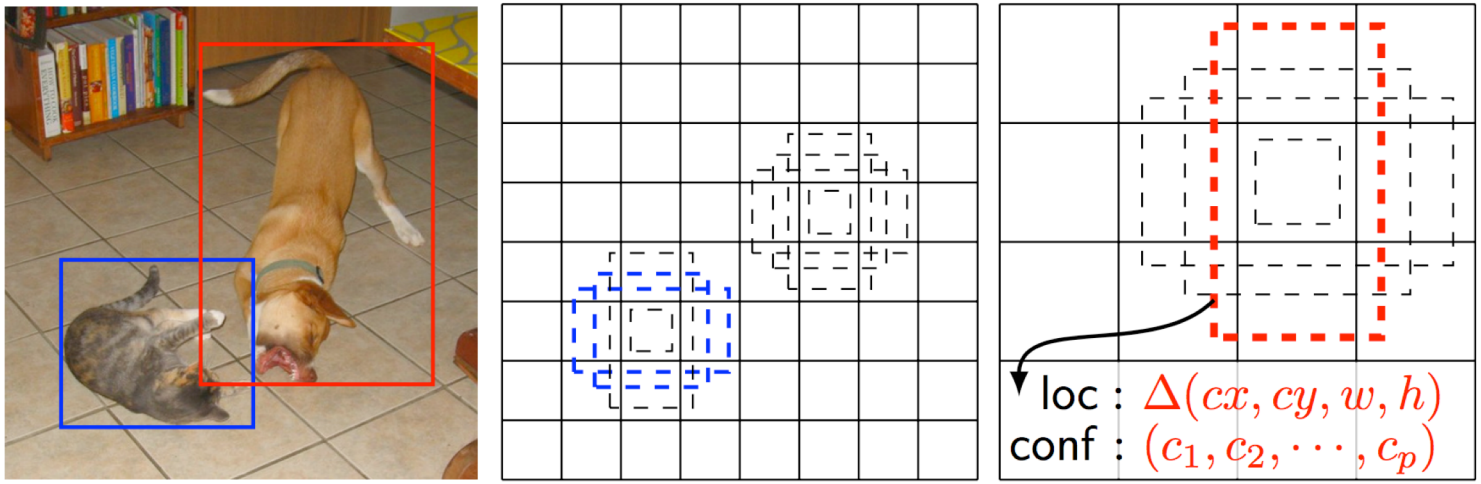
Default Boxes and Aspect Ratios

The scale of the default boxes for each of m feature maps

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

where $s_{min} = 0.2$ and $s_{max} = 0.9$

- Impose different aspect ratios for the default boxes, and denote them as $a_r \in \{1, 2, 3, {}^1/_2, {}^1/_3\}$
- Width/height for each default box: $w_k^a = s_k\sqrt{a_r}$, $h_k^a = s_k/\sqrt{a_r}$



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

Results

PASCAL VOC2007 test detection

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	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
Fast [6]	07+12	70.0	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
Faster [2]	07	69.9	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
Faster [2]	07+12	73.2	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
Faster [2]	07+12+COCO	78.8	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
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	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
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**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**

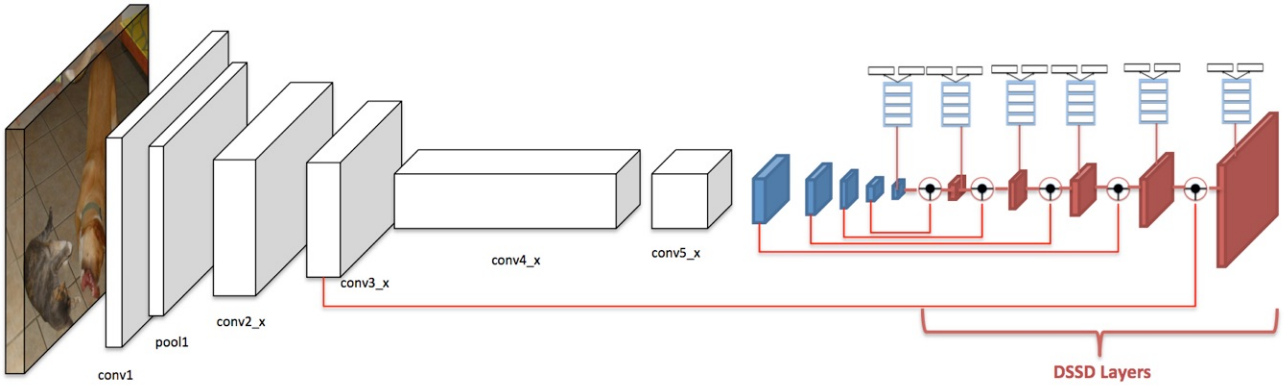
PASCAL VOC2012 test detection

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast[6]	07++12	68.4	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
Faster[2]	07++12	70.4	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
Faster[2]	07++12+COCO	75.9	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
YOLO[5]	07++12	57.9	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
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	**88.4**	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

Results

COCO test-dev2015 detection

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [23]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [23]	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
Faster [24]	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
SSD300	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	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**



DSSD

DSSD: Deconvolutional Single Shot Detector (Arxiv 2016)

Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, Alexander C. Berg

DSSD: Major Updates from SSD

1. Use Residual-101 as base network

2. Add deconvolution layers

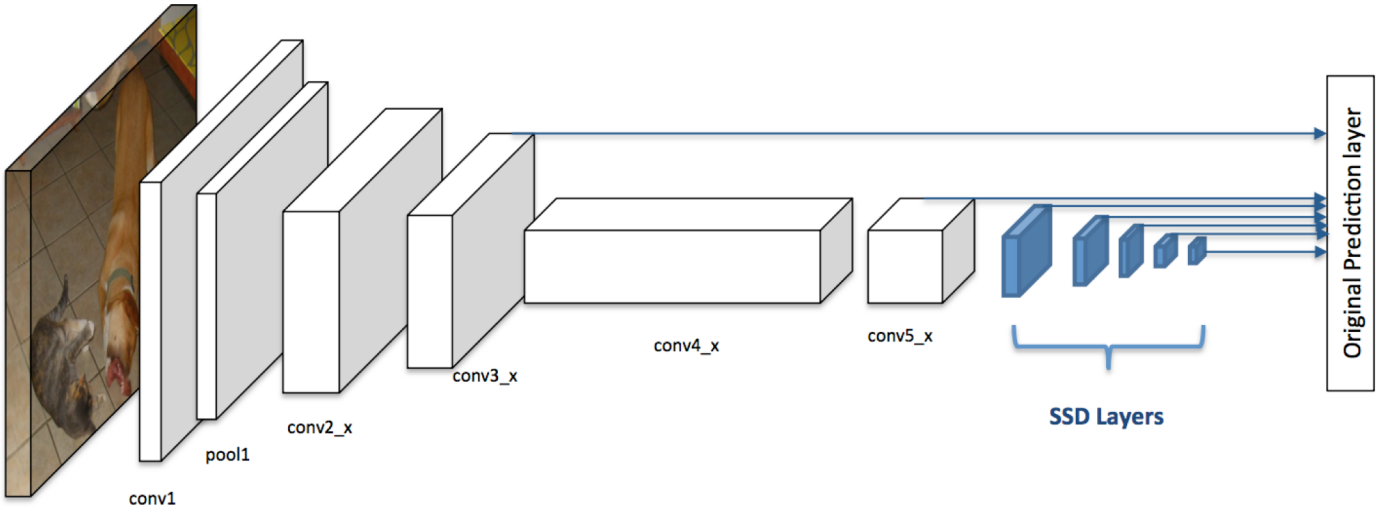
   - Introduce additional large-scale context in object detection
   - Improve accuracy, especially for small objects

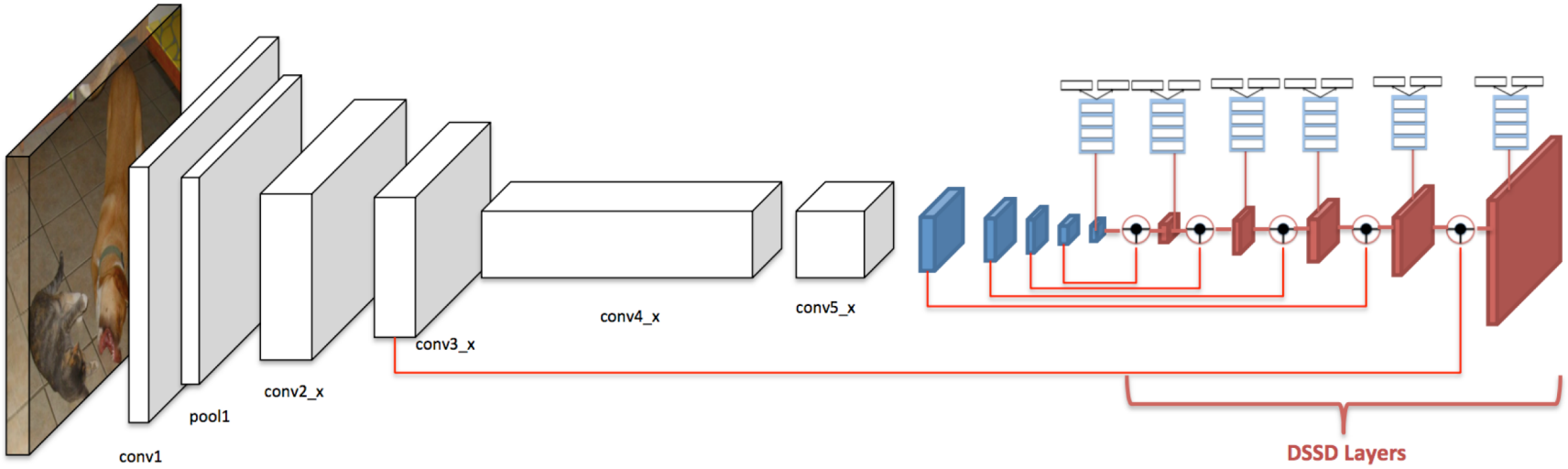
3. Prediction modules with a skip connection

   - Inspired by the success of residual networks

Works well stably with less tuning (from our experience)

SSD vs. DSSD





Two key updates:

Prediction Module

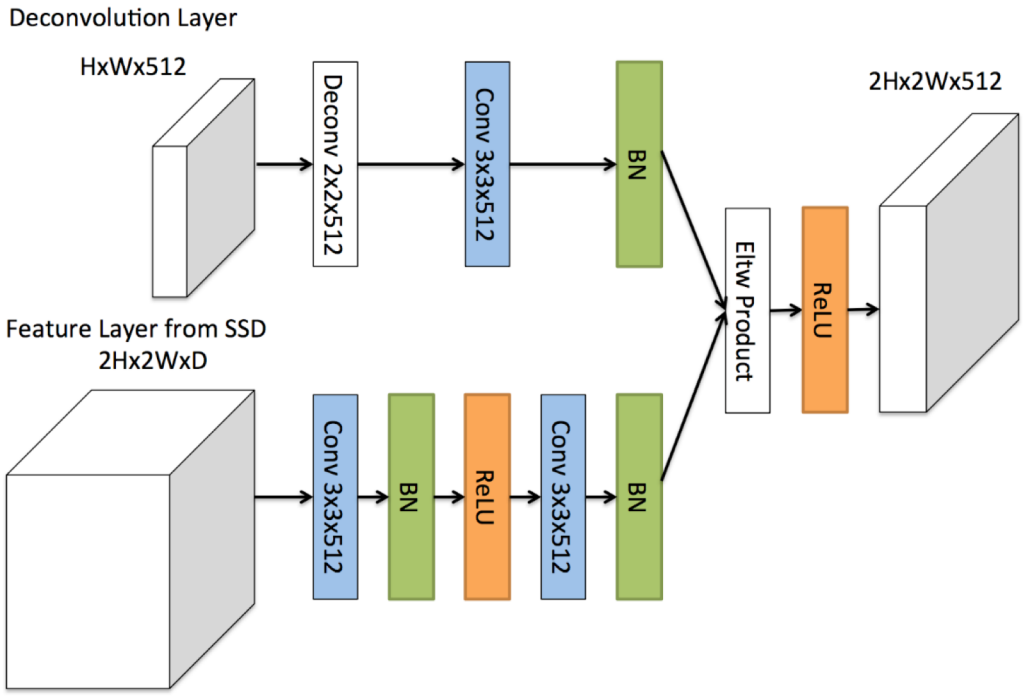
Deconvolution Module

Deconvolution Module

Why? To include more high-level context

- Convolution is local from high-resolution to low-resolution

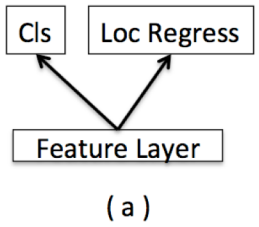
How? To integrate infomation from early feature maps and deconv layers



Variants of Prediction Module

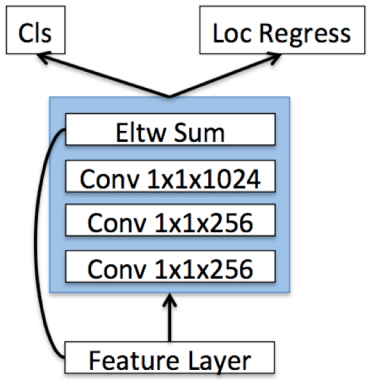
Residual blocks for prediction layer

- From ablation studies, the best was PM(c) with multiple feature combination



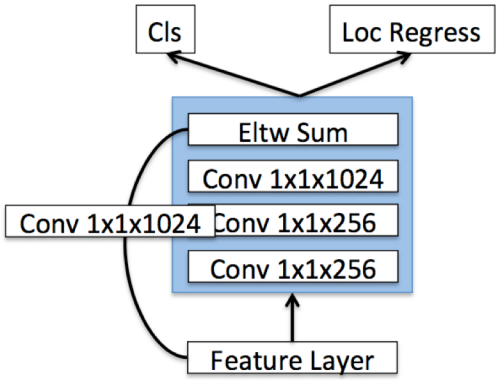
(a)

Original SSD



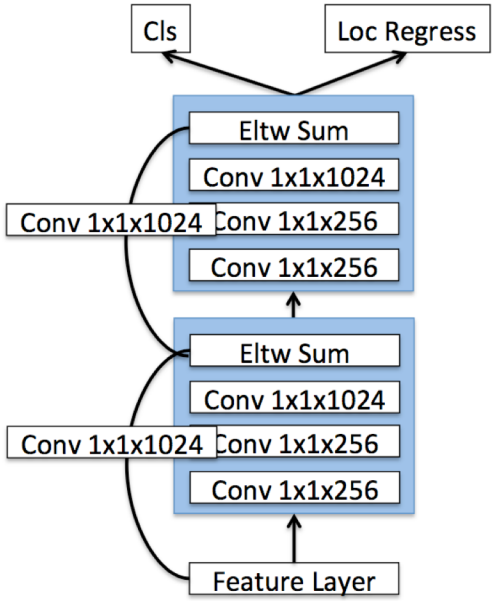
(b)

Residual block with skip connection



(c)

Another form of residual block



(d)

Two sequential residual blocks

Experimental Results

DSSD outperforms SSD

Input size matters

- 513x513 is better than 321x312

Method	network	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster [24]	VGG	73.2	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
ION [1]	VGG	75.6	79.2	83.1	77.6	65.6	54.9	85.4	85.1	87.0	54.4	80.6	73.8	85.3	82.2	82.2	74.4	47.1	75.8	72.7	84.2	80.4
Faster [14]	Residual-101	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	**89.8**	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
MR-CNN [10]	VGG	78.2	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
R-FCN [3]	Residual-101	80.5	79.9	**87.2**	81.5	72.0	**69.8**	86.8	88.5	89.8	**67.0**	**88.1**	74.5	**89.8**	**90.6**	79.9	81.2	**53.7**	81.8	81.5	85.9	79.9
SSD300*[18]	VGG	77.5	79.5	83.9	76.0	69.6	50.5	87.0	85.7	88.1	60.3	81.5	77.0	86.1	87.5	83.97	79.4	52.3	77.9	79.5	87.6	76.8
SSD 321	Residual-101	77.1	76.3	84.6	79.3	64.6	47.2	85.4	84.0	88.8	60.1	82.6	76.9	86.7	87.2	85.4	79.1	50.8	77.2	**82.6**	**87.3**	76.6
DSSD 321	Residual-101	78.6	81.9	84.9	80.5	68.4	53.9	85.6	86.2	88.9	61.1	83.5	78.7	86.7	88.7	86.7	79.7	51.7	78.0	80.9	87.2	79.4
SSD512*[18]	VGG	79.5	84.8	85.1	81.5	73.0	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79.0	86.6	80.0
SSD 513	Residual-101	80.6	84.3	87.6	**82.6**	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD 513	Residual-101	**81.5**	**86.6**	86.2	**82.6**	**74.9**	62.5	**89.0**	**88.7**	88.8	65.2	87.0	**78.7**	88.2	89.0	**87.5**	**83.7**	51.1	**86.3**	81.6	85.7	**83.7**

Table 3: **PASCAL VOC2007 `test` detection results.** R-CNN series and R-FCN use input images whose minimum dimension is 600. The two SSD models have exactly the same settings except that they have different input sizes (321×321 vs. 513×513). It order to fairly compare models, although Faster R-CNN with Residual network [14] and R-FCN [3] provide the number using multiple cropping or ensemble method in testing. We only list the number without these techniques.

YOLOV2/YOLO9000

YOLO9000: Better, Faster, Stronger (CVPR 2017)

Joseph Redmon and Ali Farhadi

https://github.com/philipperemy/yolo-9000

YOLOv2: Better

A list of heuristics to make more accurate without scaling up the network

1. Batch normalization
    - Able to remove dropout from the model without overfitting

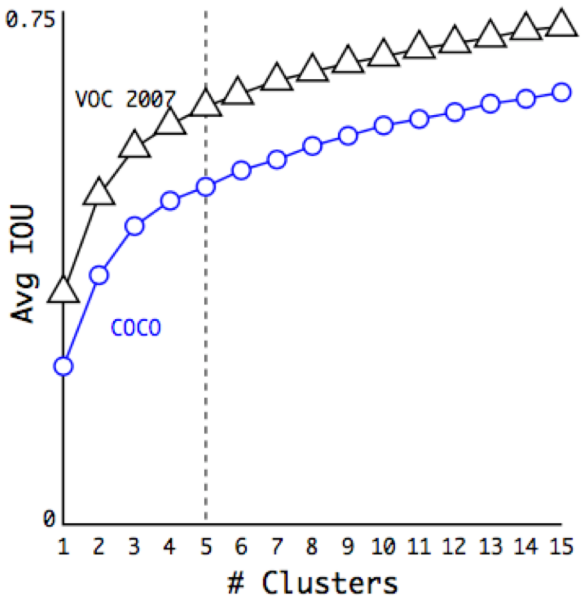
2. High resolution classifier
    - The original YOLO used 224x224 for pre-training and 448x448 for fine-tuning. YOLOv2 used 448x448 from the beginning

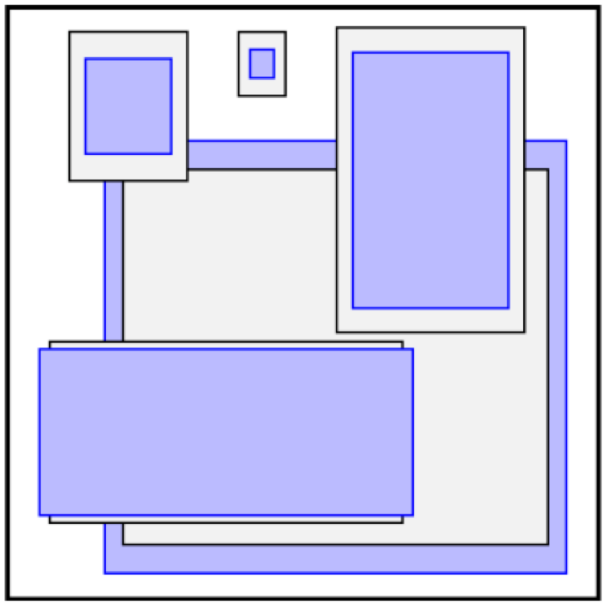
3. Convolutional with anchor boxes
    - Remove the FC layers from YOLO and use anchor boxes to predict bounding boxes

YOLOv2: Better

4. Dimension clusters

- In YOLO, anchor box dimensions are hand-picked
- Run the k-means clustering and find out k anchors (per cell of the SxS grid)



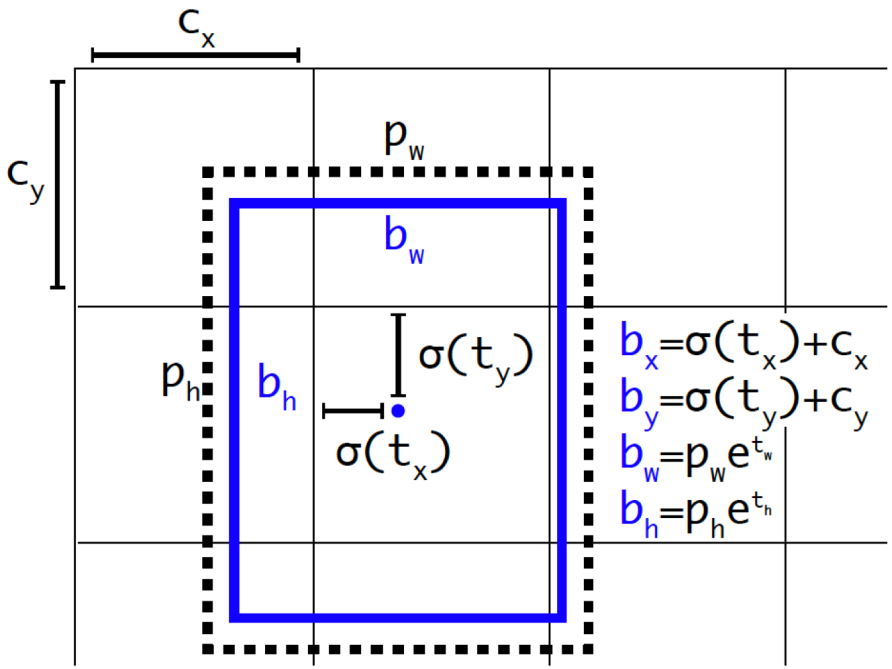


YOLOv2: Better

5. Direct location prediction

- Related to model stability
- Predict 5 coordinates for each bbox: t_x, t_y, t_w, t_h, t_o
- bbox prior by k-means: p_w, p_h

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$



YOLOv2: Better

6. Fine-grained features

- The original YOLO used only the last layer of feature map,
- YOLOv2 uses both conv16 (26x26x512) conv24 (13x13x1024)

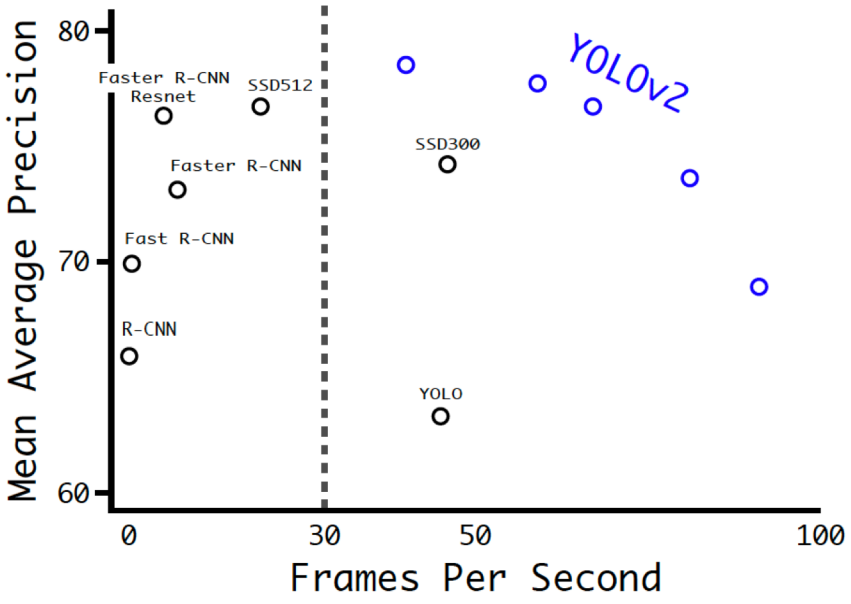
7. Multi-scale training

- The original YOLO used 448x448 input images
- Instead of fixing the input image size, randomly choose a new image dimension every 10 batches
- Pull from the following multiples of 32: {320, 352, ..., 608}, since the model downsamples by a factor of 32

YOLOv2: Faster

YOLOv2 uses DarkNet-19

- YOLO uses GoogLeNet instead of VGG-16, although its accuracy is slightly worse than VGG-16
- 19 Conv + 5 Max-pooling



Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

YOLOv2: Results

Path from YOLO to YOLOv2

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

YOLOv2: Results

Tradeoff between speed and accuracy at different resolutions

- PASCAL VOC 2007

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

YOLOv2: Results

PASCAL VOC 2012test detection

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	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
Faster R-CNN [15]	07++12	70.4	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
YOLO [14]	07++12	57.9	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
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

COCO test-dev 2015

		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [5]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN[1]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN[15]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	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
Faster R-CNN[10]	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
SSD300 [11]	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 [11]	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**
YOLOv2 [11]	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

YOLOv2: Stronger

Joint classification and detection

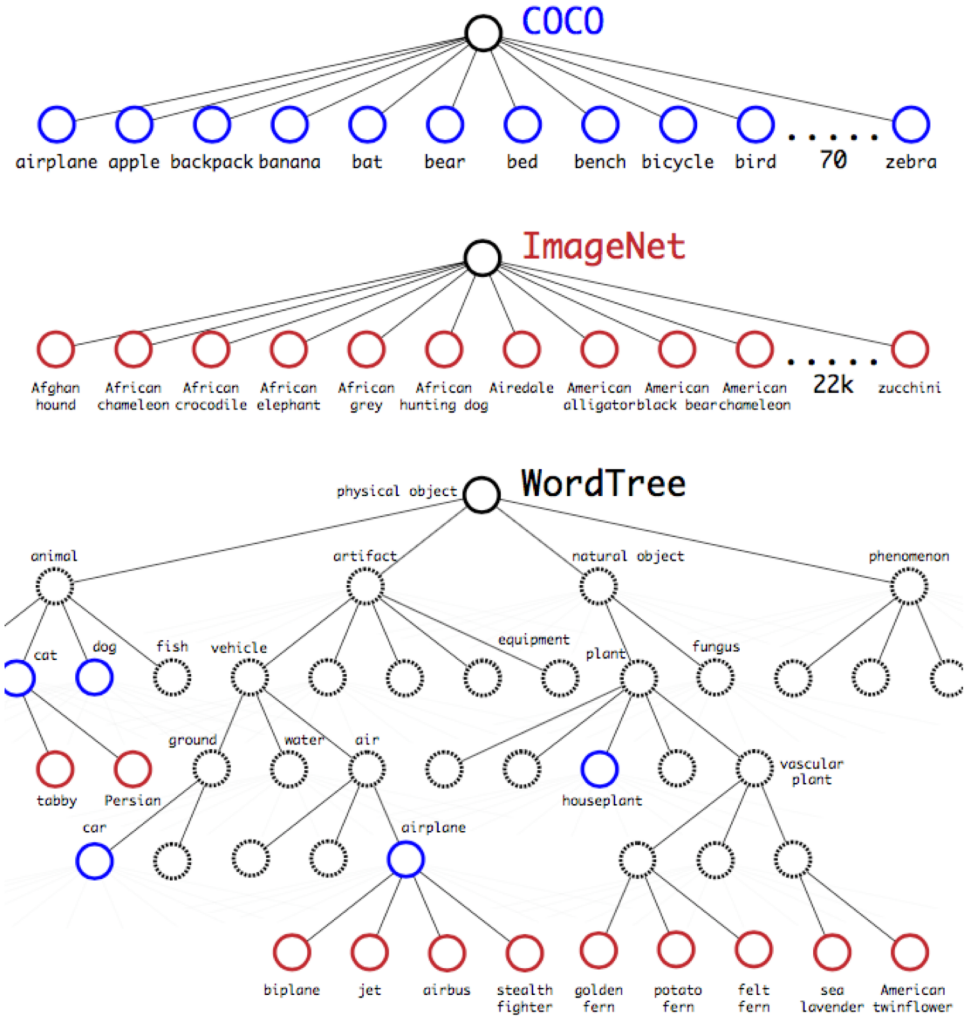
- Use labeled images for detection, to learn detection-specific information (e.g. bbox coordinates, objectness, common object prediction)
- Use labeled images for *classification*, to expand the number of categories it can detect

However, it is not straightforward …

- Classification datasets have a much wider and deeper range of labels
- e.g. ImageNet has more than a hundred breeds of dog, such as *Norfolk terrier*, *Yorkshire terrier*, *Bedlington terrier*

YOLOv2: Stronger

Build a WordTree (a hierarchical model of visual concepts)

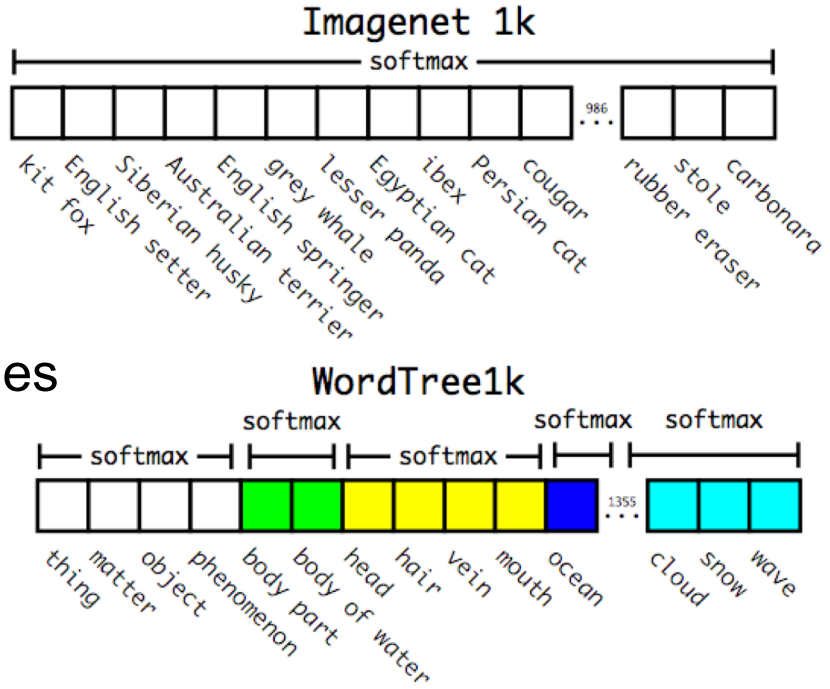


- ImageNet labels are pulled from WordNet (a directed graph)
- Build a tree using some heuristics from WordNet
- Combine multiple datasets together

YOLOv2: Stronger

Hierarchical classification

- Perform multiple softmax operations over co-hyponyms
- Then we can compute the following conditional probabilities



$$Pr(\text{Norfolk terrier}|\text{terrier})$$
$$Pr(\text{Yorkshire terrier}|\text{terrier})$$
$$Pr(\text{Bedlington terrier}|\text{terrier})$$
$$\ldots$$

- We can also compute absolute probability for any node by simply following the path through the tree to the root

$$Pr(\text{Norfolk terrier}) = Pr(\text{Norfolk terrier}|\text{terrier}) * Pr(\text{terrier}|\text{hunting dog}) * \ldots * Pr(\text{mammal}|Pr(\text{animal}) * Pr(\text{animal}|\text{physical object})$$

YOLOv2: Stronger

Joint classification and detection

- When the network sees a *detection* image we backpropagate loss as normal
- When it sees a *classification* image we only backpropagate classification loss
- To do this they simply find the bbox that predicts the highest probability for that class and we compute the loss on its predicted tree

For ImageNet detection task

- YOLO9000 gets 19.7 mAP overall with 16.0 mAP on the disjoint 156 object classes that it has never seen any labelled detection data for

YOLOv2: Demo

https://youtu.be/VOC3huqHrss

Outline

- Two-stage Models
  - R-CNN (CVPR'14), SPPNet (ECCV'14), Fast R-CNN (ICCV'15), Faster R-CNN (NIPS'15)
- Single-stage Models
  - YOLO (CVPR' 16), YOLO v2/9000 (CVPR'17), SSD (ECCV' 16), DSSD (ArXiv'17)
- Application: OCR
  - Text Detection
  - CRAFT Model (Naver Clova)

Text as a Cue in Visual Recognition

Text is complementary to other visual cues, such as contour, color and texture

- Recognizing places, objects, signs and more

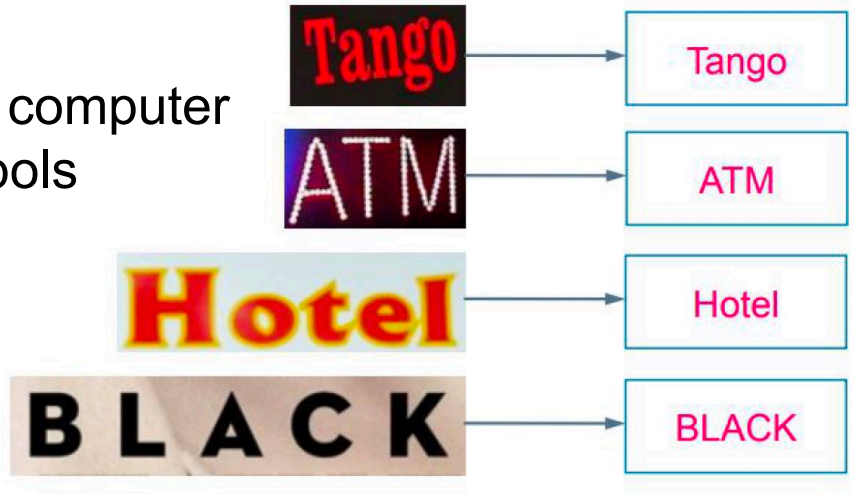
Problem Definition

Scene text detection

- Predicting the presence of text and localizing each instance (if any), usually at word or line level, in natural scenes

Scene text recognition

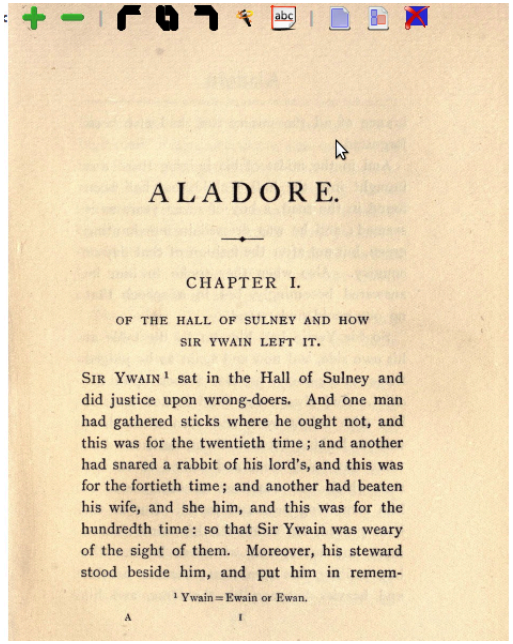
- Converting text regions into computer readable and editable symbols



Challenges

Traditional OCR vs. text detection and recognition

- Clean background vs. cluttered background
- Regular font vs. various fonts
- Plain layout vs. complex layouts
- Monotone color vs. different colors



ALADORE.

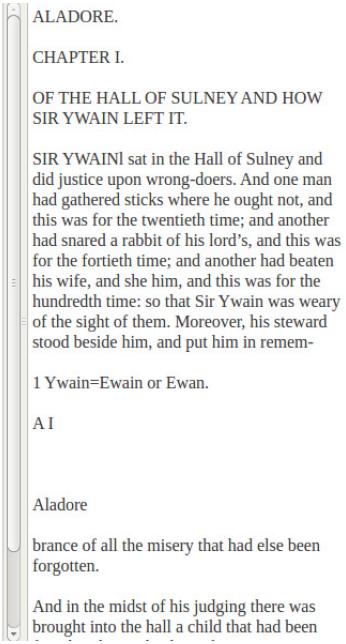
CHAPTER I.

OF THE HALL OF SULNEY AND HOW SIR YWAIN LEFT IT.

SIR YWAIN[1] sat in the Hall of Sulney and did justice upon wrong-doers. And one man had gathered sticks where he ought not, and this was for the twentieth time; and another had snared a rabbit of his lord's, and this was for the fortieth time; and another had beaten his wife, and she him, and this was for the hundredth time: so that Sir Ywain was weary of the sight of them. Moreover, his steward stood beside him, and put him in remem-

[1] Ywain=Ewain or Ewan.

A I



ALADORE.

CHAPTER I.

OF THE HALL OF SULNEY AND HOW SIR YWAIN LEFT IT.

SIR YWAINl sat in the Hall of Sulney and did justice upon wrong-doers. And one man had gathered sticks where he ought not, and this was for the twentieth time; and another had snared a rabbit of his lord's, and this was for the fortieth time; and another had beaten his wife, and she him, and this was for the hundredth time: so that Sir Ywain was weary of the sight of them. Moreover, his steward stood beside him, and put him in remem-

1 Ywain=Ewain or Ewan.

A I

Aladore

brance of all the misery that had else been forgotten.

And in the midst of his judging there was brought into the hall a child that had been

Challenges

Diversity of scene text

- Different colors, scales, orientations, fonts, languages…

Challenges

Complexity of background

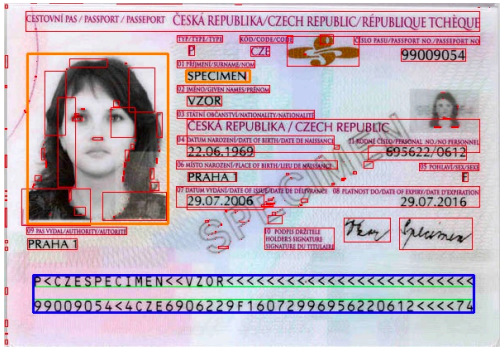
- Elements like signs, fences, bricks, and grasses are virtually indistinguishable from true text

Challenges

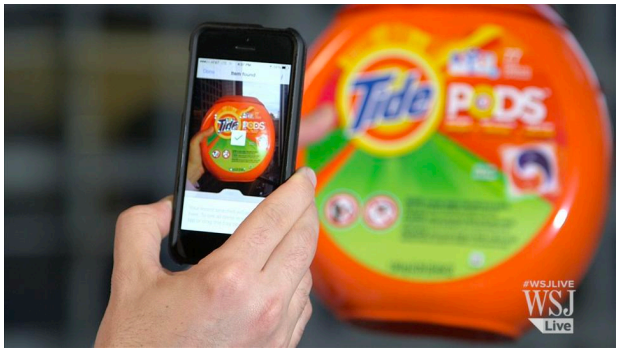
Various interference factors

- Noise, blur, non-uniform illumination, low resolution, partial occlusion…

Applications

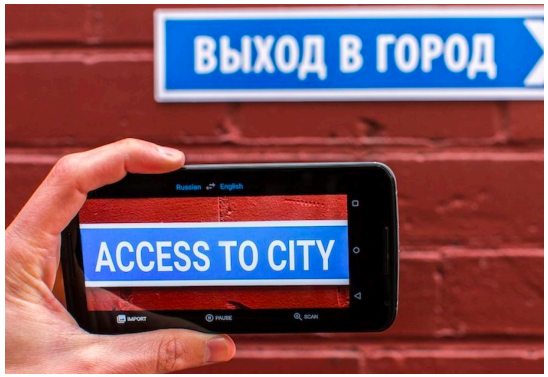


Identity recognition

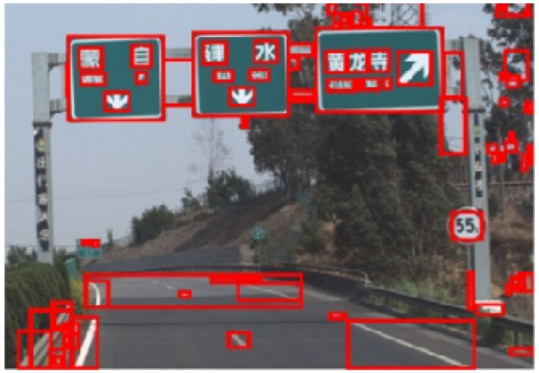


Product search

Navigation



Translation



Self-driving car

Outline

- Two-stage Models
  - R-CNN (CVPR'14), SPPNet (ECCV'14), Fast R-CNN (ICCV'15), Faster R-CNN (NIPS'15)
- Single-stage Models
  - YOLO (CVPR' 16), YOLO v2/9000 (CVPR'17), SSD (ECCV' 16), DSSD (ArXiv'17)
- Application: OCR
  - Text Detection
  - CRAFT Model (Naver Clova)

Y. Baek et al. (Naver Clova), Character Region Awareness for Text Detection, CVPR 2019

Motivation

Previous works mainly focus on

- Detection units: word or line
- Detection shape: rectangle/quadrangle/polygons

Challenging Issues

- Most Issues come from the ambiguity of word-level text detection

Extreme aspect ratio

Distortion

Scale variant

Y. Baek et al. (Naver Clova), Character Region Awareness for Text Detection, CVPR 2019

Approach

Goal: localize texts in a scene image

Character region awareness

- Detect each character and link them into text instance
- Bottom-up approach
- No character level annotation → weakly-supervised training

Character representation

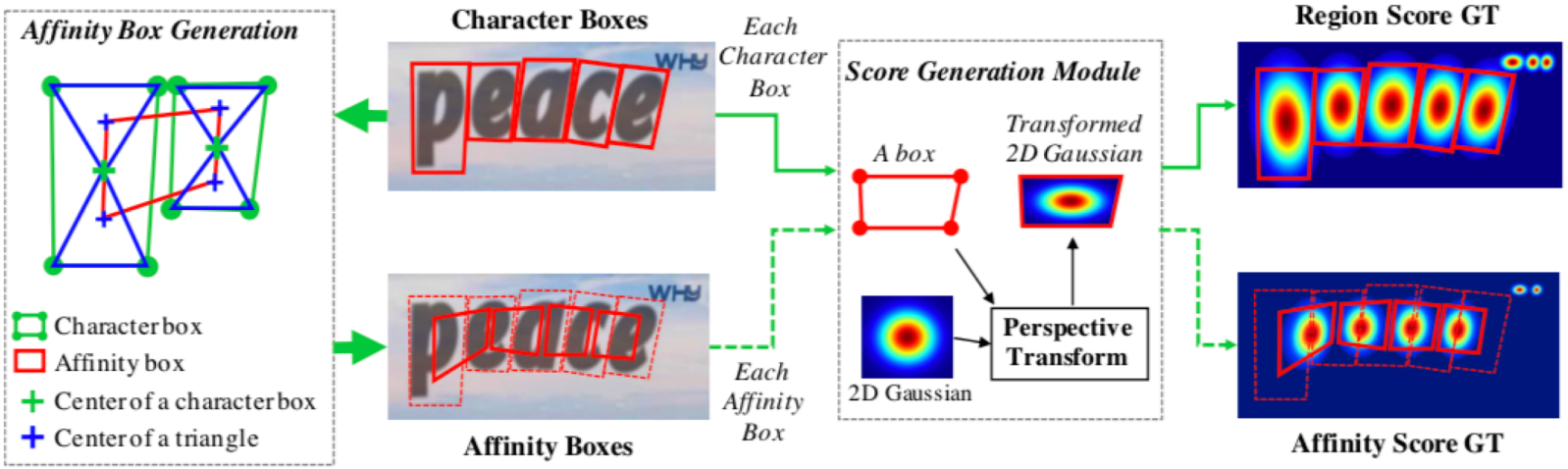
- Bounding box/binary segmentation is not appropriate
- Gaussian heatmap → the probability of the character center

https://github.com/clovaai/CRAFT-pytorch

Proposed Methods

Ground truth generation

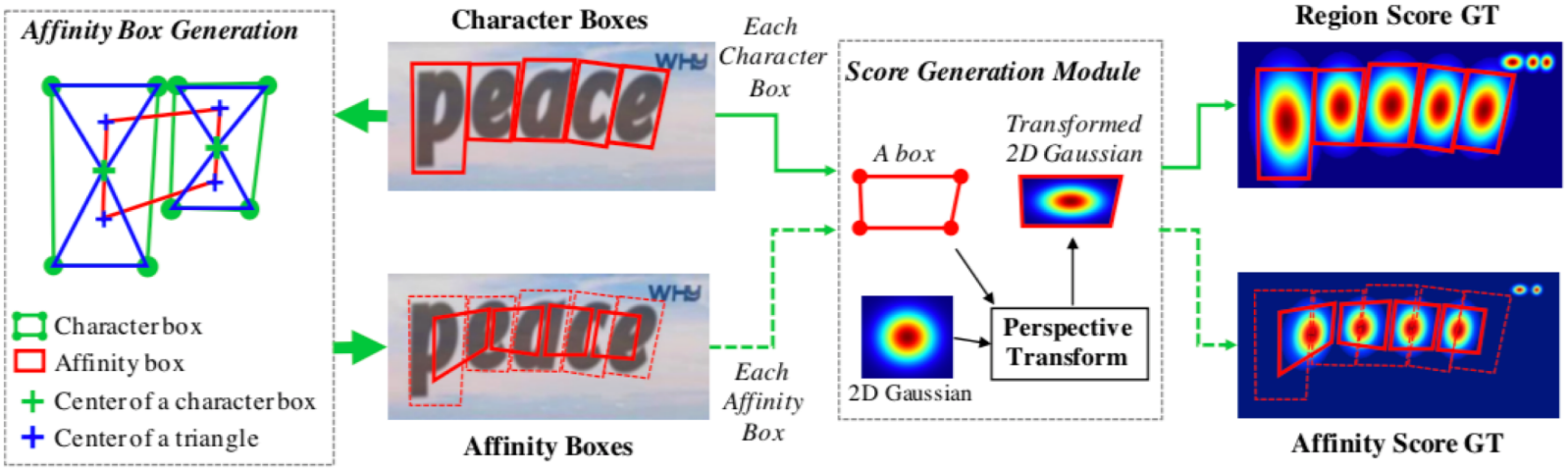
- Input: character GT boxes in synthetic images (e.g. adding text in scene image)
- Output: region score GT and affinity score GT
- Encode the probability of the character/affinity center with a Gaussian heatmap



Proposed Methods

Ground truth generation

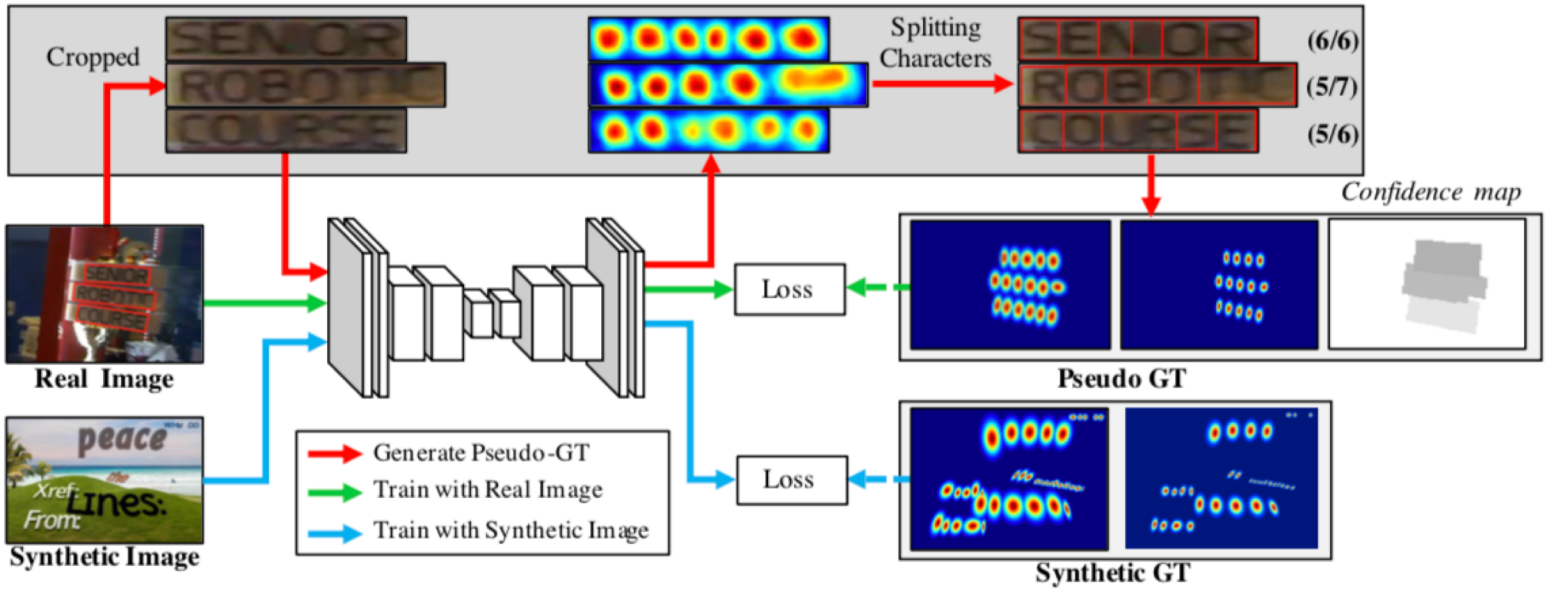
- Since character bounding boxes on an image are generally distorted via perspective projection,
  1. Prepare a 2-dimensional isotropic Gaussian map
  2. Compute perspective transform between the Gaussian map region and each character box
  3. Wrap Gaussian map to the box area



Proposed Methods

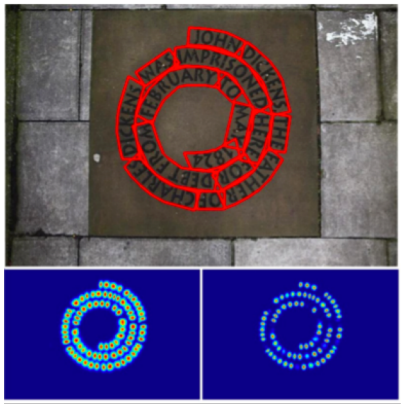
Architecture

- U-net like encoder and decoder network since the input and output are both images
- VGG-16 with batch normalization is adopted as backbone
- Loss: L2 loss between predicted and GT region and affinity scores



Results

TotalText dataset



CTW-1500 dataset

Results

SOTA performance and 1st place in ICDAR challenge

Method	IC13(DetEval)			IC15			IC17			MSRA-TD500			FPS
	R	P	H	R	P	H	R	P	H	R	P	H	
Zhang et al. [39]	78	88	83	43	71	54	-	-	-	67	83	74	0.48
Yao et al. [37]	80.2	88.8	84.3	58.7	72.3	64.8	-	-	-	75.3	76.5	75.9	1.61
SegLink [32]	83.0	87.7	85.3	76.8	73.1	75.0	-	-	-	70	86	77	20.6
SSTD [8]	86	89	88	73	80	77	-	-	-	-	-	-	7.7
Wordsup [12]	87.5	93.3	90.3	77.0	79.3	78.2	-	-	-	-	-	-	1.9
EAST* [40]	-	-	-	78.3	83.3	80.7	-	-	-	67.4	87.3	76.1	13.2
He et al. [11]	81	92	86	80	82	81	-	-	-	70	77	74	1.1
R2CNN [13]	82.6	93.6	87.7	79.7	85.6	82.5	-	-	-	-	-	-	0.4
TextSnake [24]	-	-	-	80.4	84.9	82.6	-	-	-	73.9	83.2	78.3	1.1
TextBoxes++* [17]	86	92	89	78.5	87.8	82.9	-	-	-	-	-	-	2.3
EAA [10]	*87*	*88*	*88*	*83*	*84*	*83*	-	-	-	-	-	-	-
Mask TextSpotter [25]	*88.1*	*94.1*	*91.0*	*81.2*	*85.8*	*83.4*	-	-	-	-	-	-	4.8
PixelLink* [4]	87.5	88.6	88.1	82.0	85.5	83.7	-	-	-	73.2	83.0	77.8	3.0
RRD* [19]	86	92	89	80.0	88.0	83.8	-	-	-	73	87	79	10
Lyu et al.* [26]	84.4	92.0	88.0	79.7	89.5	84.3	**70.6**	74.3	72.4	76.2	87.6	81.5	5.7
FOTS [21]	-	-	*87.3*	*82.0*	*88.8*	*85.3*	*57.5*	*79.5*	*66.7*	-	-	-	23.9
CRAFT(ours)	**93.1**	**97.4**	**95.2**	**84.3**	**89.8**	**86.9**	68.2	**80.6**	**73.9**	**78.2**	**88.2**	**82.9**	8.6

Method	TotalText			CTW-1500		
	R	P	H	R	P	H
CTD+TLOC [38]	-	-	-	69.8	77.4	73.4
MaskSpotter [25]	55.0	69.0	61.3	-	-	-
TextSnake [24]	74.5	82.7	78.4	**85.3**	67.9	75.6
CRAFT(ours)	**79.9**	**87.6**	**83.6**	81.1	**86.0**	**83.5**

Method	IC13	IC15	IC17
Mask TextSpotter [25]	91.7	86.0	-
EAA [10]	90	87	-
FOTS [21]	92.8	**89.8**	70.8
CRAFT(ours)	**95.2**	86.9	**73.9**