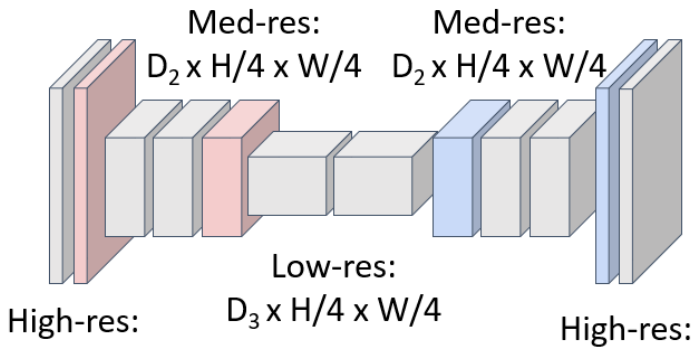
Segmentation and Detection

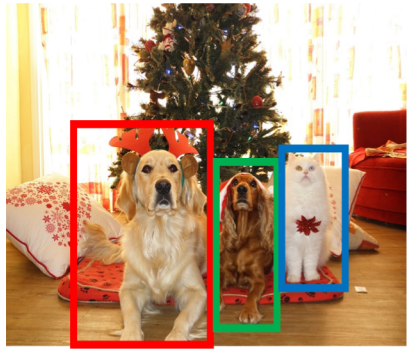
Computer Vision (UW EE/CSE 576)



Richard Szeliski

Facebook & UW

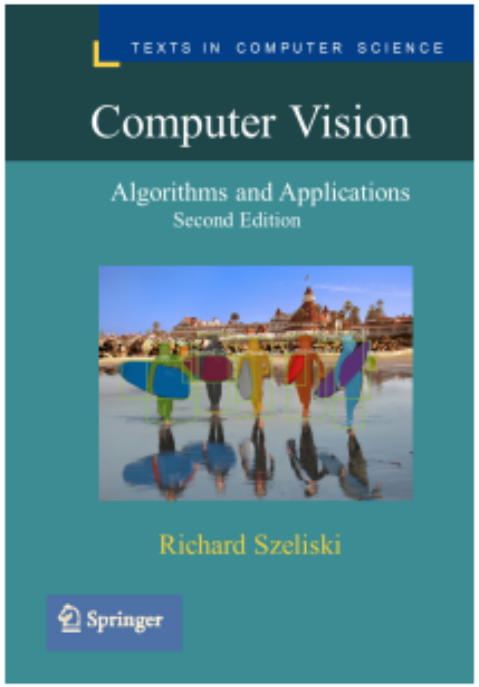
Lecture 10 – Apr 30, 2020

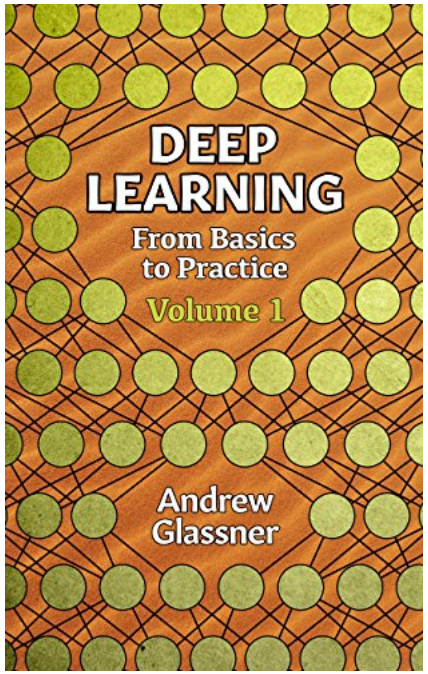


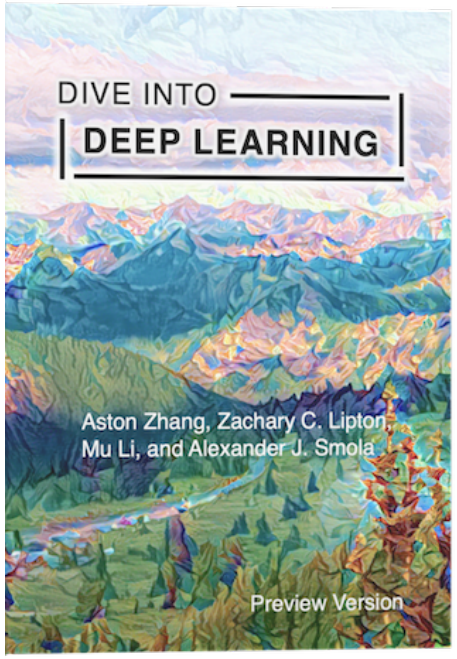
Class calendar

Date	Topic	Slides	Reading	Homework
April 9	Filters and convolutions	Google Slides	Szeliski, Chapter 3	HW1 due, HW2 assigned
April 14	Interpolation and Optimization	pdf, pptx	Szeliski, Chapter 4	
April 16	Machine Learning	pdf, pptx	Szeliski, Chapter 5.1-5.2	
April 21	Deep Neural Networks	pdf, pptx	Szeliski, Chapter 5.3	
April 23	Convolutional Neural Networks	pdf, pptx	Szeliski, Chapter 5.4	HW2 due, HW3 assigned
April 28	Network Architectures	pdf, pptx	Szeliski, Chapter 5.4	
April 30	Segmentation and Detection		Szeliski, Chapter 6.3	
May 5	DL Languages, Instance Segmentation		Szeliski, Chapter 6.4	
May 7	Edges, features, matching, RANSAC		Szeliski, Chapter 7.1-7.2, 8.1-8.2	HW3 due, HW4 assigned

References







https://d2l.ai/

Readings

Chapter 6

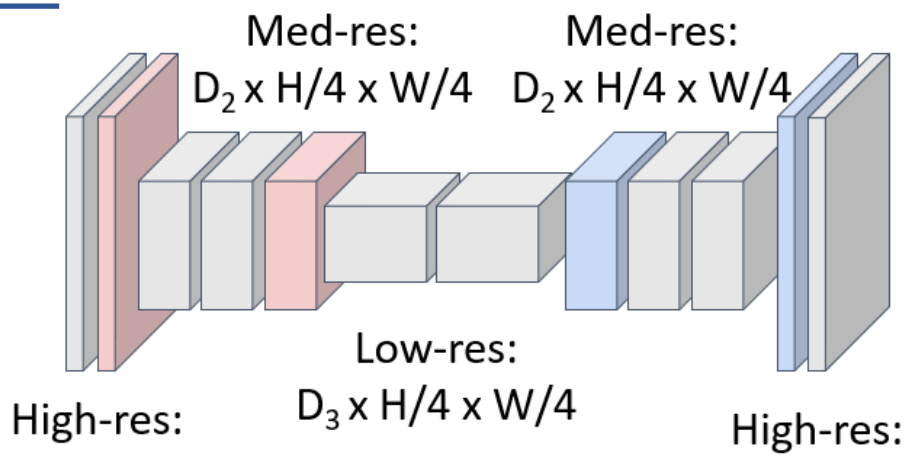
Recognition

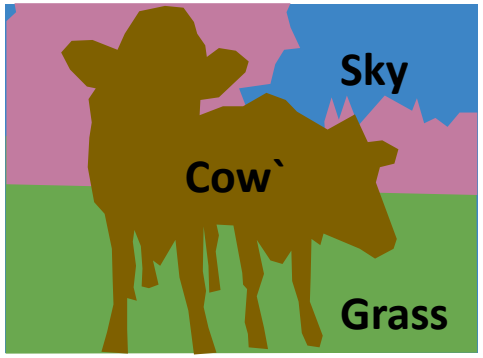
6.1 Instance recognition 319
6.2 Image classification 322
6.2.1 Feature-based methods 323
6.2.2 Deep networks 335
6.2.3 Face recognition 335
6.3 Object detection 341
6.3.1 Face detection 342
6.3.2 Pedestrian detection 350
6.3.3 General object detection 352
6.3.4 *Application*: Image search 354
6.4 Semantic segmentation 355
6.4.1 *Application*: Medical image segmentation 357
6.4.2 Instance segmentation 357
6.4.3 Pose estimation 357
6.4.4 Panoptic segmentation 358
6.4.5 *Application*: Intelligent photo editing 358
6.5 Video understanding 360
6.6 Vision and language 360
6.7 Datasets and benchmarks 360

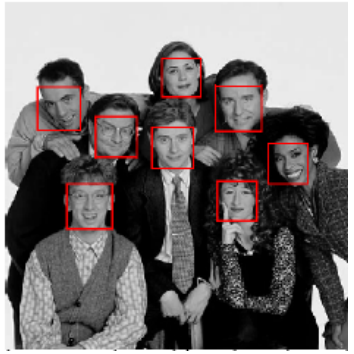
Unfortunately, not yet ready ☹

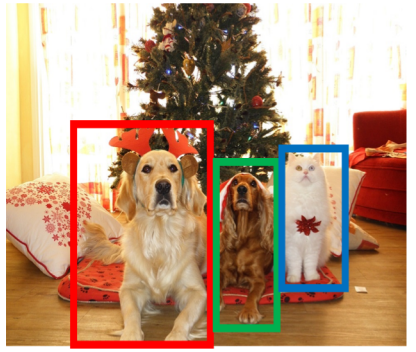
Segmentation and detection

- Adversarial examples
- Bottlenecks and U-Nets
- Segmentation
- Face detection
- Object detection









As before, I'm borrowing slides from



UNIVERSITY OF MICHIGAN

EECS 498-007 / 598-005
Deep Learning for Computer Vision
Fall 2019

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification and object detection. Recent developments in neural network approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of neural-network based deep learning methods for computer vision. During this course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. We will cover learning algorithms, neural network architectures, and practical engineering tricks for training and fine-tuning networks for visual recognition tasks.

Instructor

Graduate Student Instructors

Justin Johnson

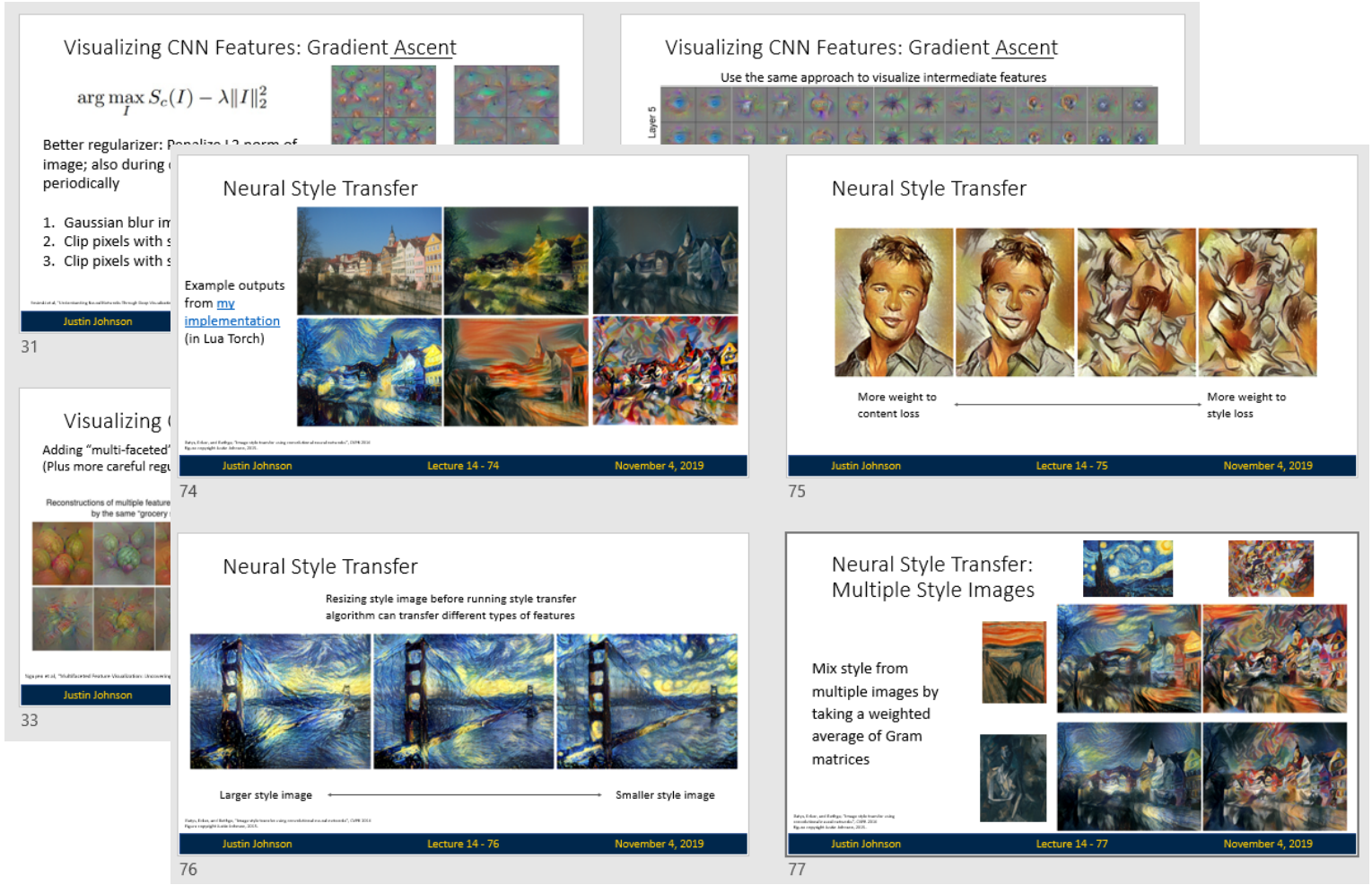
Yunseok Jang

Kibok Lee

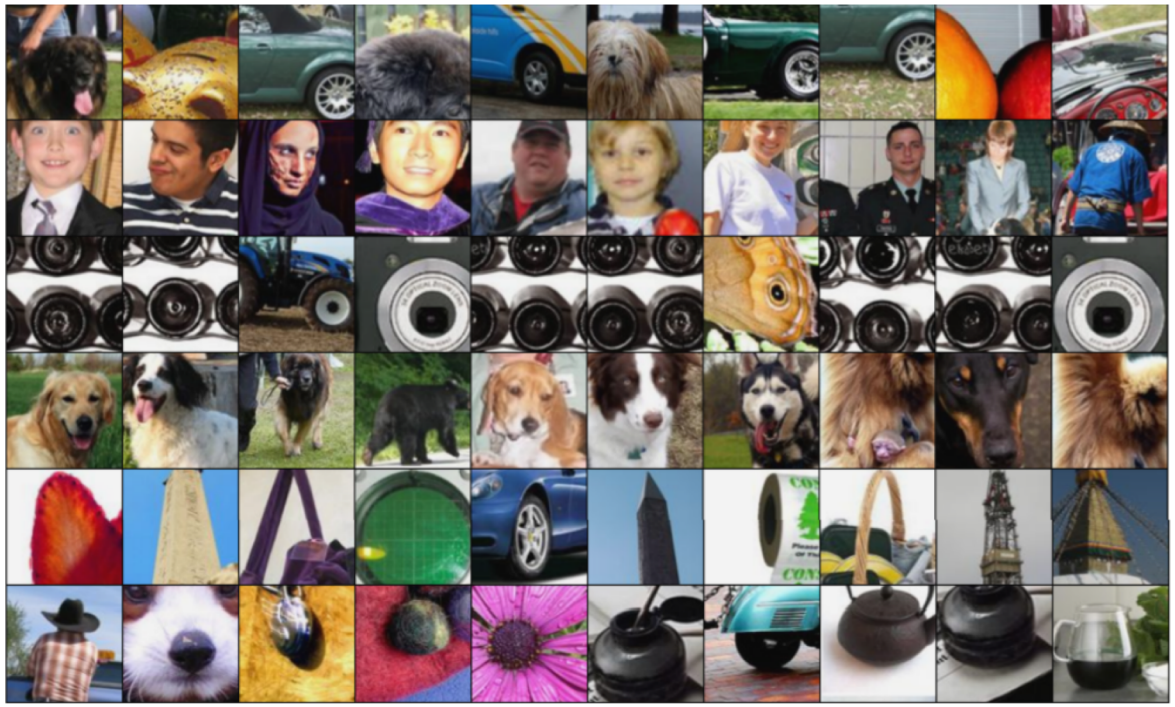
Luowei Zhou

Lecture 14: Visualizing CNNs

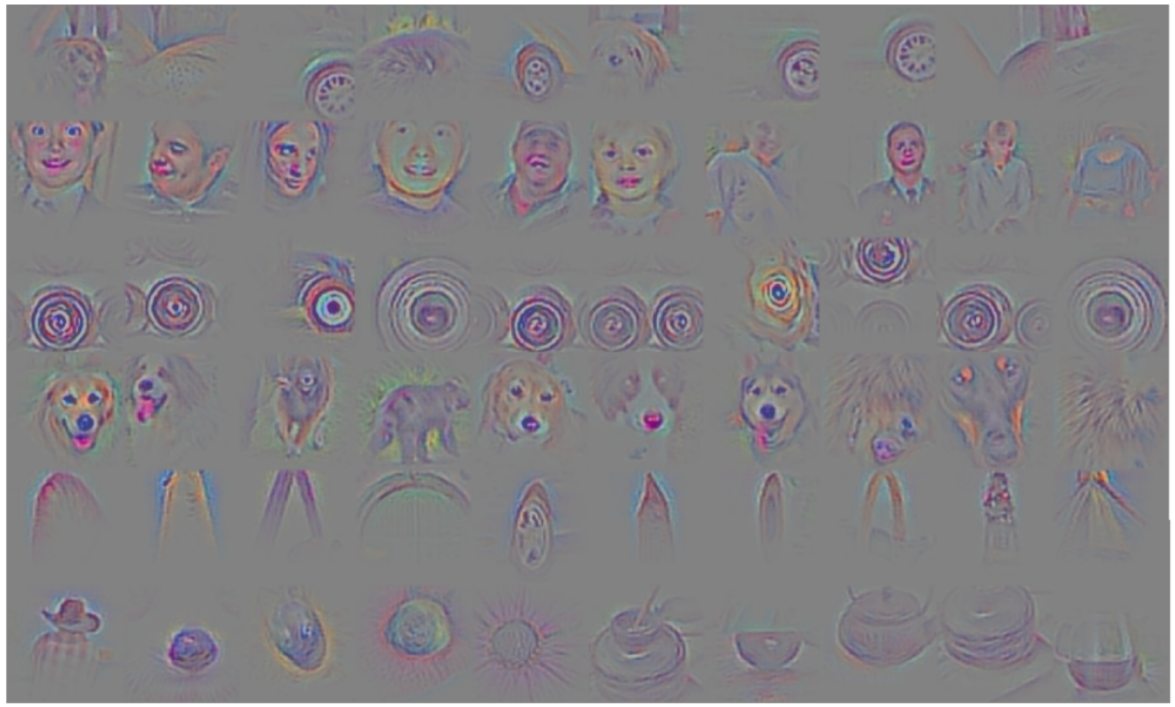
(see slides on Web site)



Intermediate Features via (guided) backprop



Maximally activating patches
(Each row is a different neuron)



Guided Backprop

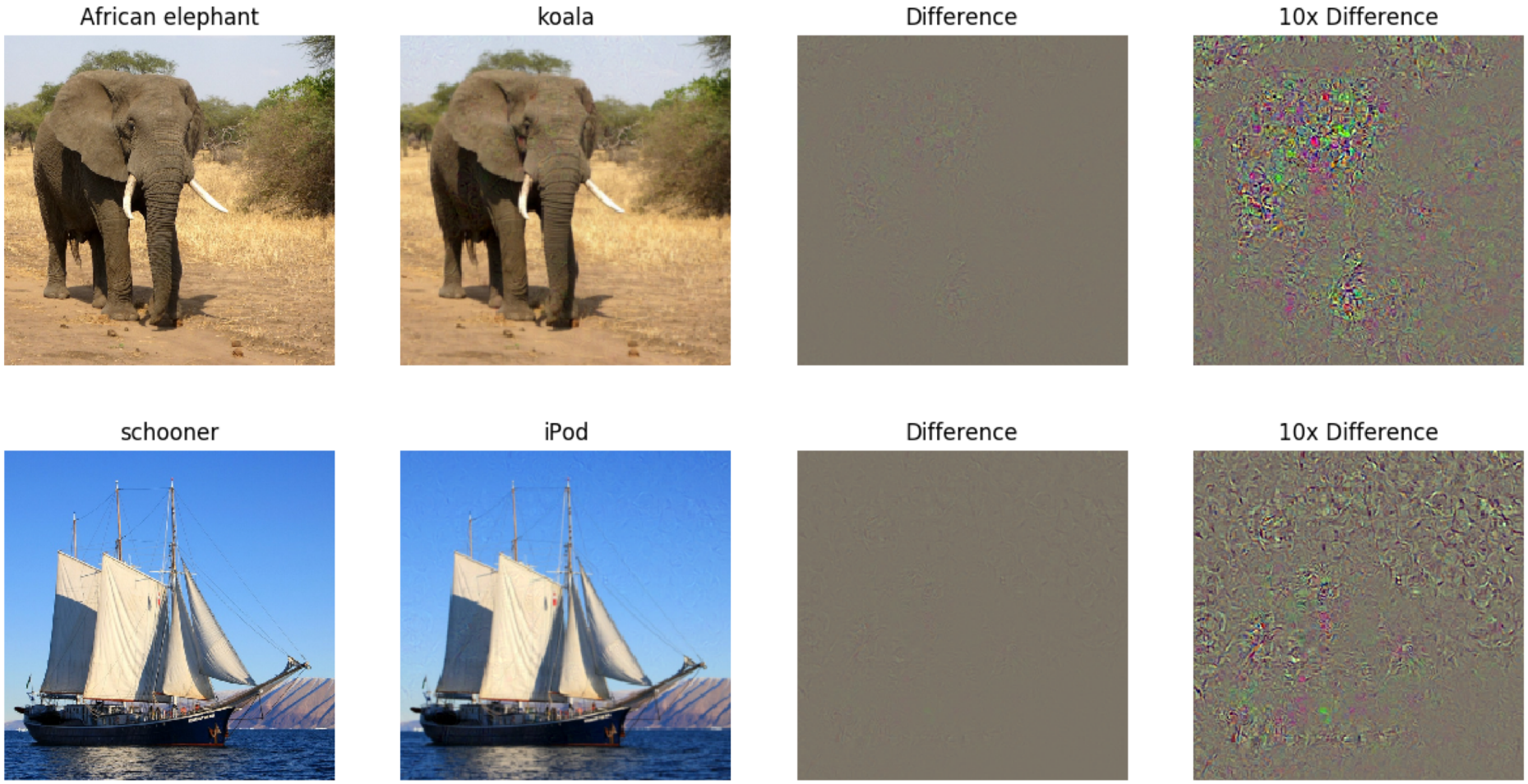
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Adversarial Examples

1. Start from an arbitrary image
2. Pick an arbitrary category
3. Modify the image (via gradient ascent) to maximize the class score
4. Stop when the network is fooled

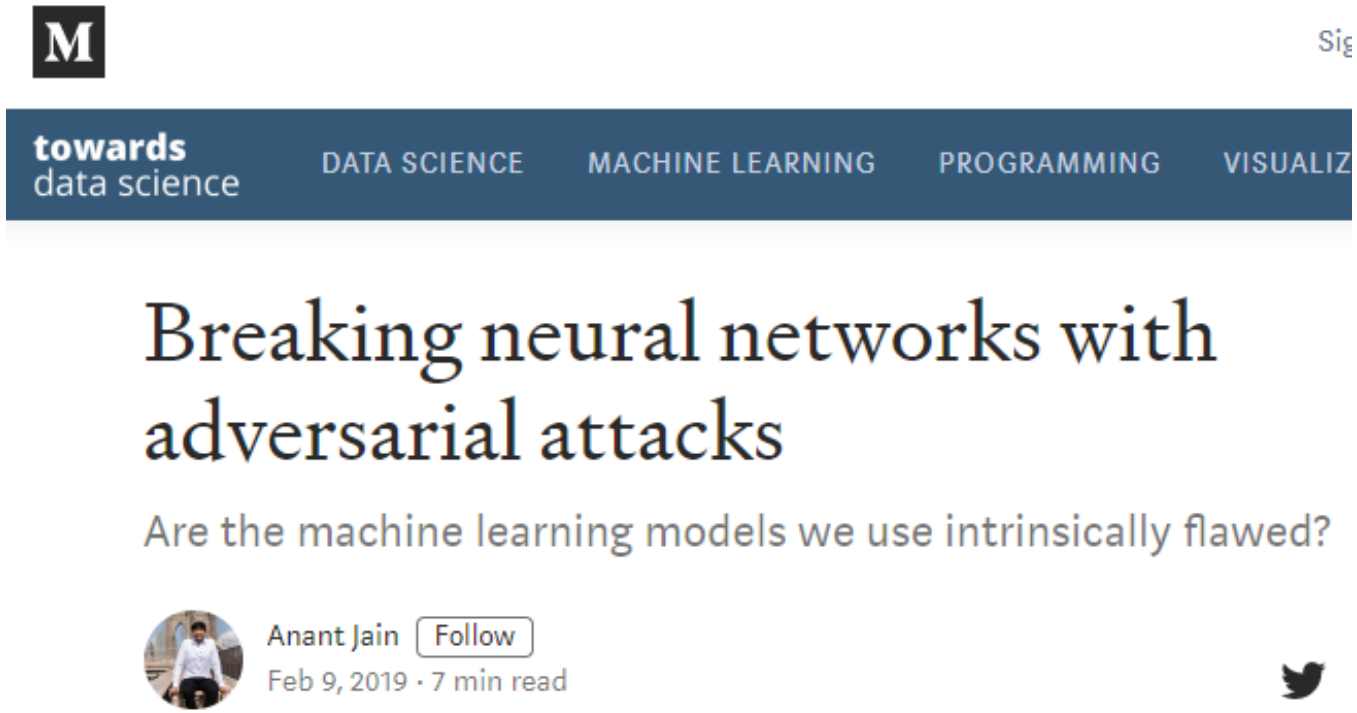
Szegedy et al, “Intriguing properties of neural networks”, 2013

Adversarial Examples



Boat image is CC0 public domain
Elephant image is CC0 public domain

Adversarial examples (con't)



https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa

Adversarial examples (con't)



The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious. The right image shows a physical perturbation applied to a Stop sign. The systems classify the sign on the right as a Speed Limit: 45 mph sign! Source: Robust Physical-World Attacks on Deep Learning Visual Classification.



Why 45?

https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa

Adversarial examples (con't)

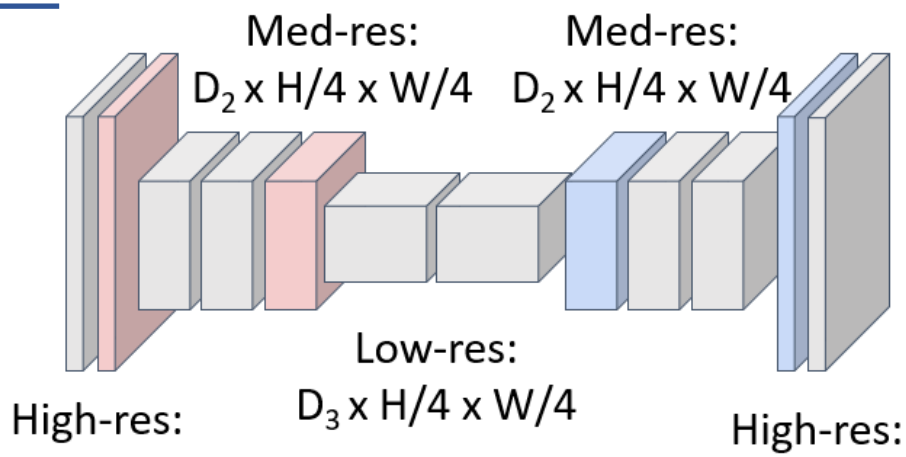


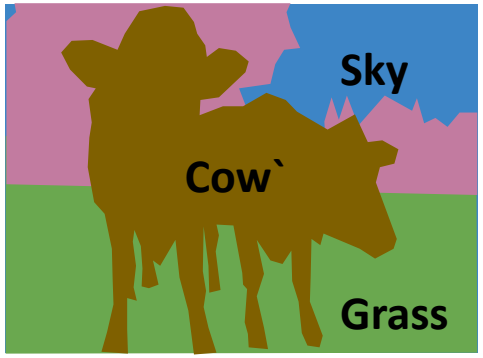
Source: Adversarial Patch: https://arxiv.org/pdf/1712.09665.pdf

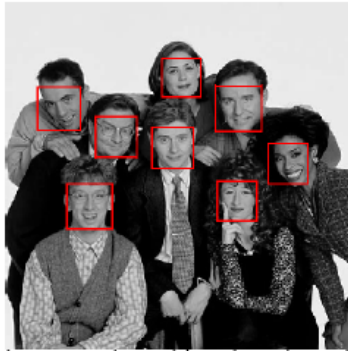
https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa

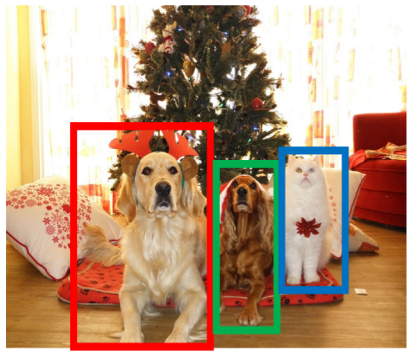
Segmentation and detection

- Adversarial examples
- **Bottlenecks and U-Nets**
- **Segmentation**
- Face detection
- Object detection









Lecture 16: Semantic Segmentation

So far: Image Classification



This image is CC0 public domain

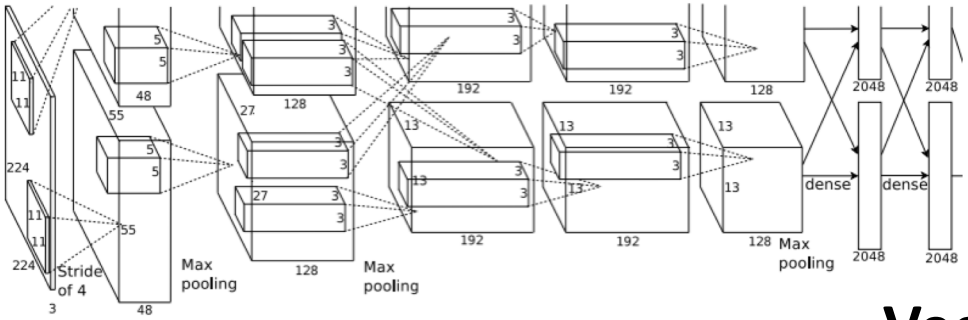


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Computer Vision Tasks: Semantic Segmentation

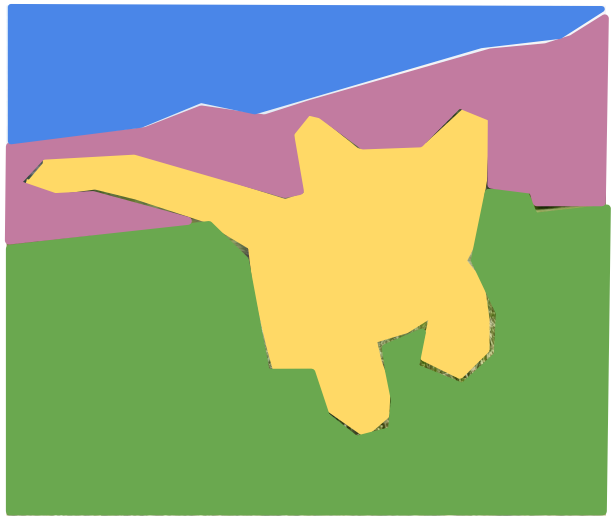
Classification



CAT

No spatial extent

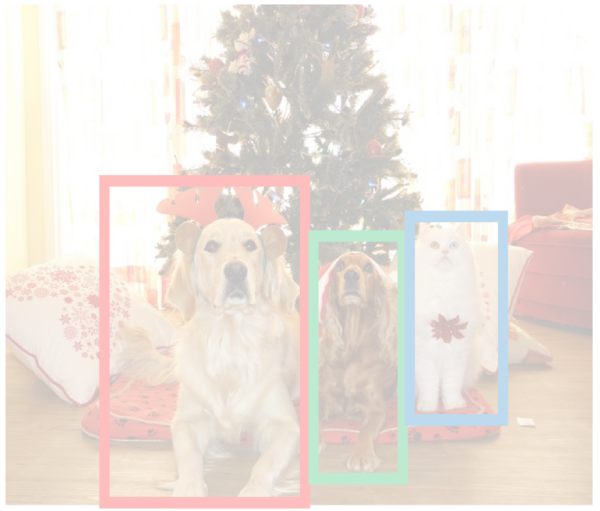
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Objects

Semantic Segmentation

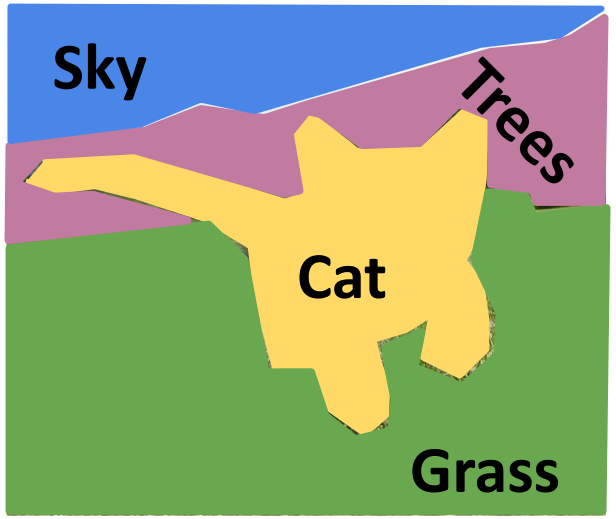
Label each pixel in the image with a category label

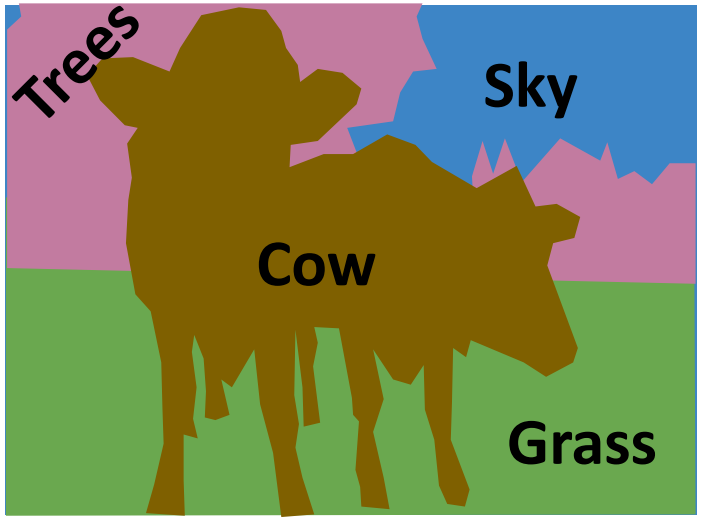
Don’t differentiate instances, only care about pixels



This image is CC0 public domain

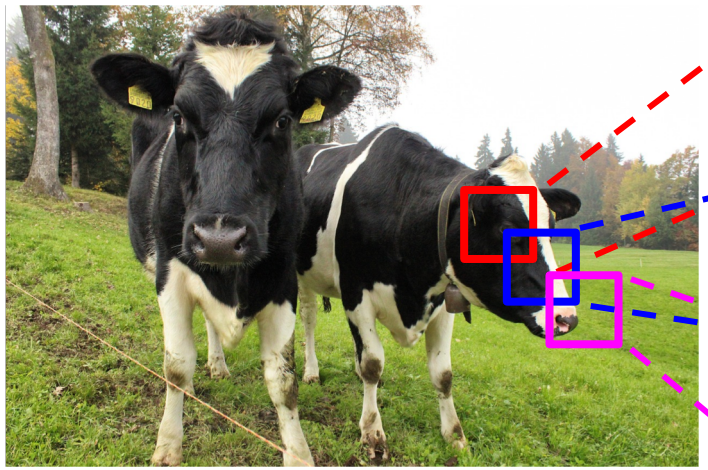




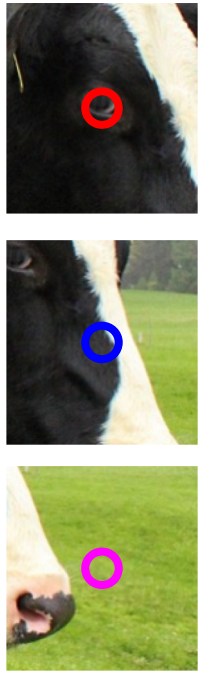


Semantic Segmentation Idea: Sliding Window

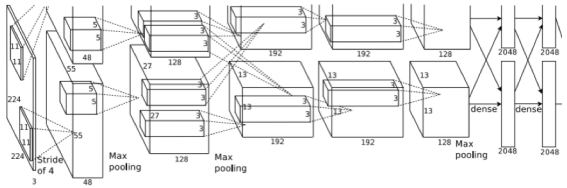
Full image



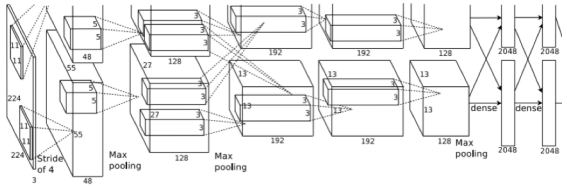
Extract patch



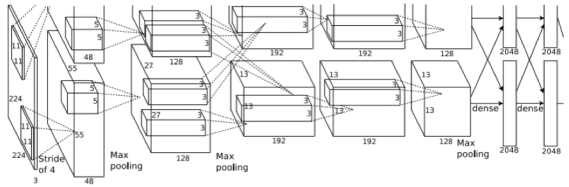
Classify center pixel with CNN



Cow



Cow



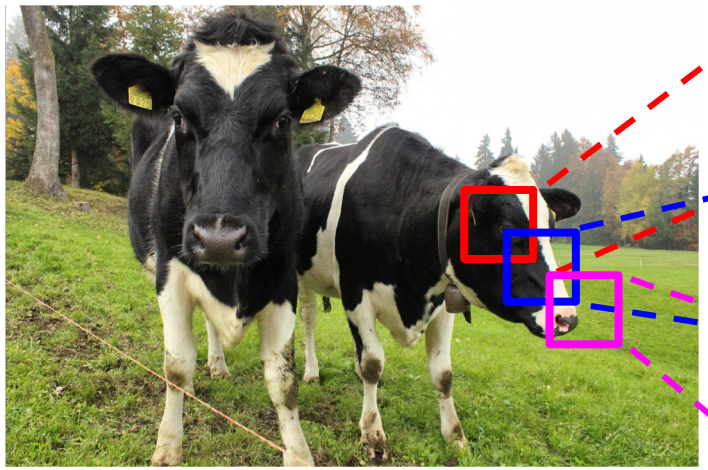
Grass

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

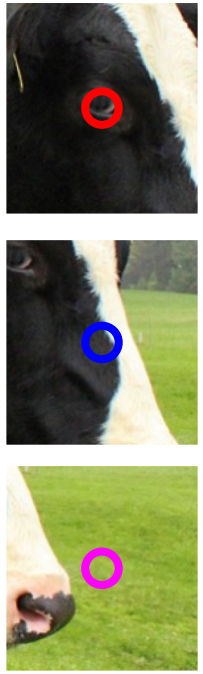
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Sliding Window

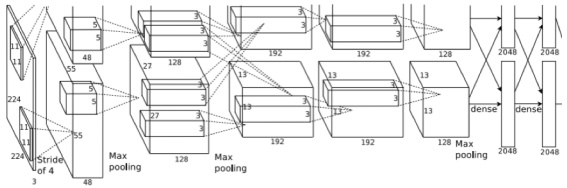
Full image



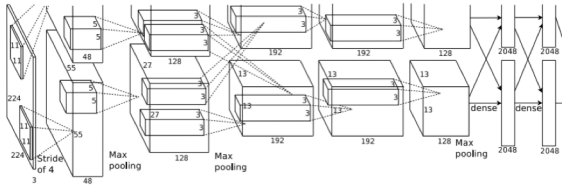
Extract patch



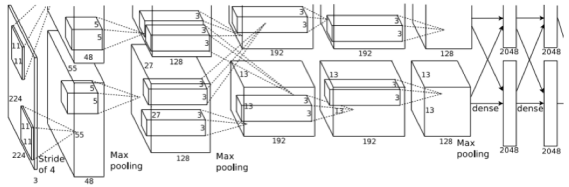
Classify center pixel with CNN



Cow



Cow



Grass

Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

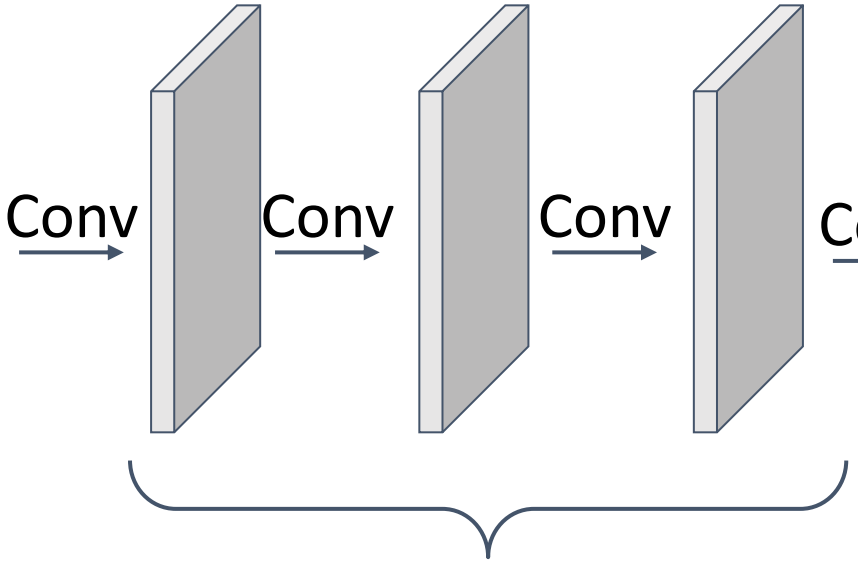
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation: Fully Convolutional Network

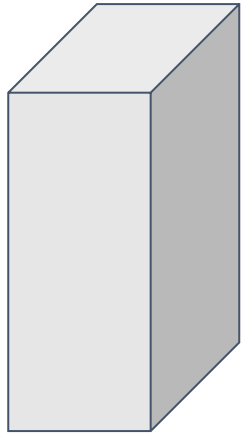
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W



Convolutions:
D x H x W



Scores:
C x H x W

argmax



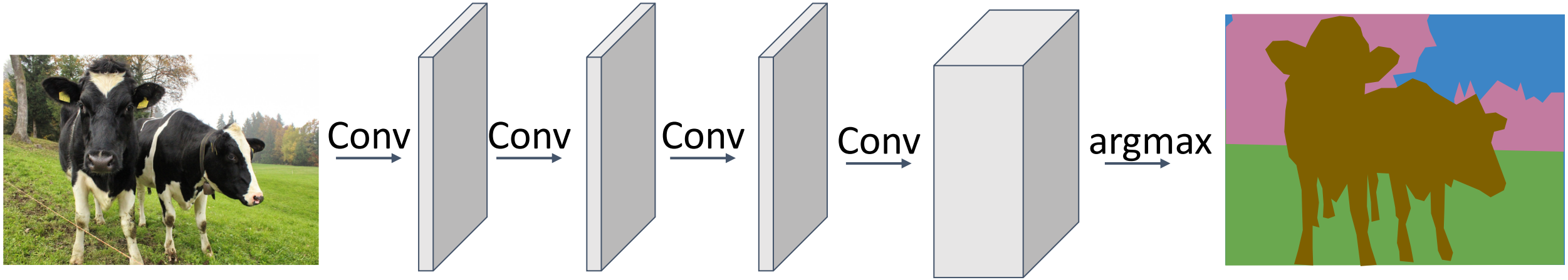
Predictions:
H x W

Loss function: Per-Pixel cross-entropy

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



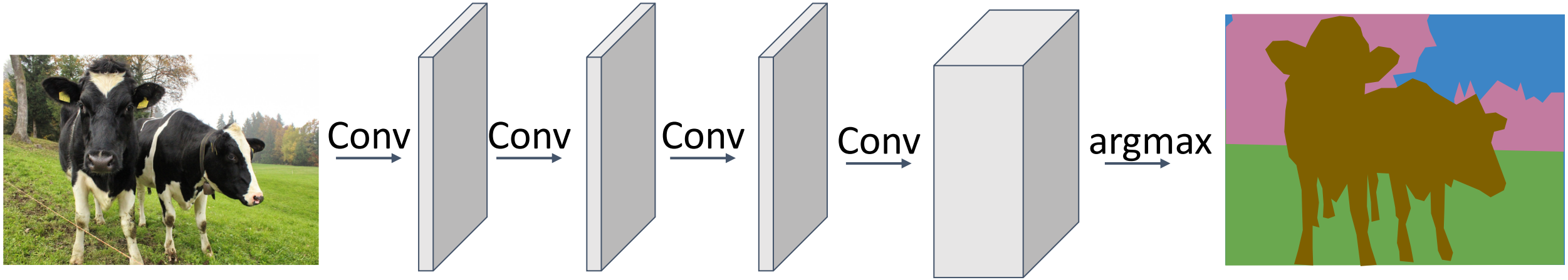
Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples

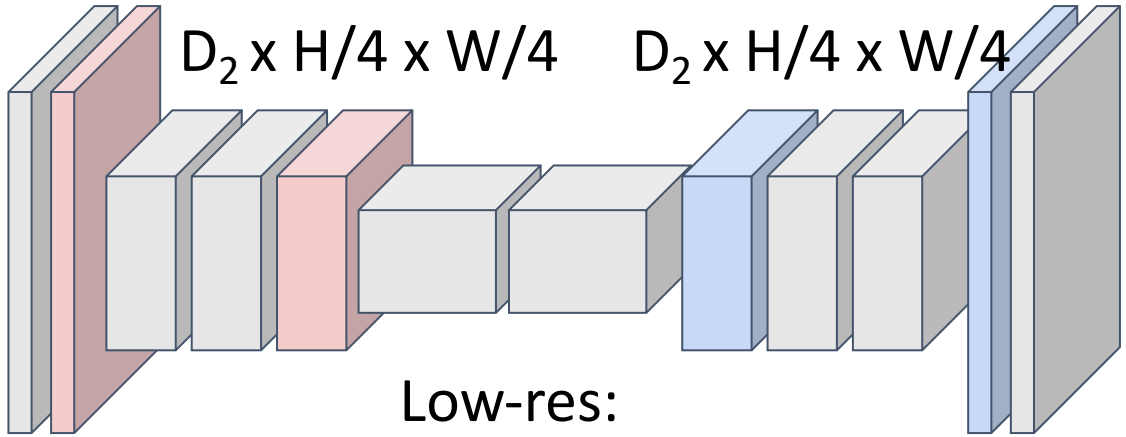
Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Input:
3 x H x W





Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation: Fully Convolutional Network

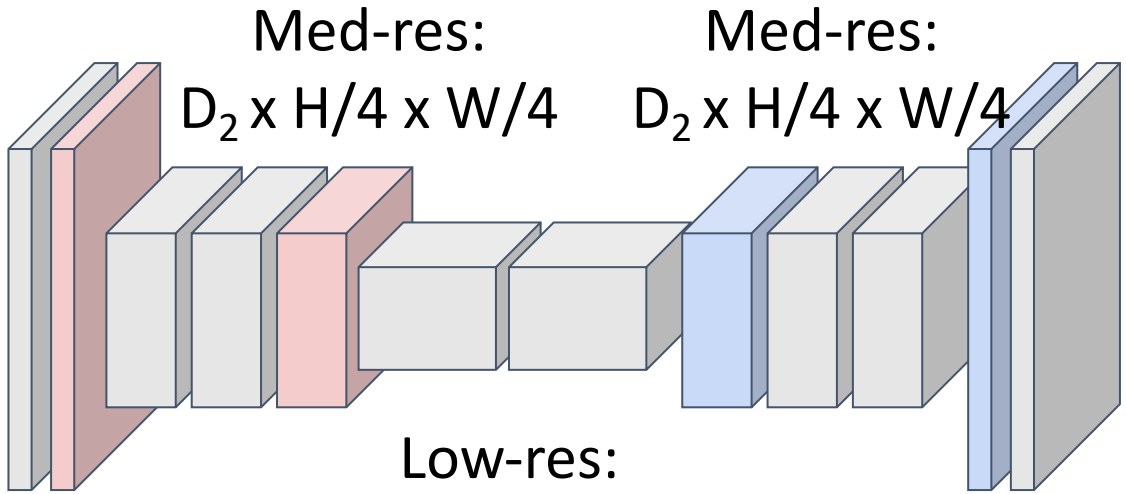
Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: ???



Input:
3 x H x W



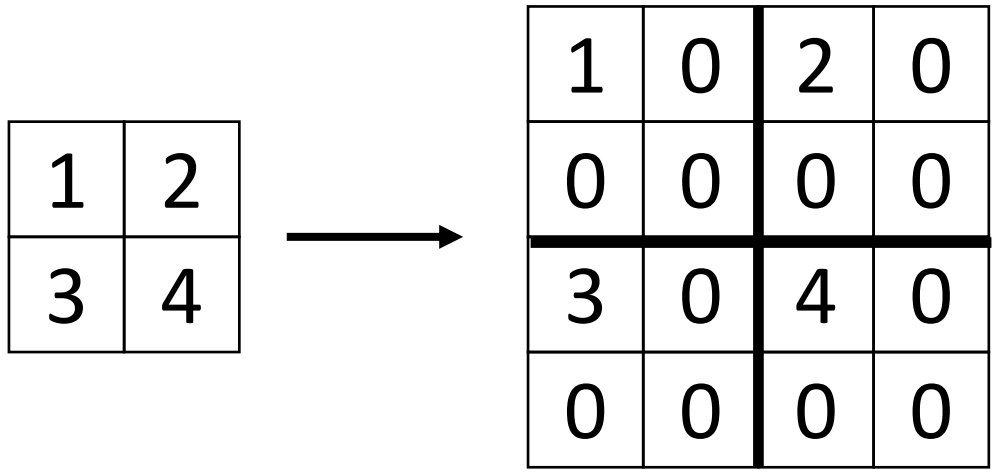


Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network Upsampling: “Unpooling”

Bed of Nails

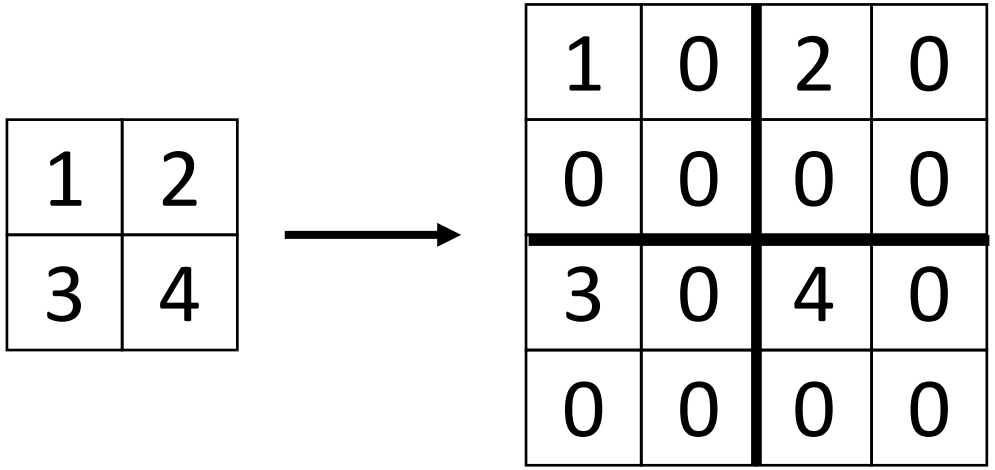


Input
C x 2 x 2

Output
C x 4 x 4

In-Network Upsampling: “Unpooling”

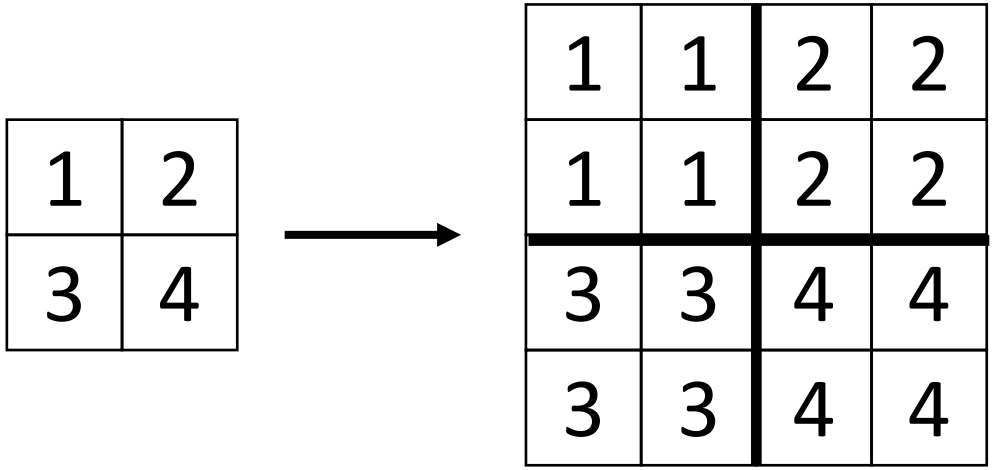
Bed of Nails



Input
C x 2 x 2

Output
C x 4 x 4

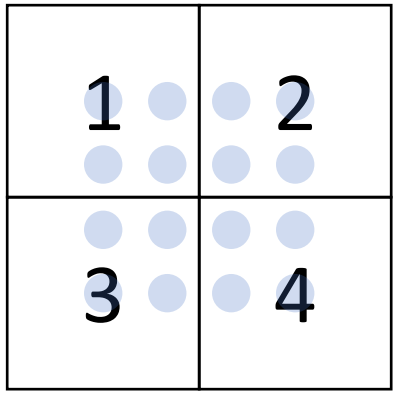
Nearest Neighbor



Input
C x 2 x 2

Output
C x 4 x 4

In-Network Upsampling: Bilinear Interpolation



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: C x 2 x 2

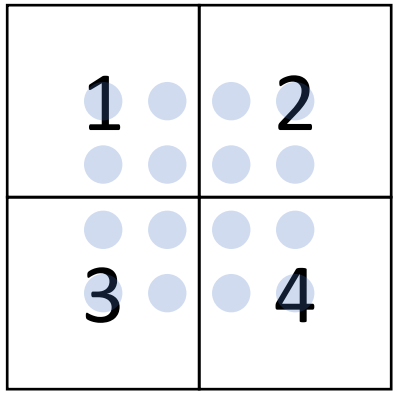
Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$$

$$i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\}$$
$$j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

In-Network Upsampling: Bicubic Interpolation



0.68	1.02	1.56	1.89
1.35	1.68	2.23	2.56
2.44	2.77	3.32	3.65
3.11	3.44	3.98	4.32

Input: C x 2 x 2

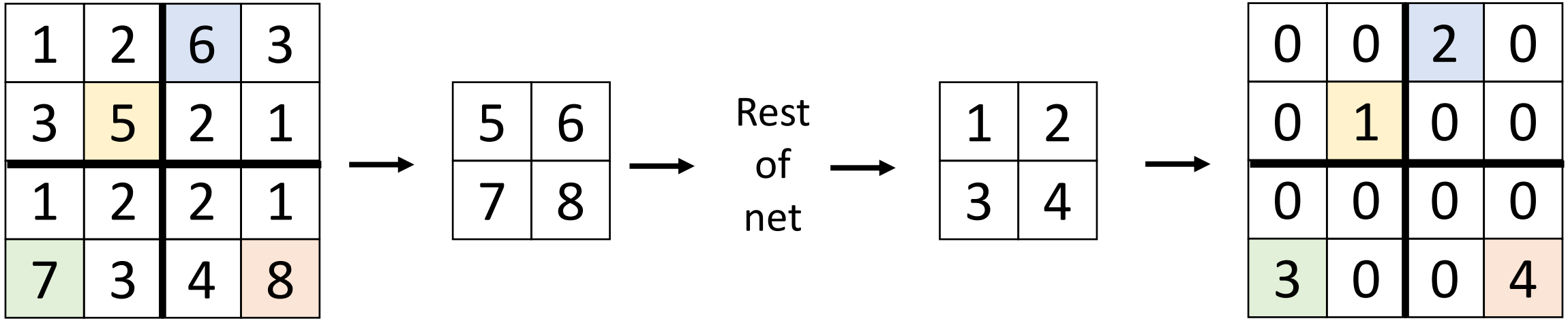
Output: C x 4 x 4

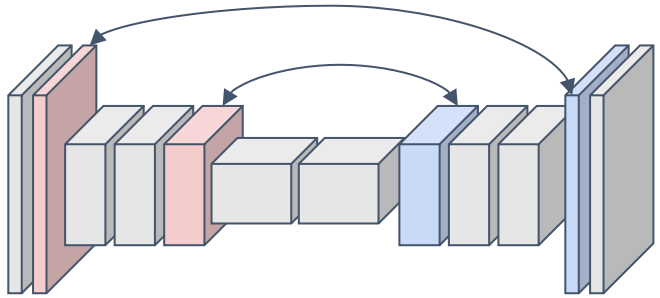
Use **three** closest neighbors in x and y to construct **cubic** approximations
(This is how we normally resize images!)

In-Network Upsampling: "Max Unpooling"

Max Pooling: Remember which position had the max

Max Unpooling: Place into remembered positions



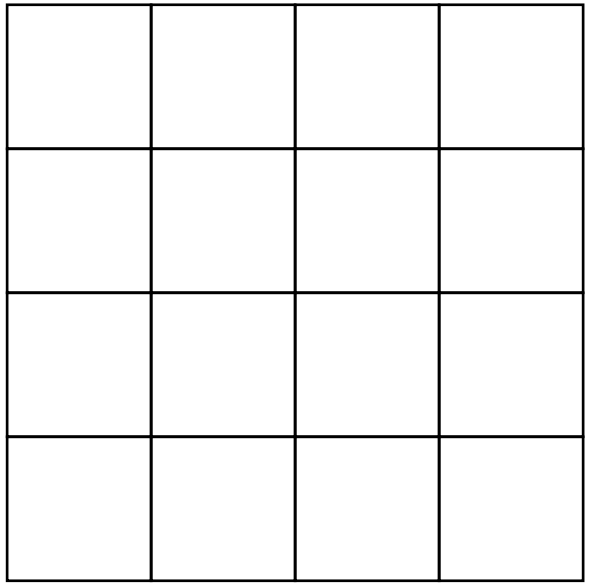


Pair each downsampling layer with an upsampling layer

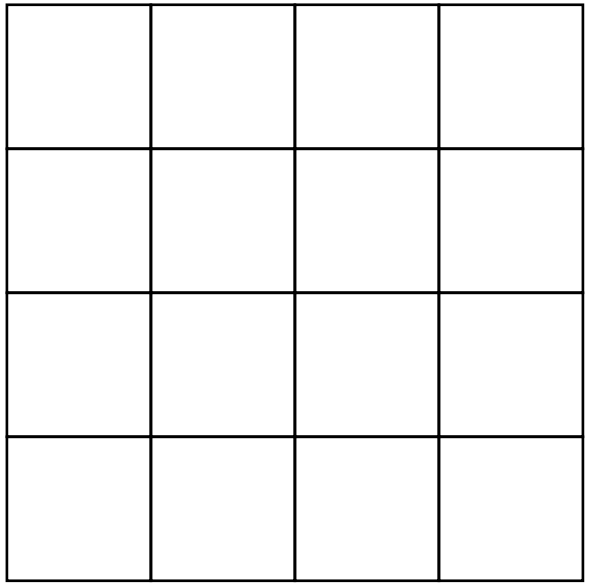
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1



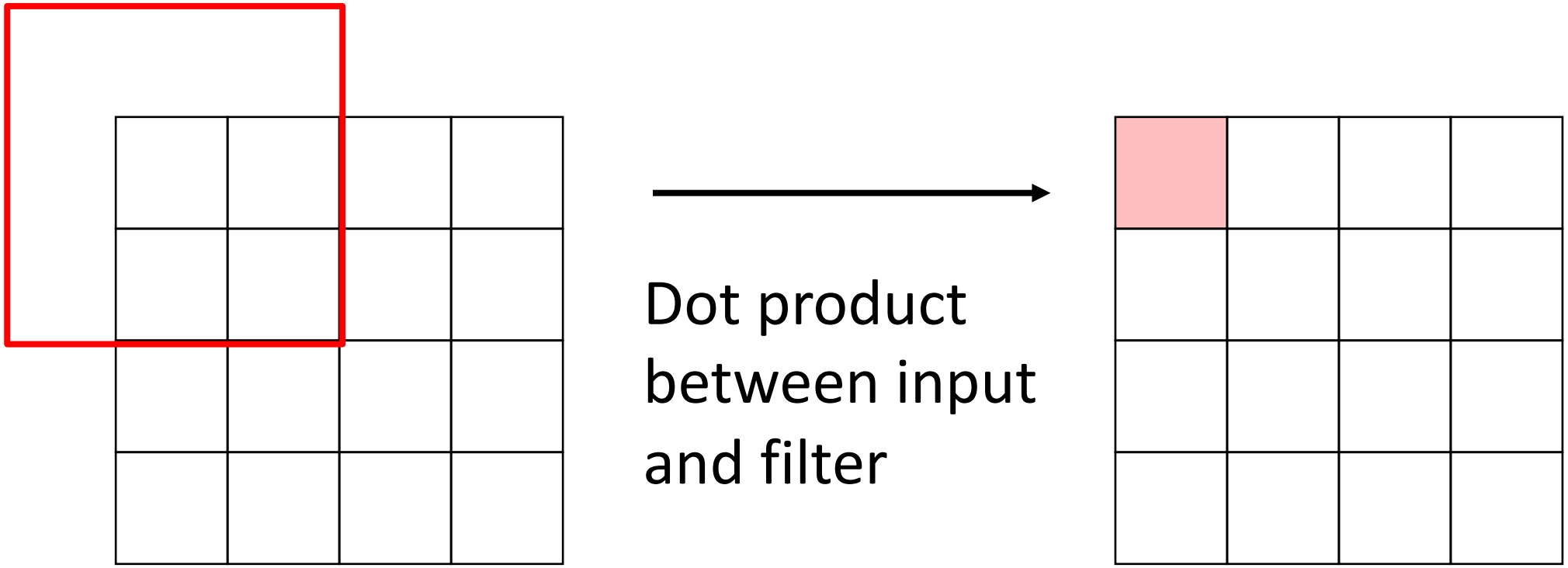
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1

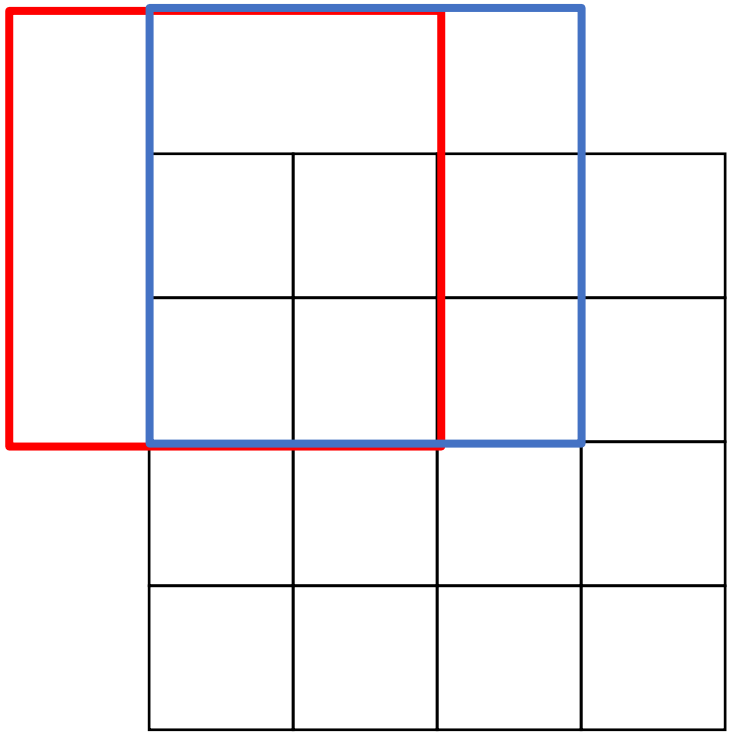


Input: 4 x 4

Output: 4 x 4

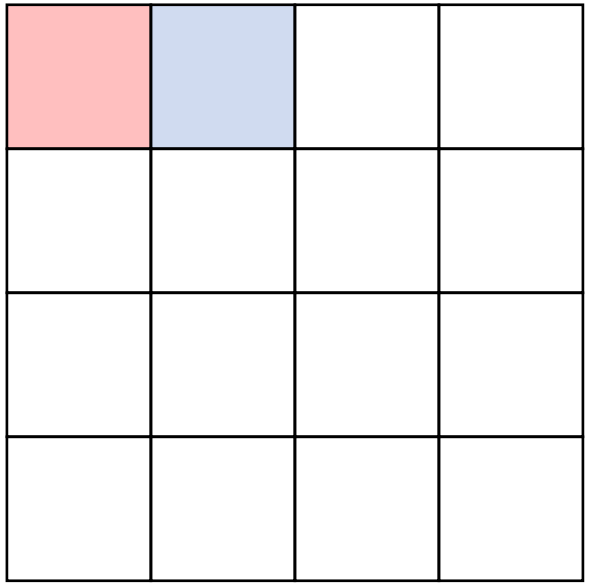
Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1



Input: 4 x 4

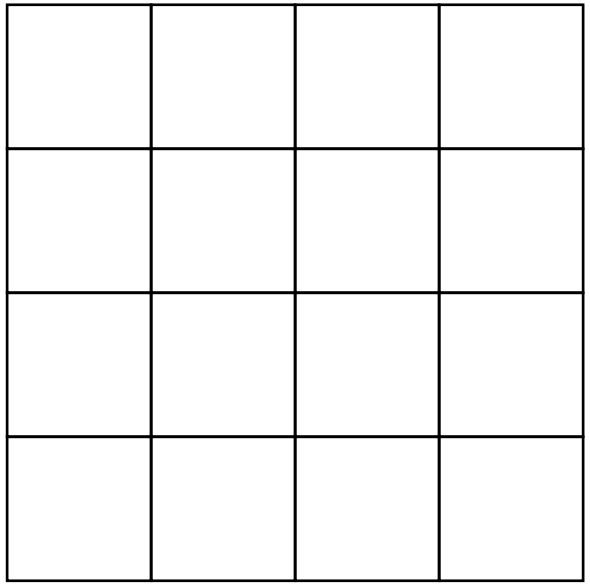
Dot product between input and filter



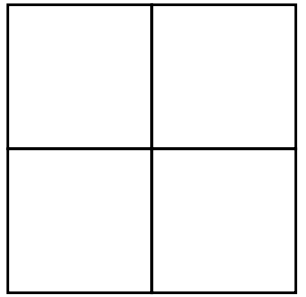
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



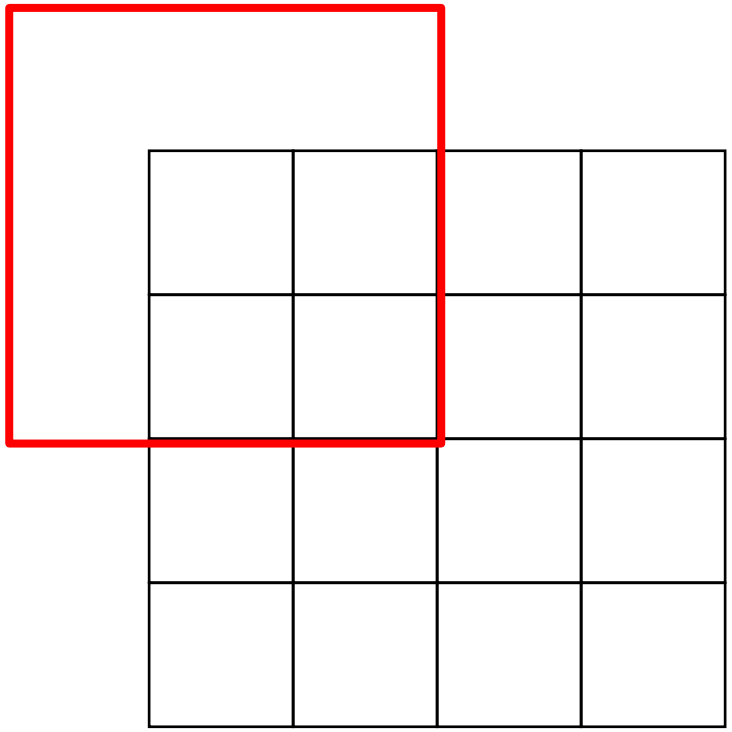
Input: 4 x 4



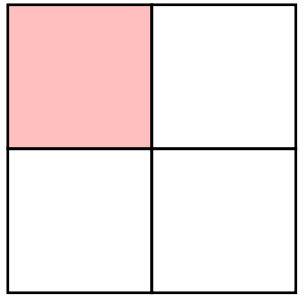
Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Dot product between input and filter

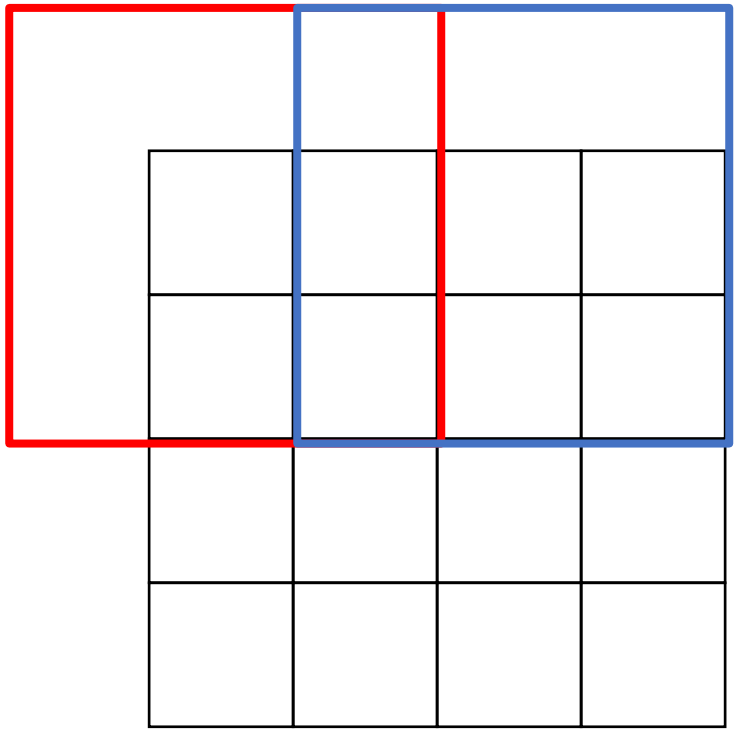


Input: 4 x 4

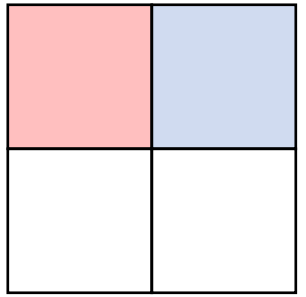
Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Dot product between input and filter

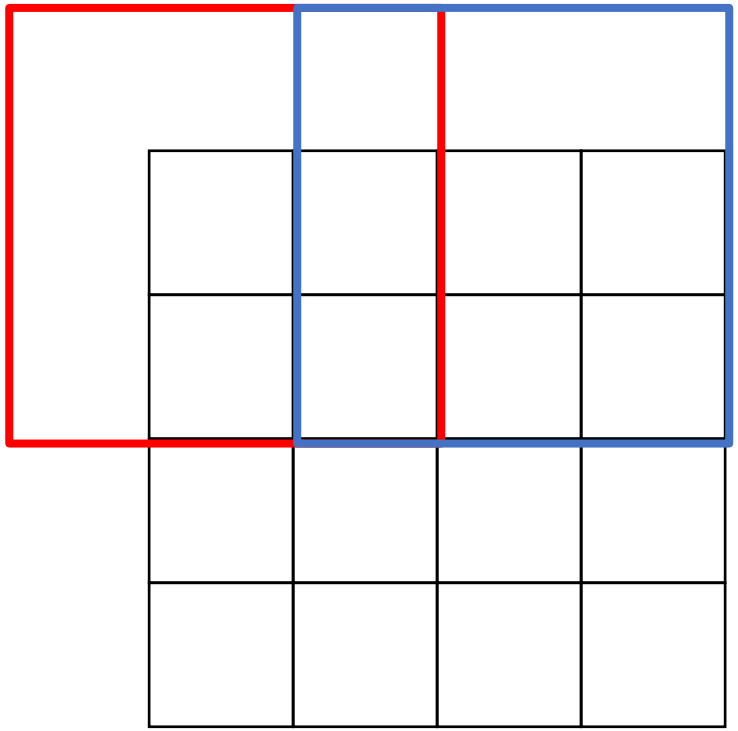


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transposed Convolution

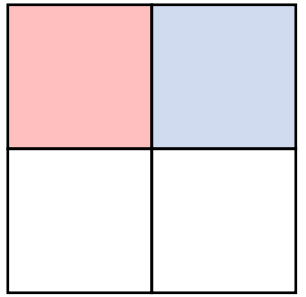
Recall: Normal 3 x 3 convolution, stride 2, pad 1



Convolution with stride > 1 is “Learnable Downsampling”

Can we use stride < 1 for “Learnable Upsampling”?

Dot product between input and filter

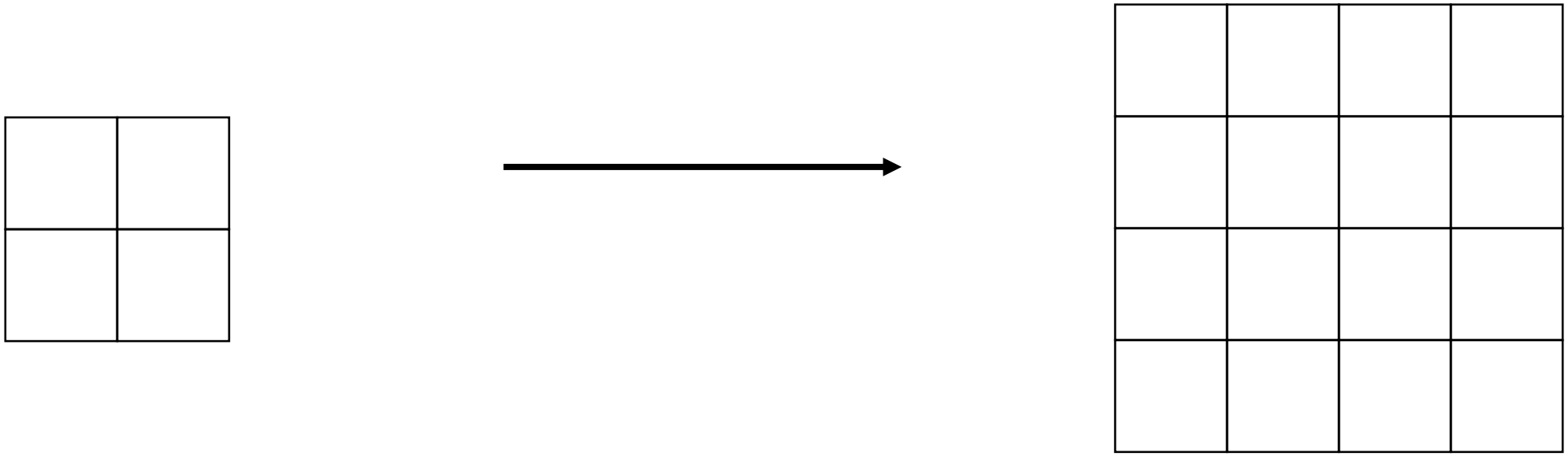


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transposed Convolution

3 x 3 **convolution transpose**, stride 2

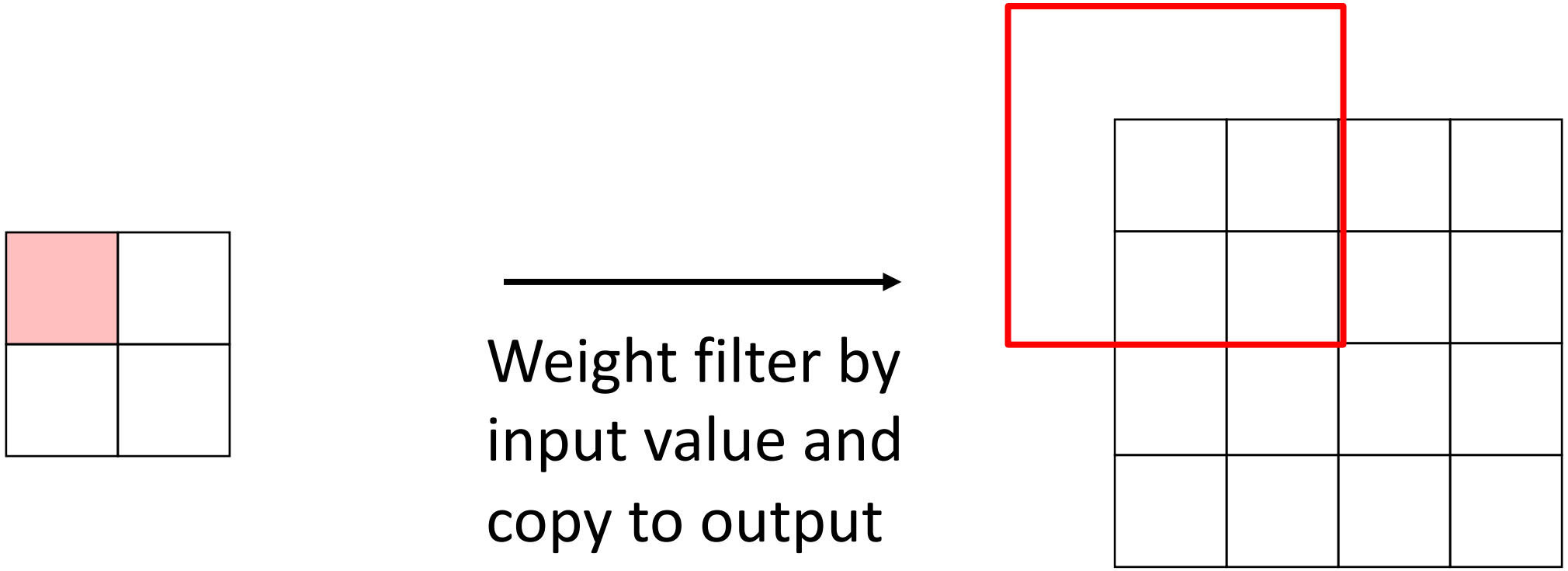


Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

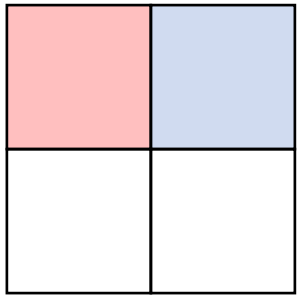
3 x 3 **convolution transpose**, stride 2



Learnable Upsampling: Transposed Convolution

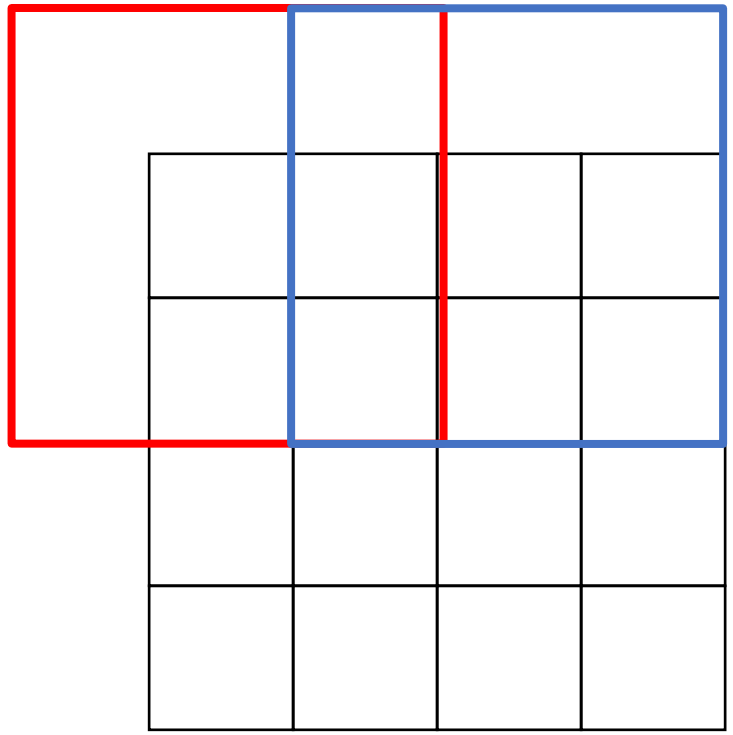
3 x 3 **convolution transpose**, stride 2

Filter moves 2 pixels in output
for every 1 pixel in input



Input: 2 x 2

Weight filter by
input value and
copy to output

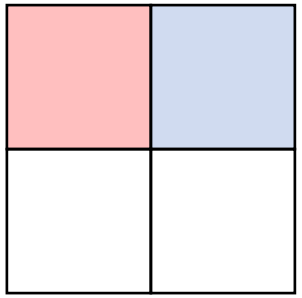


Output: 4 x 4

Learnable Upsampling: Transposed Convolution

3 x 3 **convolution transpose**, stride 2

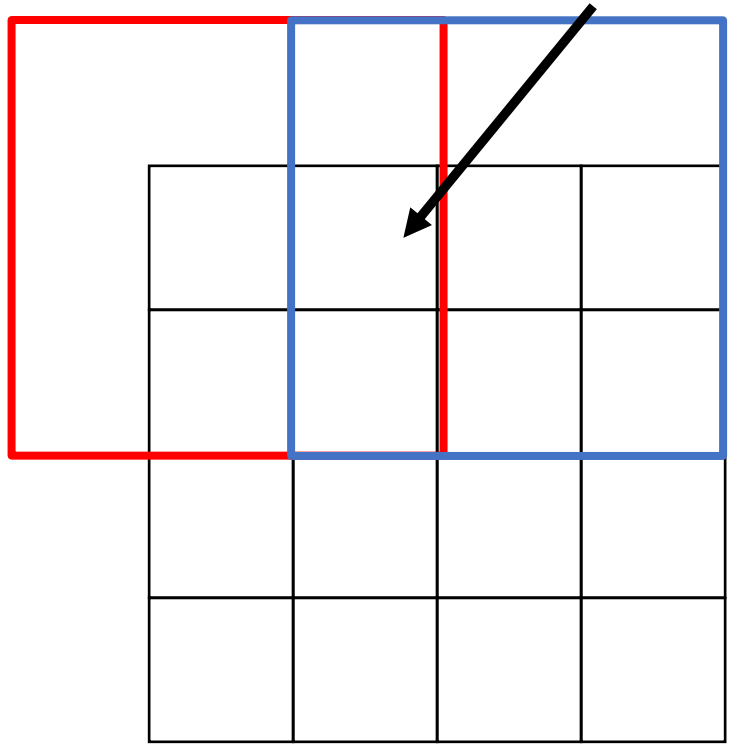
Filter moves 2 pixels in output
for every 1 pixel in input



Input: 2 x 2

Weight filter by input value and copy to output

Sum where output overlaps



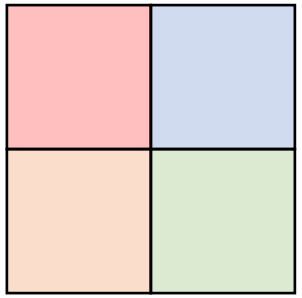
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

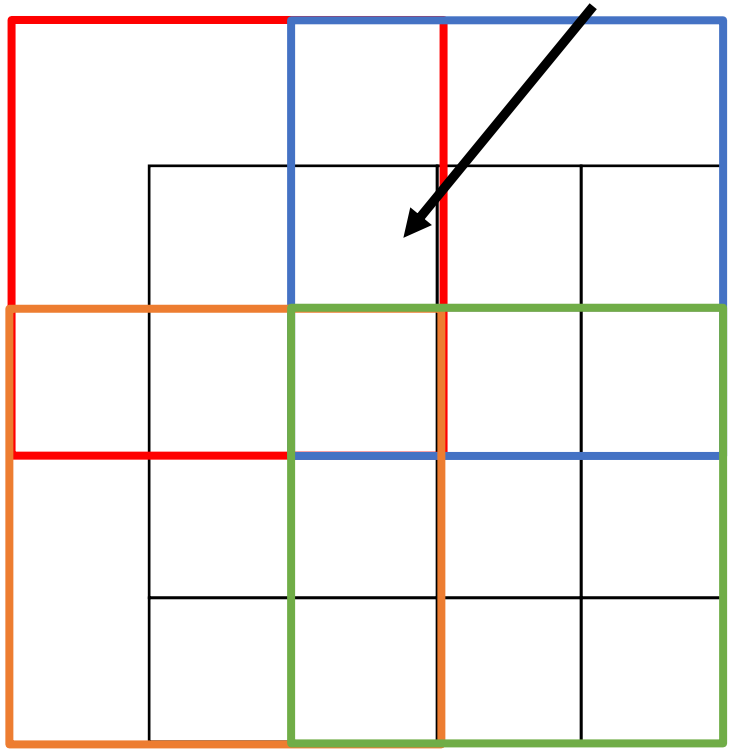
3 x 3 **convolution transpose**, stride 2

Sum where output overlaps

This gives 5x5 output – need to trim one pixel from top and left to give 4x4 output



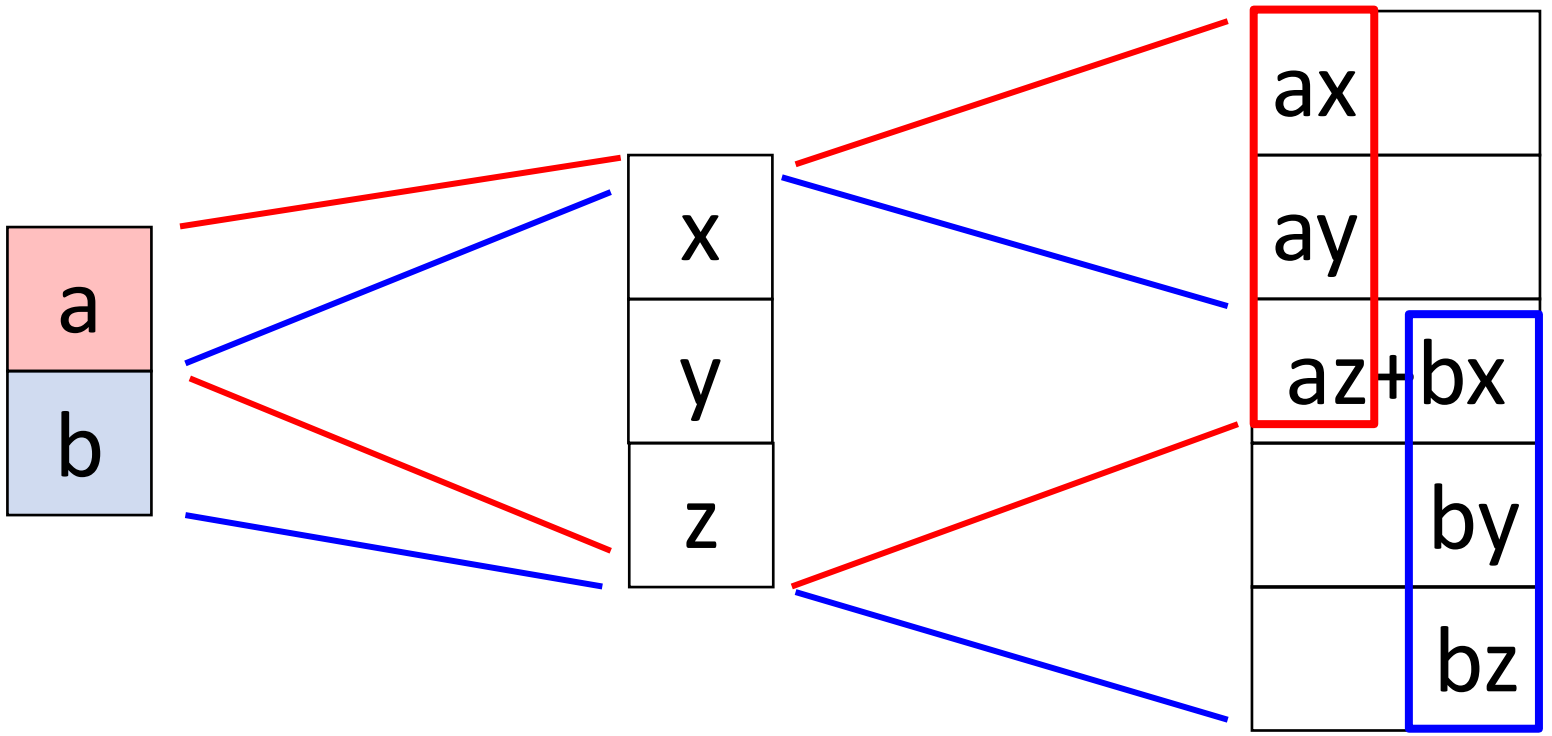
Weight filter by input value and copy to output



Input: 2 x 2

Output: 4 x 4

Transposed Convolution: 1D example

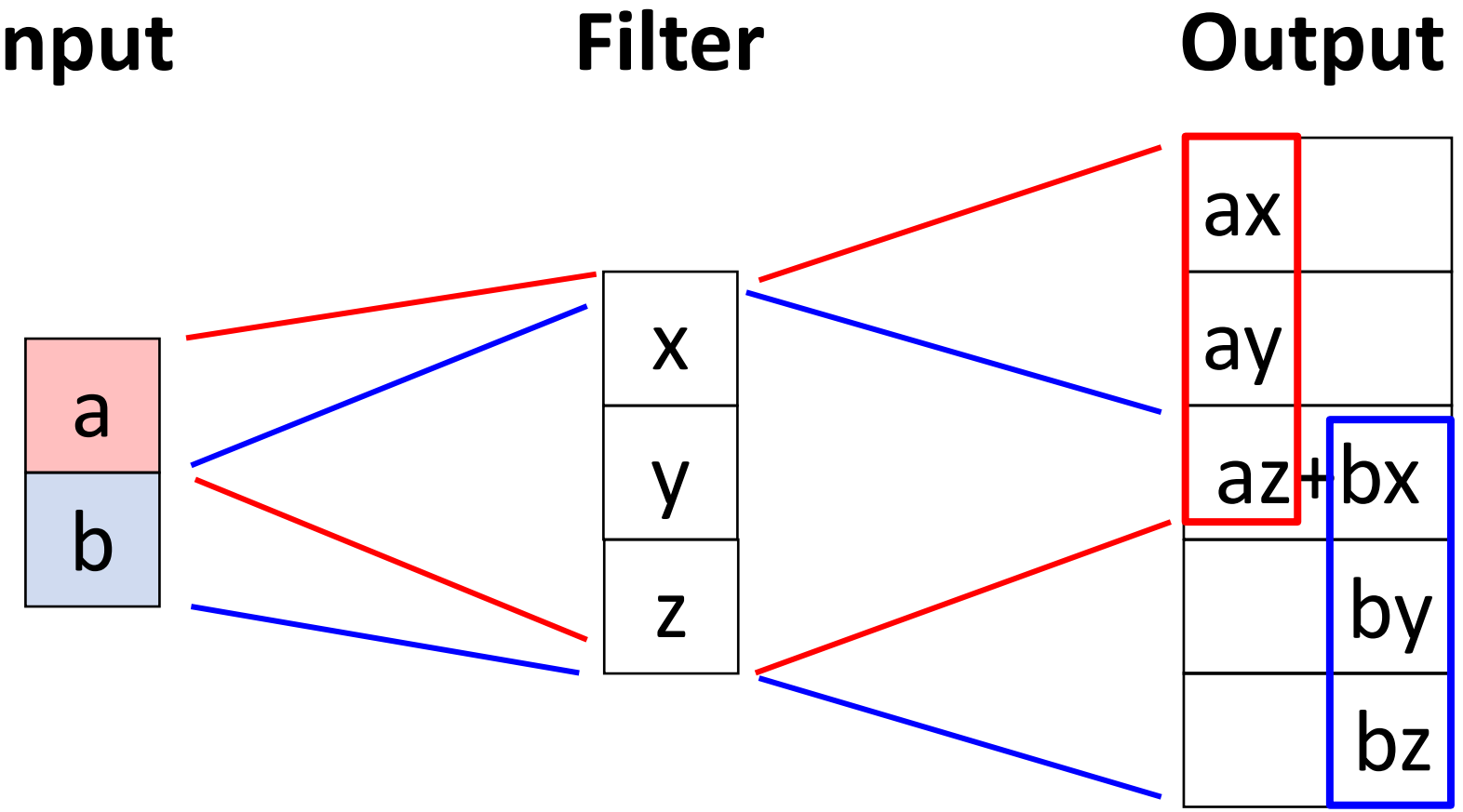


Output has copies of filter weighted by input

Stride 2: Move 2 pixels output for each pixel in input

Sum at overlaps

Transposed Convolution: 1D example



This has many names:

- ~~Deconvolution (bad)!~~
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution
- Transposed Convolution (best name)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, transposed conv is just a regular conv (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & 0 & x & y & x & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & 0 & x & y & x & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, transposed convolution cannot be expressed as normal conv

Semantic Segmentation: Fully Convolutional Network

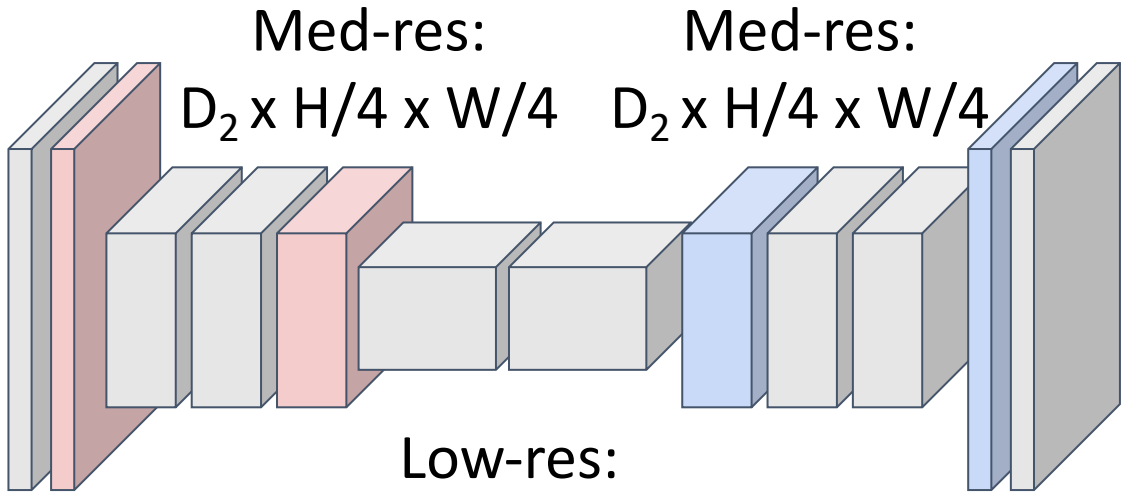
Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: linterpolation, transposed conv



Input:
3 x H x W





Predictions:
H x W

Loss function: Per-Pixel cross-entropy

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

U-Nets

- Skip connections between encoding and decoding stages
- Widely used in image denoising, inpainting, flow, stereo, …

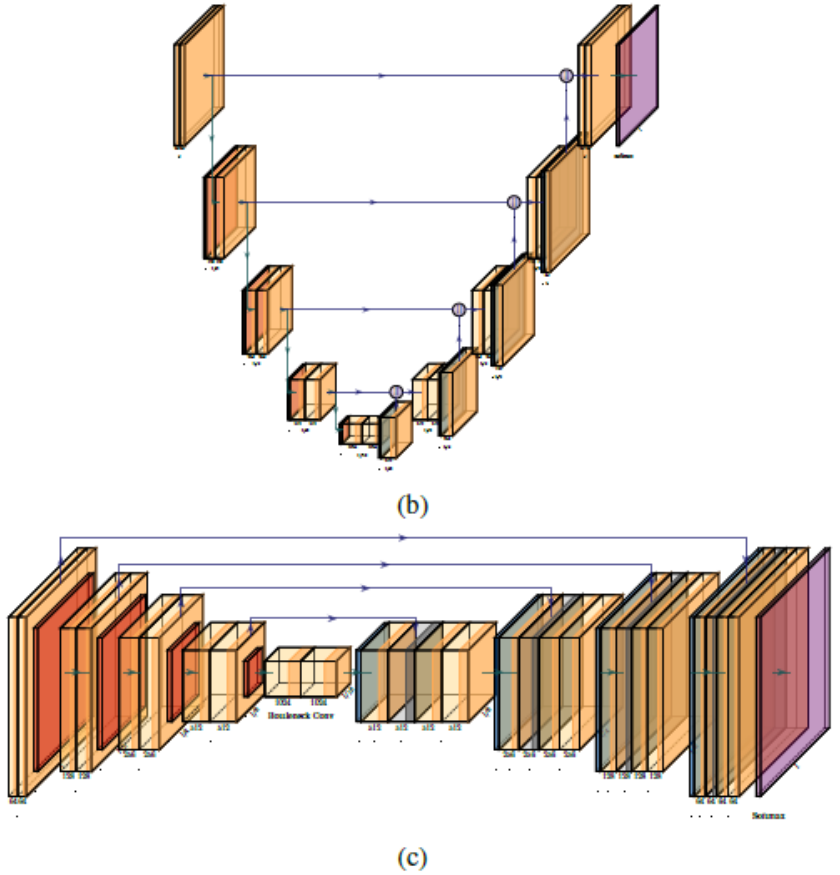
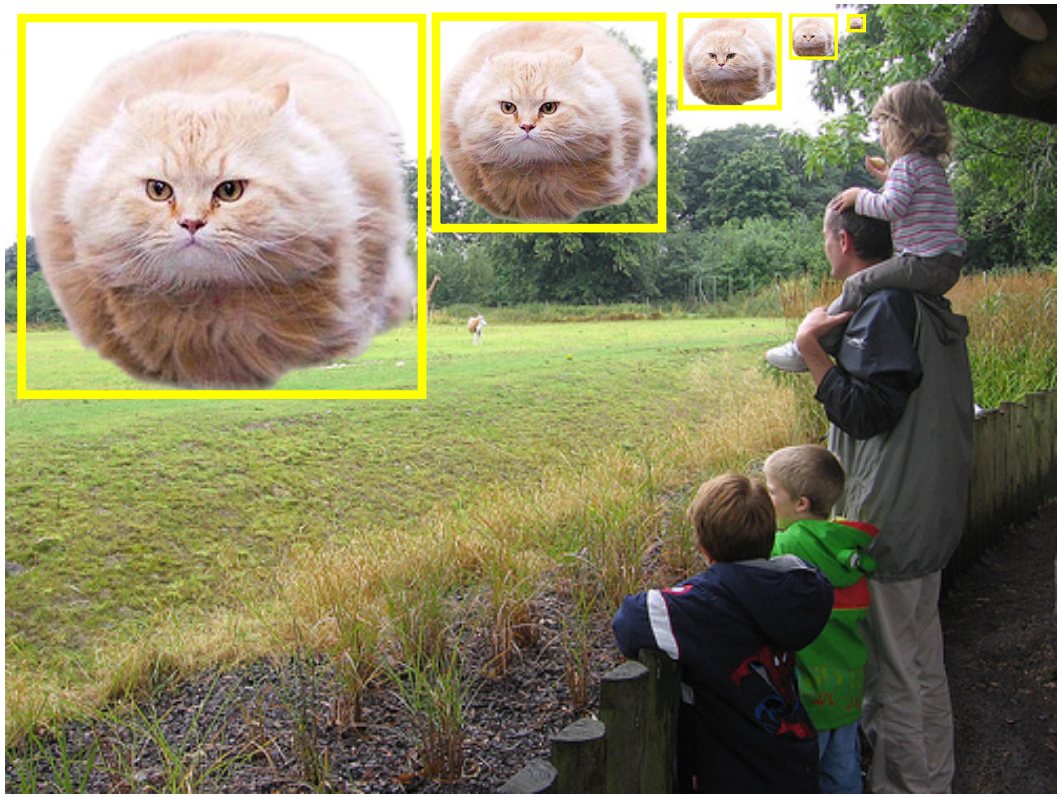


Figure 5.41 *(a) The deconvolution network of Noh, Hong, and Han (2015) © 2015 IEEE and (b–c) the U-Net of Ronneberger, Fischer, and Brox (2015), redrawn using the PlotNeuralNet LaTeX package by Matt Dietke. In addition to the fine-to-coarse-to-fine bottleneck used in (a), the U-Net also has skip connections between encoding and decoding layers at the same resolution.*

Feature Pyramid Network



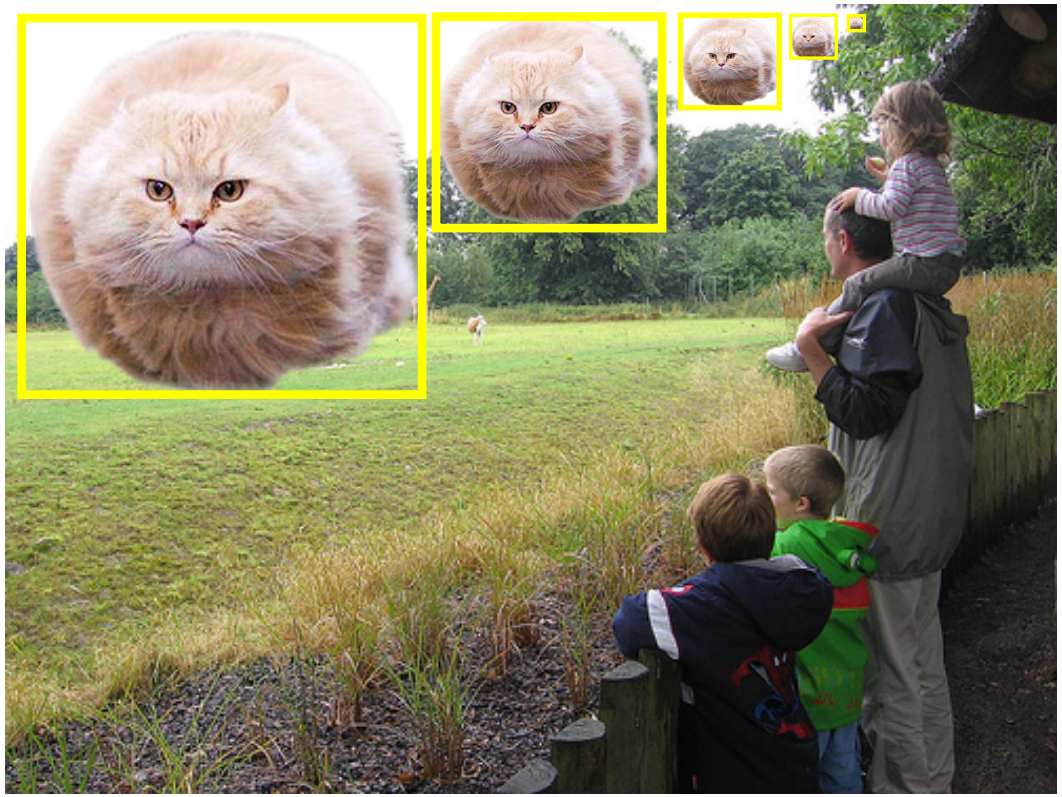


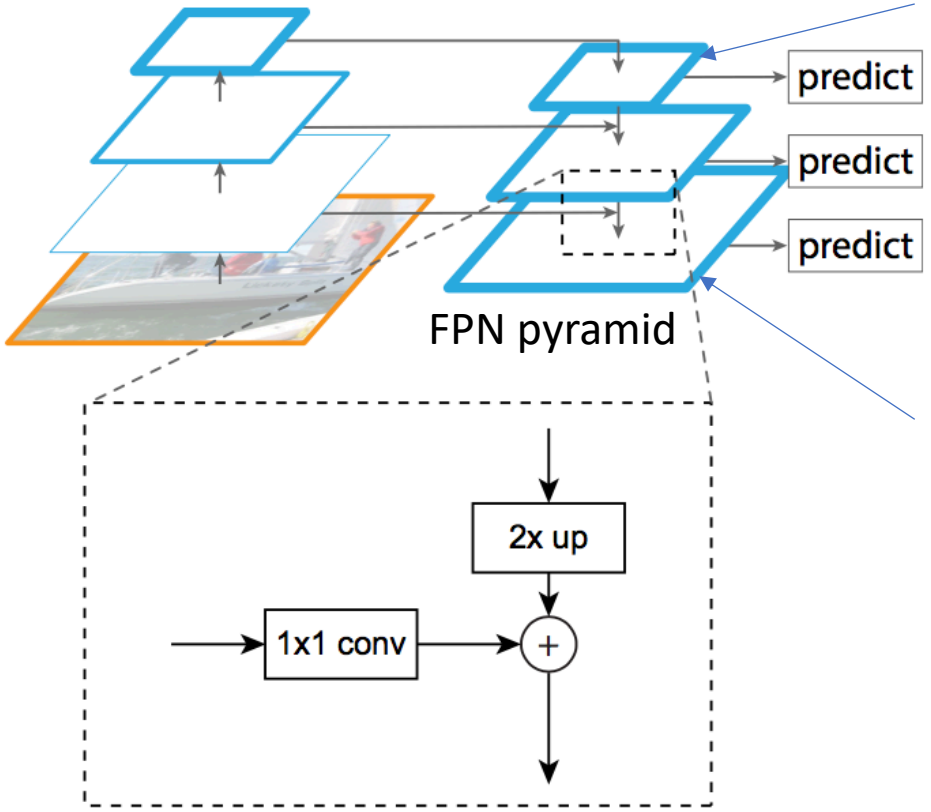
(d) Feature Pyramid Network

Top-down enrichment of high-res features – *fast, less suboptimal*

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

No Compromise on Feature Quality, Still Fast



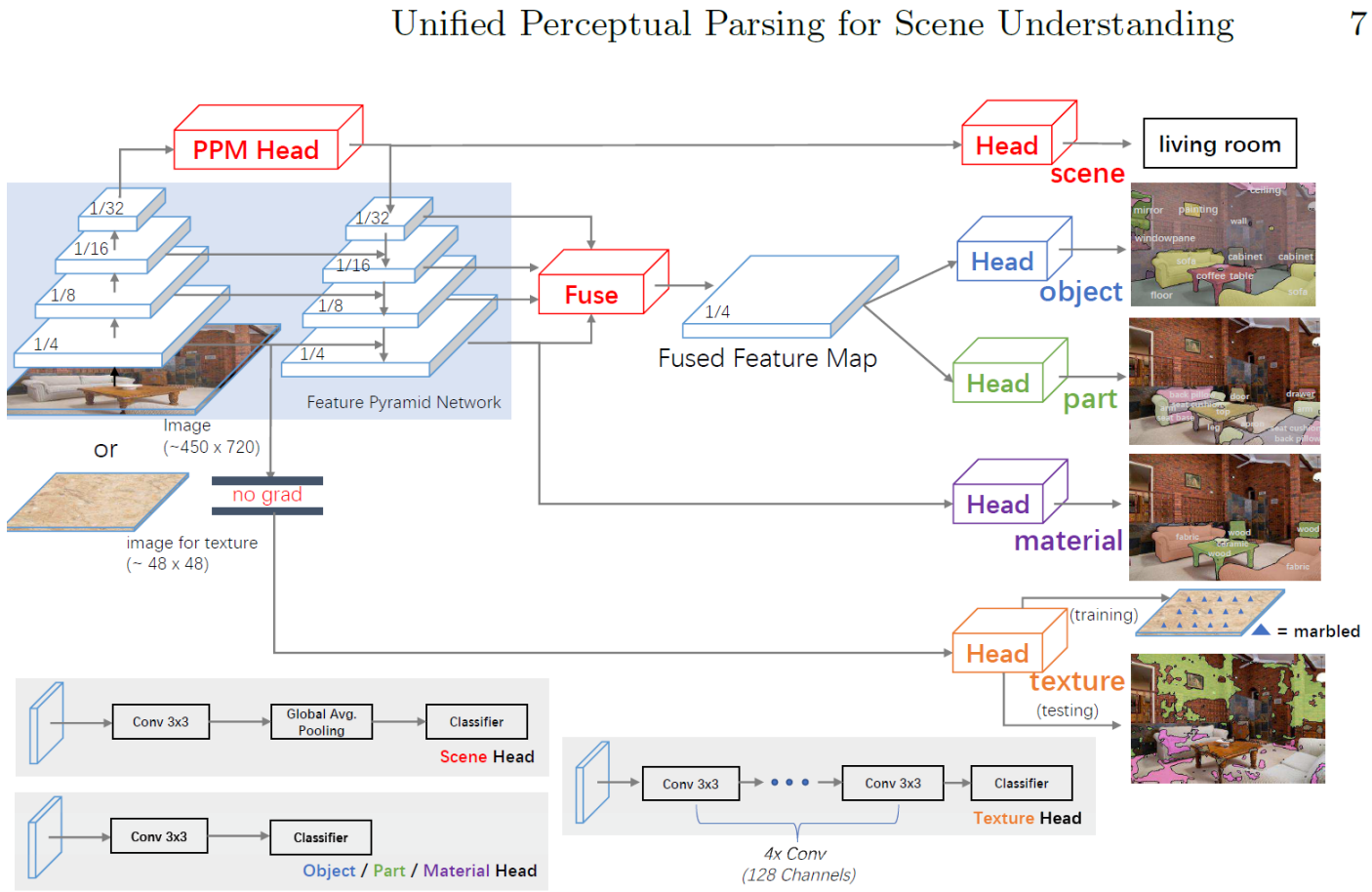


Low resolution,
Strong features

High resolution,
Strong features

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

UPerNet: FPN + fusion



Application: monocular depth estimation

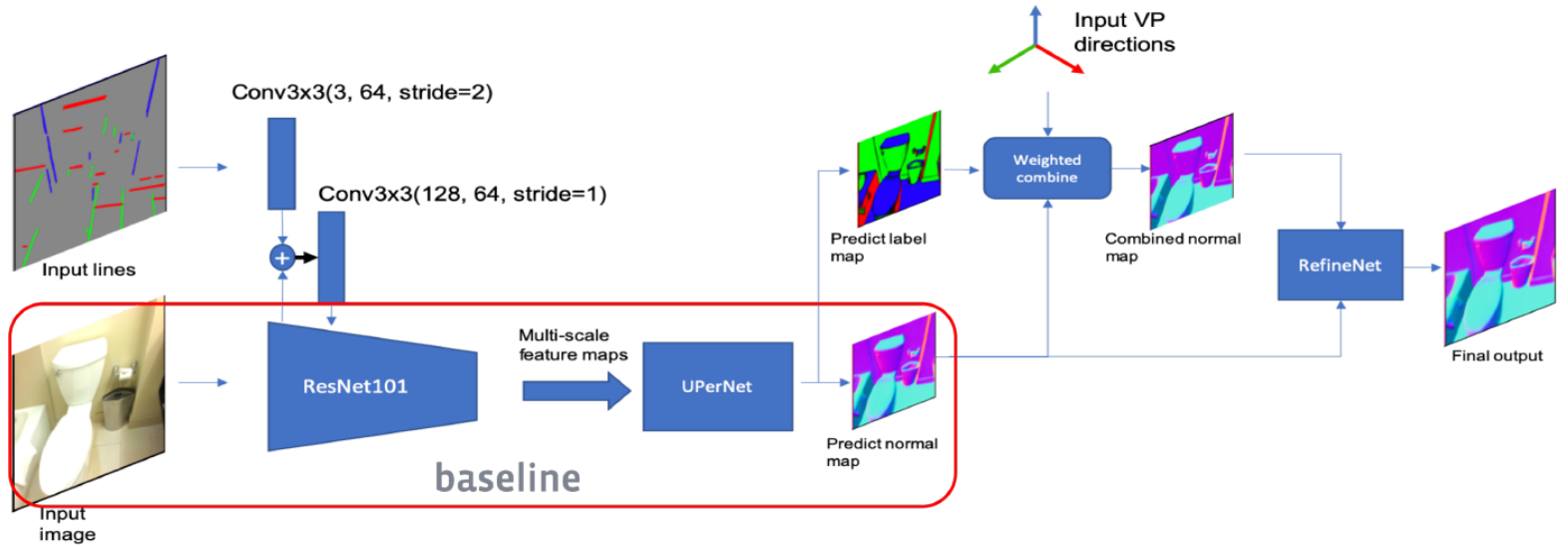
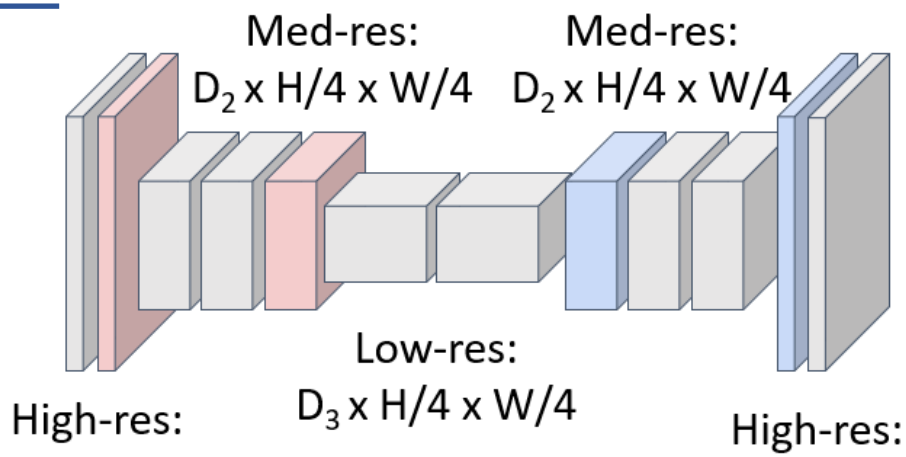


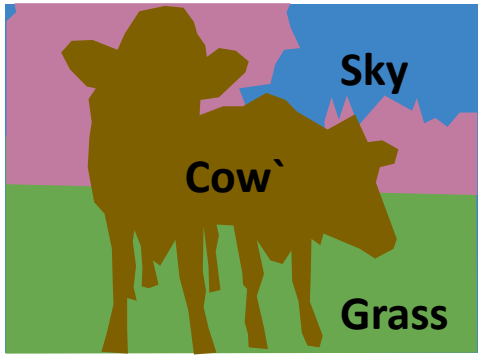
Figure 2. The full pipeline of our proposed model. The network takes an RGB image and a Manhattan line map as input, and produces a Manhattan label map and a raw normal prediction as intermediate output. These intermediate outputs are then combined with the analytically computed dominant vanishing points to generate a "combined normal map". This operation is differentiable. Finally, the combined and raw normal maps are fused through a refinement network to produce the final normal prediction.

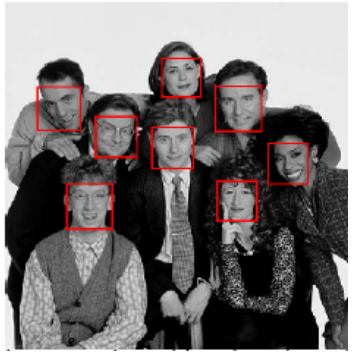
VPLNet: Deep Single View Normal Estimation with Vanishing Points and Lines
accepted to CVPR'2020

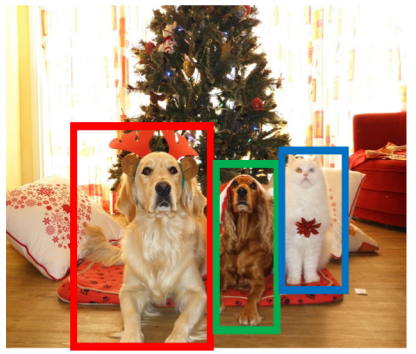
Segmentation and detection

- Adversarial examples
- Bottlenecks and U-Nets
- Segmentation
- **Face detection**
- Object detection









Classic face detection

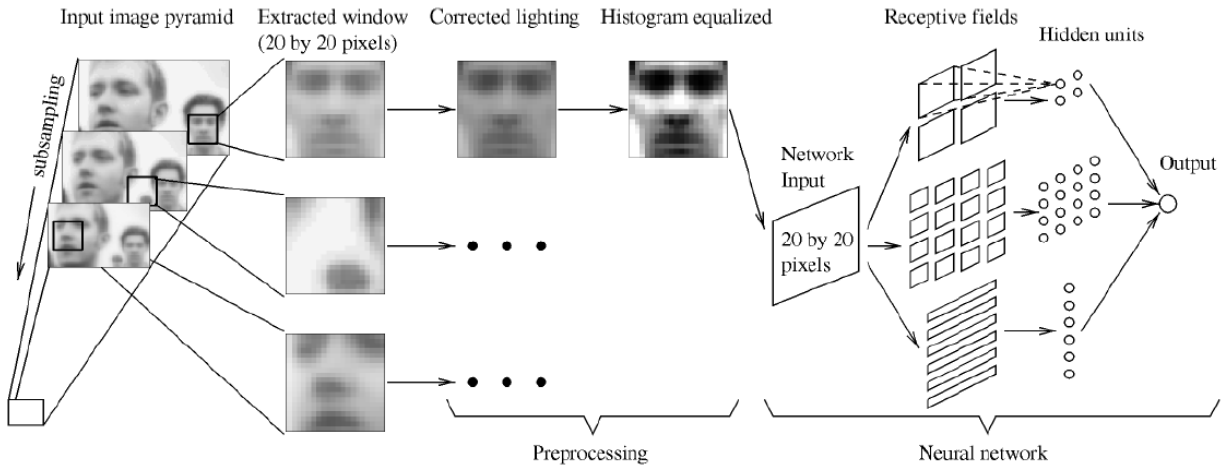


Figure 6.27 *A neural network for face detection (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE. Overlapping patches are extracted from different levels of a pyramid and then pre-processed as shown in Figure 6.25b. A three-layer neural network is then used to detect likely face locations.*

Can you find the one false positive?



Face Recognition and Detection



The “Margaret Thatcher Illusion”, by Peter Thompson

Computer Vision

CSE576, Spring 2008

Richard Szeliski

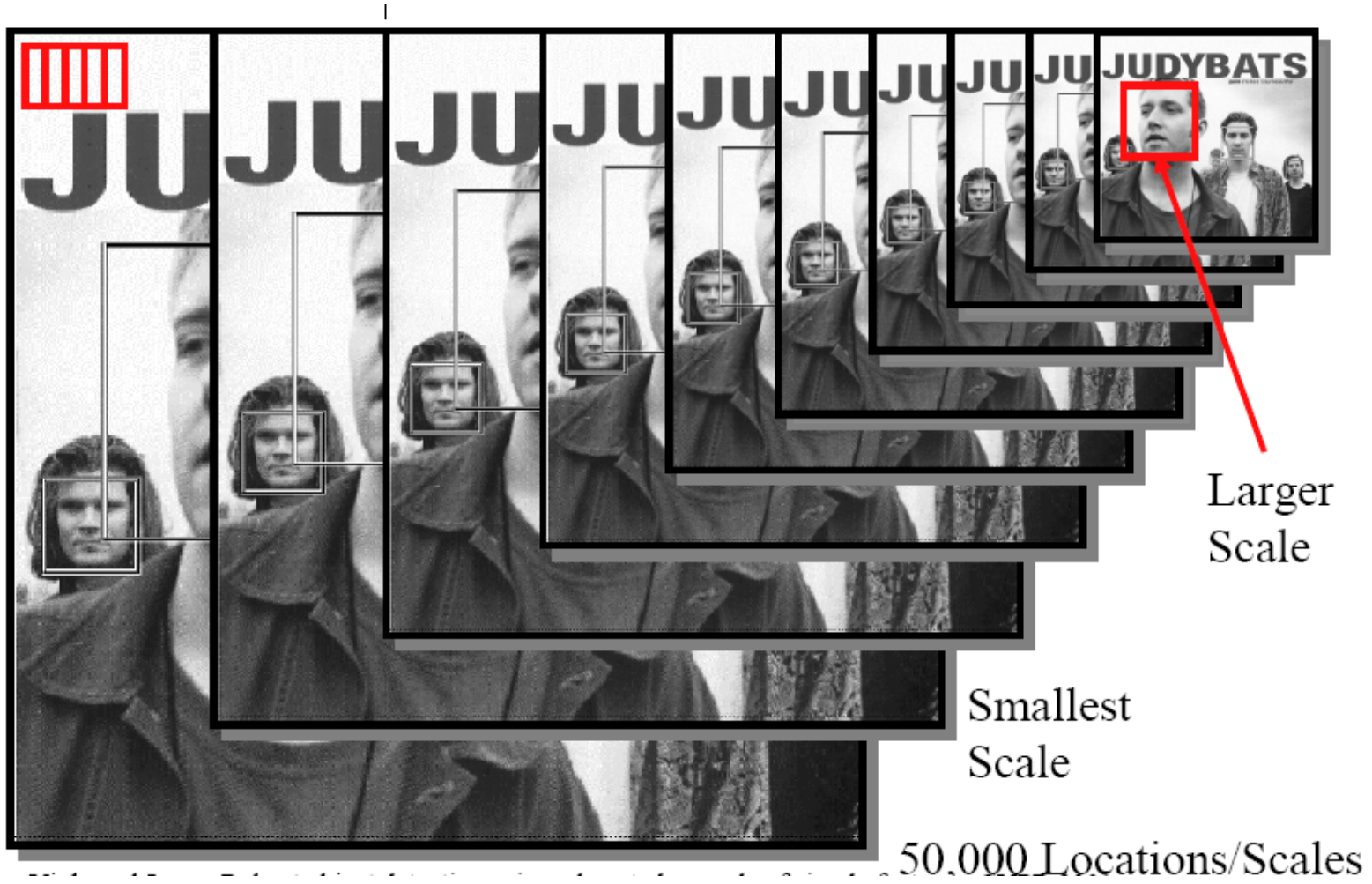
Robust real-time face detection

Paul A. Viola and Michael J. Jones

Intl. J. Computer Vision
57(2), 137–154, 2004
(originally in CVPR'2001)

(slides adapted from Bill Freeman, MIT 6.869, April 2005)

Scan classifier over locs. & scales



Learn classifier from data

- Training Data
- 5000 faces (frontal)
- 10^8 non faces
- Faces are normalized
  - Scale, translation
- Many variations
- Across individuals
- Illumination
- Pose (rotation both in plane and out)



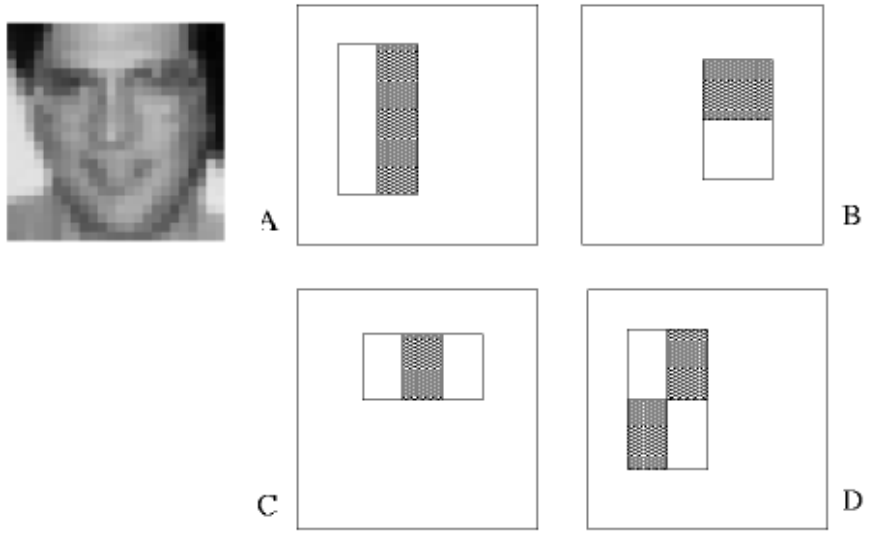
Characteristics of algorithm

- Feature set (…is huge about 16M features)
- Efficient feature selection using AdaBoost
- New image representation: Integral Image
- Cascaded Classifier for rapid detection

➢Fastest known face detector for gray scale images (2001)

Image features

- "Rectangle filters"
  - Similar to Haar wavelets
- Differences between sums of pixels in adjacent rectangles



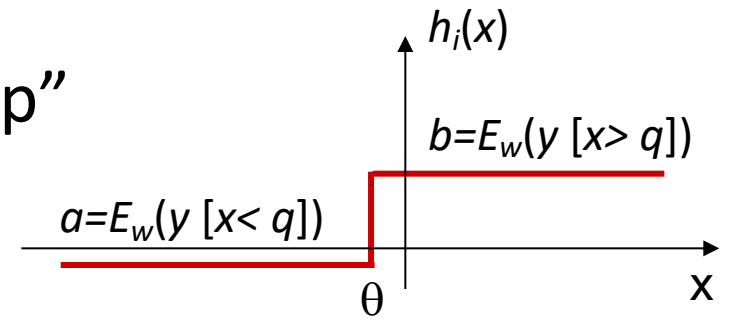
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

Constructing the classifier

- Perceptron yields a sufficiently powerful classifier

$$C(x) = \theta\left(\sum_i \alpha_i h_i(x) + b\right)$$

- Use AdaBoost to efficiently choose best features
- add a new $h_i(x)$ at each round
- each $h_i(x_k)$ is a "decision stump"



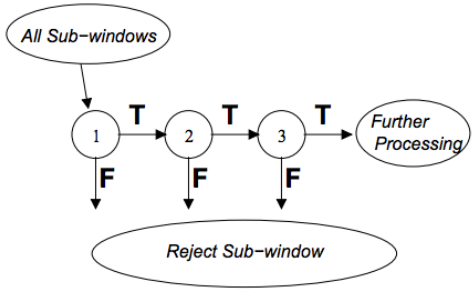
Good reference on boosting

- Friedman, J., Hastie, T. and Tibshirani, R. *Additive Logistic Regression: a Statistical View of Boosting*

  http://www-stat.stanford.edu/~hastie/Papers/boost.ps

- *"We show that boosting fits an additive logistic regression model by stagewise optimization of a criterion very similar to the log-likelihood, and present likelihood based alternatives. We also propose a multi-logit boosting procedure which appears to have advantages over other methods proposed so far."*

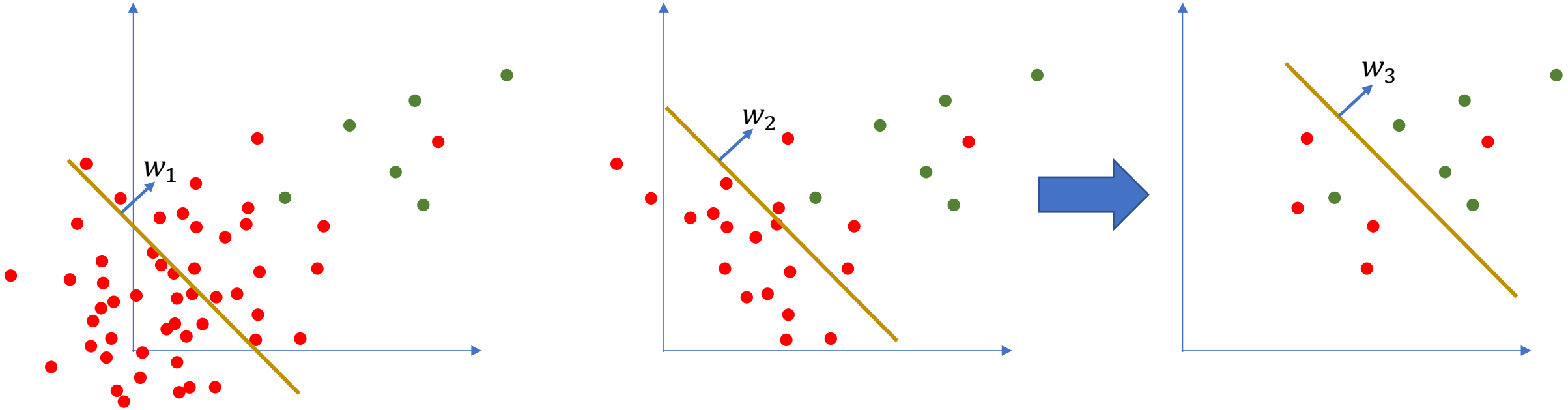
Handling Imbalance with Cascades



(slide courtesy of Ross Girshick)

eliminate easy background step-by-step, leading to a …

… balanced problem



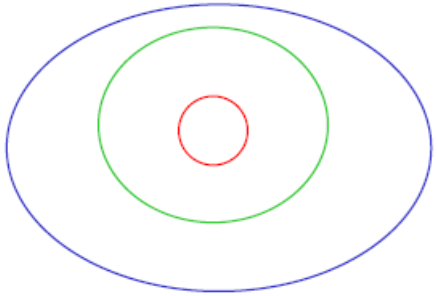
Stage 1

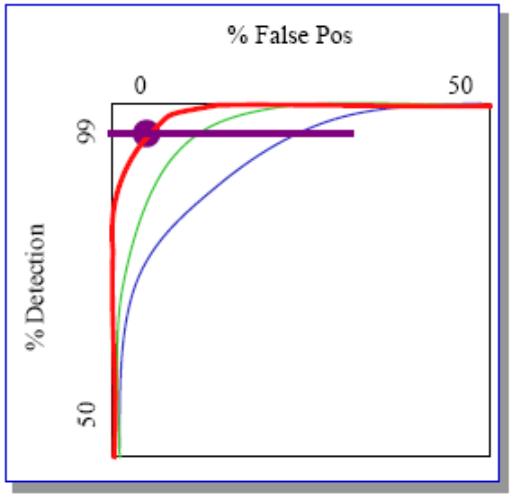
Stage 2

Final Stage

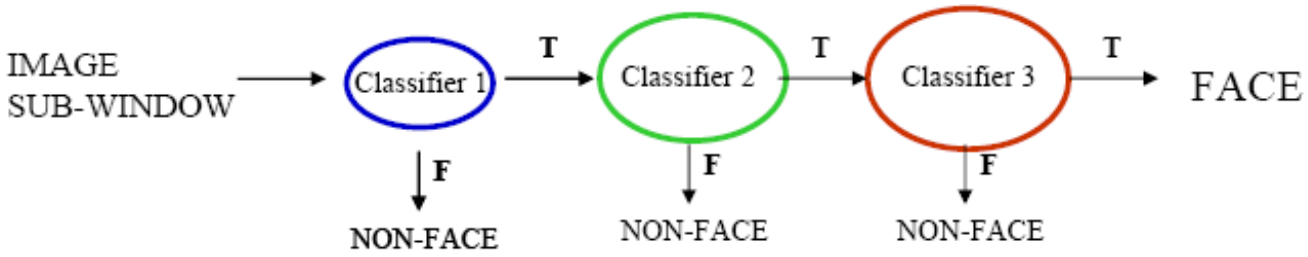
Trading speed for accuracy

- Given a nested set of classifier hypothesis classes

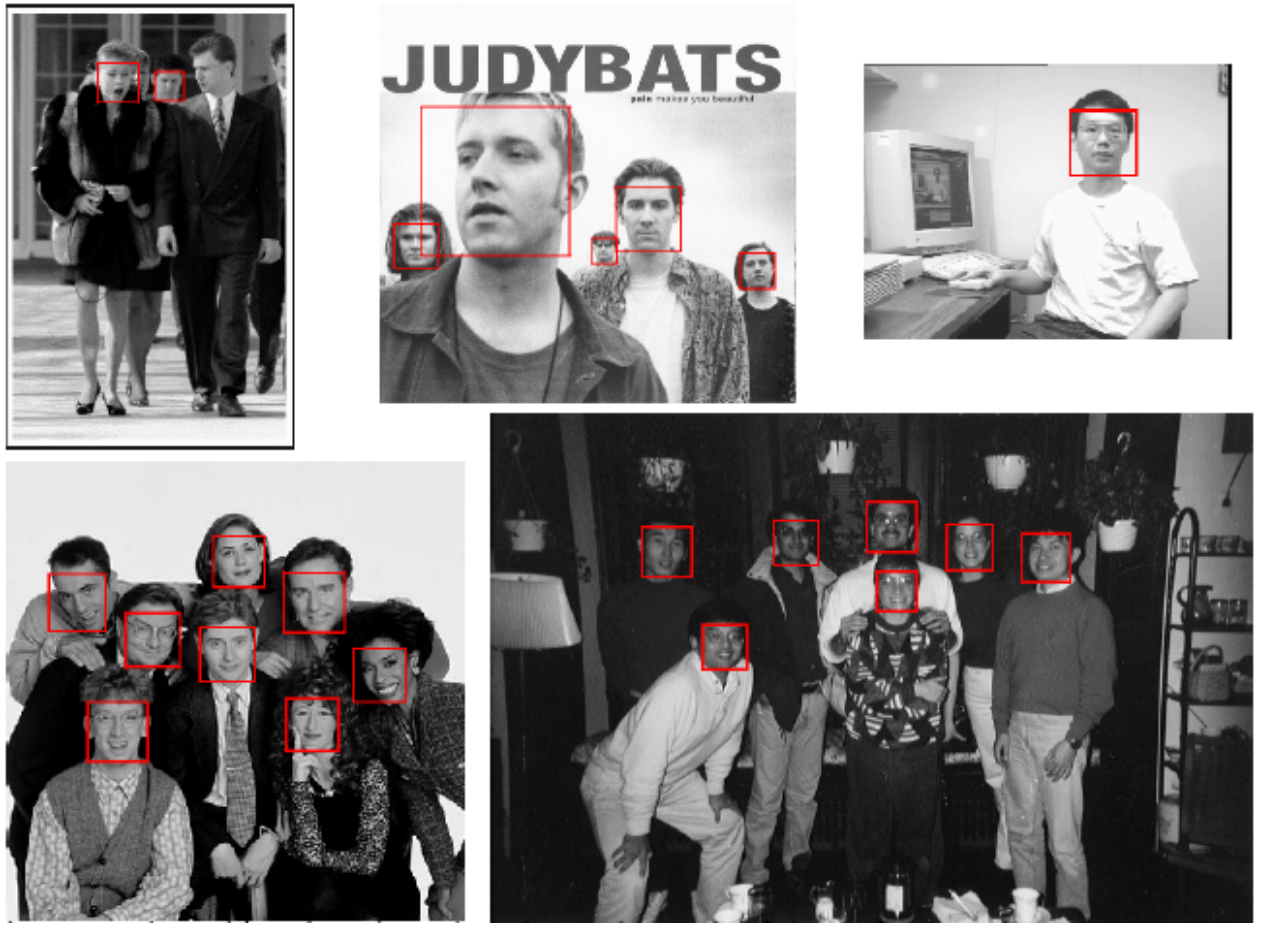




- Computational Risk Minimization



Sample results



Summary (Viola-Jones)

- Fastest known face detector for gray images
- Three contributions with broad applicability:
  - Cascaded classifier yields rapid classification
  - AdaBoost as an extremely efficient feature selector
  - Rectangle Features + Integral Image can be used for rapid image analysis

Modern face detectors

- Now all use DNNs

RetinaFace: Single-stage Dense Face Localisation in the Wild

Jiankang Deng * [1,2,4] Jia Guo * [2] Yuxiang Zhou [1]
Jinke Yu [2] Irene Kotsia [3] Stefanos Zafeiriou [1,4]
[1]Imperial College London [2]InsightFace [3]Middlesex University London [4]FaceSoft

Abstract

Though tremendous strides have been made in uncontrolled face detection, accurate and efficient face localisation in the wild remains an open challenge. This paper presents a robust single-stage face detector, named RetinaFace, which performs pixel-wise face localisation on various scales of faces by taking advantages of joint extra-supervised and self-supervised multi-task learning. Specifically, We make contributions in the following five aspects: (1) We manually annotate five facial landmarks on the WIDER FACE dataset and observe significant improvement in hard face detection with the assistance of this extra supervision signal. (2) We further add a self-supervised mesh decoder branch for predicting a pixel-wise 3D shape face information in parallel with the existing su-

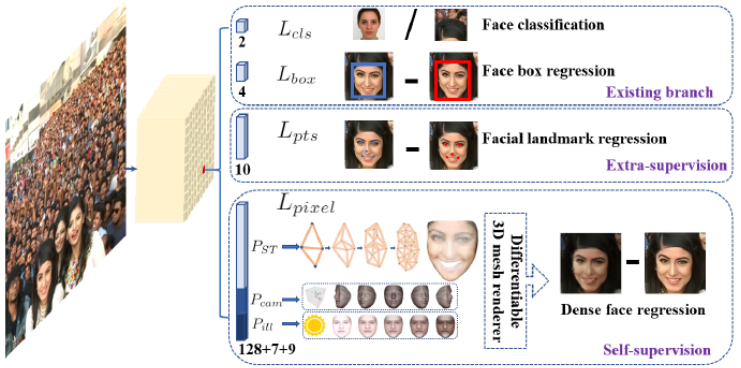


Figure 1. The proposed single-stage pixel-wise face localisation method employs extra-supervised and self-supervised multi-task learning in parallel with the existing box classification and regression branches. Each positive anchor outputs (1) a face score, (2) a face box, (3) five facial landmarks, and (4) dense 3D face vertices projected on the image plane.

Modern face detectors

- Now all use DNNs

RetinaFace: Single-stage Dense Face Localisation in the Wild

Jiankang Deng * [1,2,4] Jia Guo * [2] Yuxiang Zhou [1]
Jinke Yu [2] Irene Kotsia [3] Stefanos Zafeiriou [1,4]
[1]Imperial College London [2]InsightFace [3]Middlesex University London [4]FaceSoft

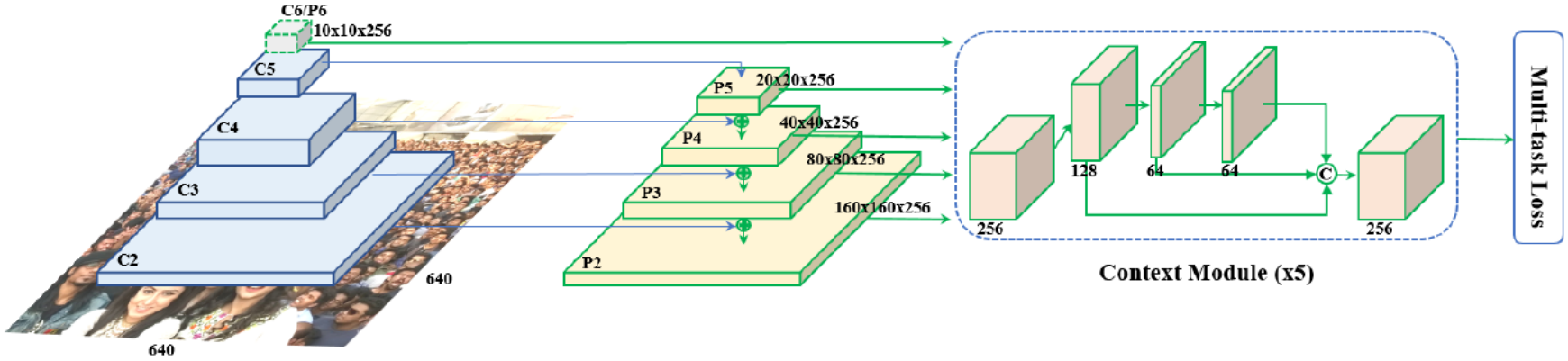
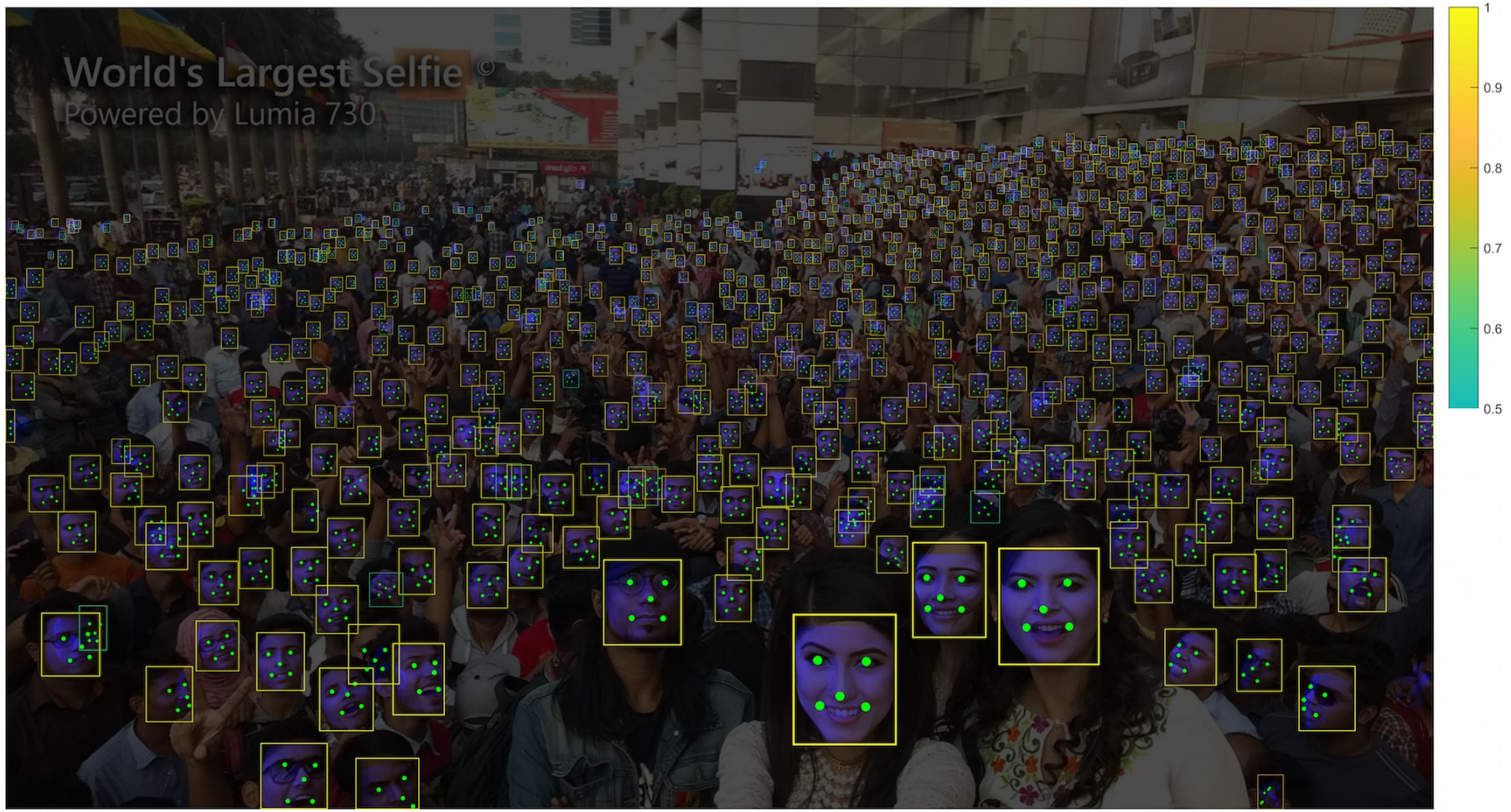


Figure 2. An overview of the proposed single-stage dense face localisation approach. RetinaFace is designed based on the feature pyramids with independent context modules. Following the context modules, we calculate a multi-task loss for each anchor.

Modern face detectors

RetinaFace: Single-stage Dense Face Localisation in the Wild

Jiankang Deng [*1,2,4] Jia Guo [*2] Yuxiang Zhou [1]
Jinke Yu [2] Irene Kotsia [3] Stefanos Zafeiriou [1,4]
[1]Imperial College London [2]InsightFace [3]Middlesex University London [4]FaceSoft

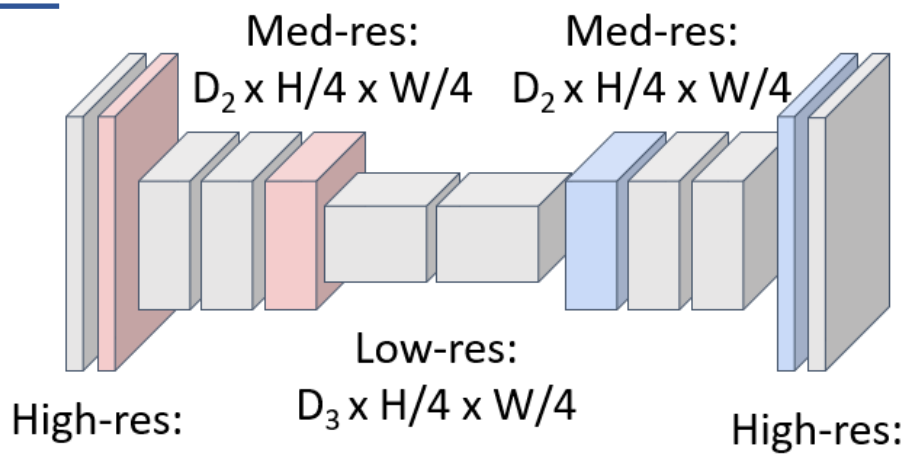


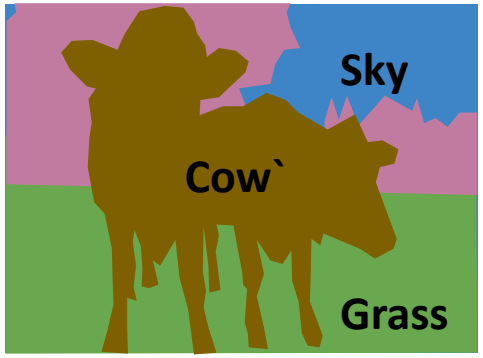
WIDER FACE benchmark

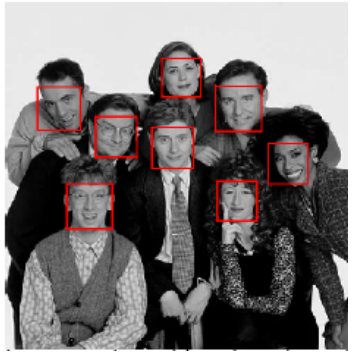


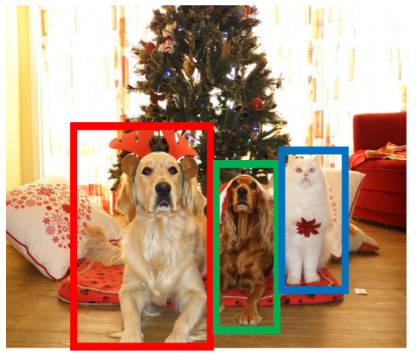
Segmentation and detection

- Adversarial examples
- Bottlenecks and U-Nets
- Segmentation
- Face detection
- **Object detection**









Lecture 15: Object Detection

Today: Object Detection

Classification



CAT

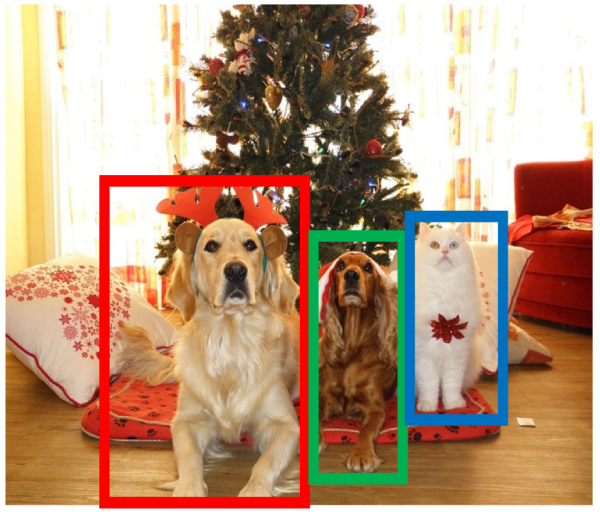
No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Objects

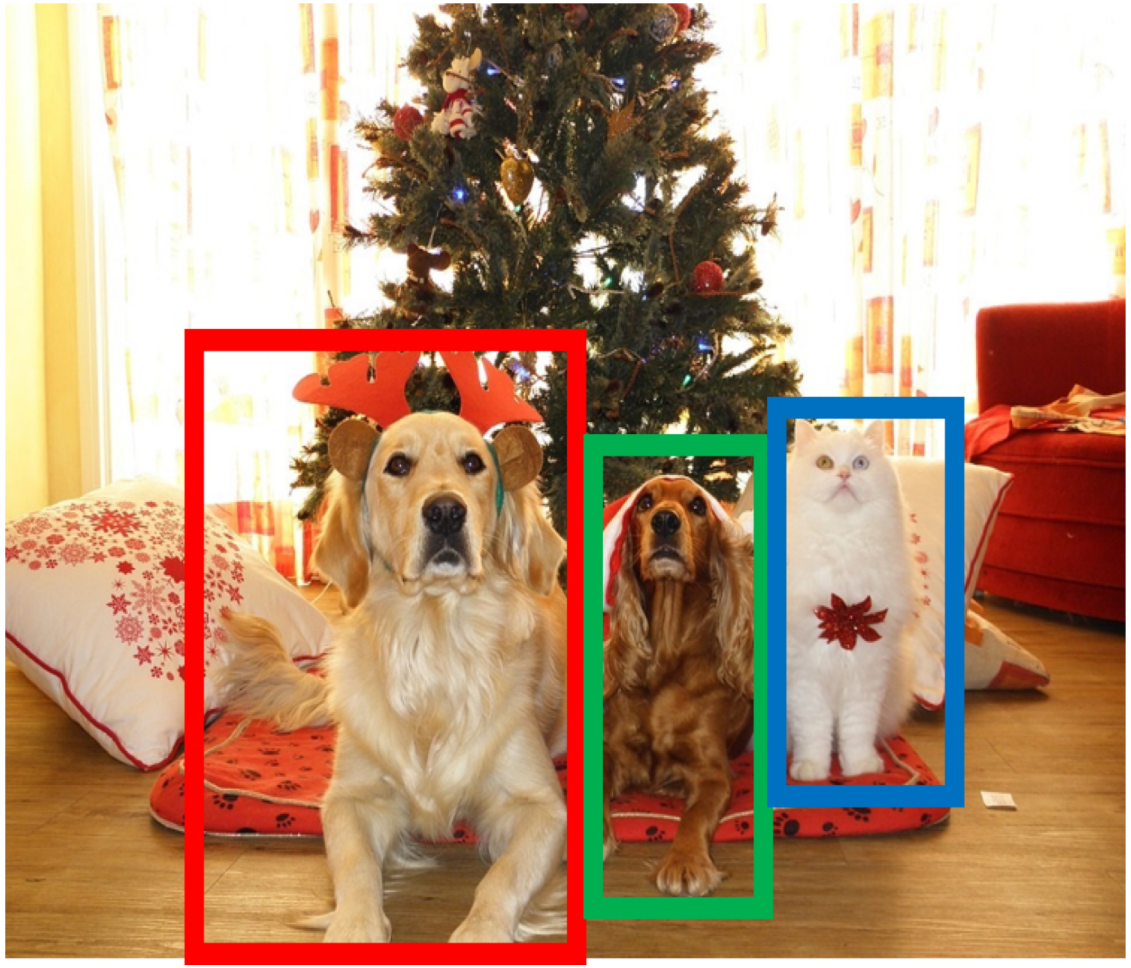
This image is CC0 public domain

Object Detection: Task Definition

Input: Single RGB Image

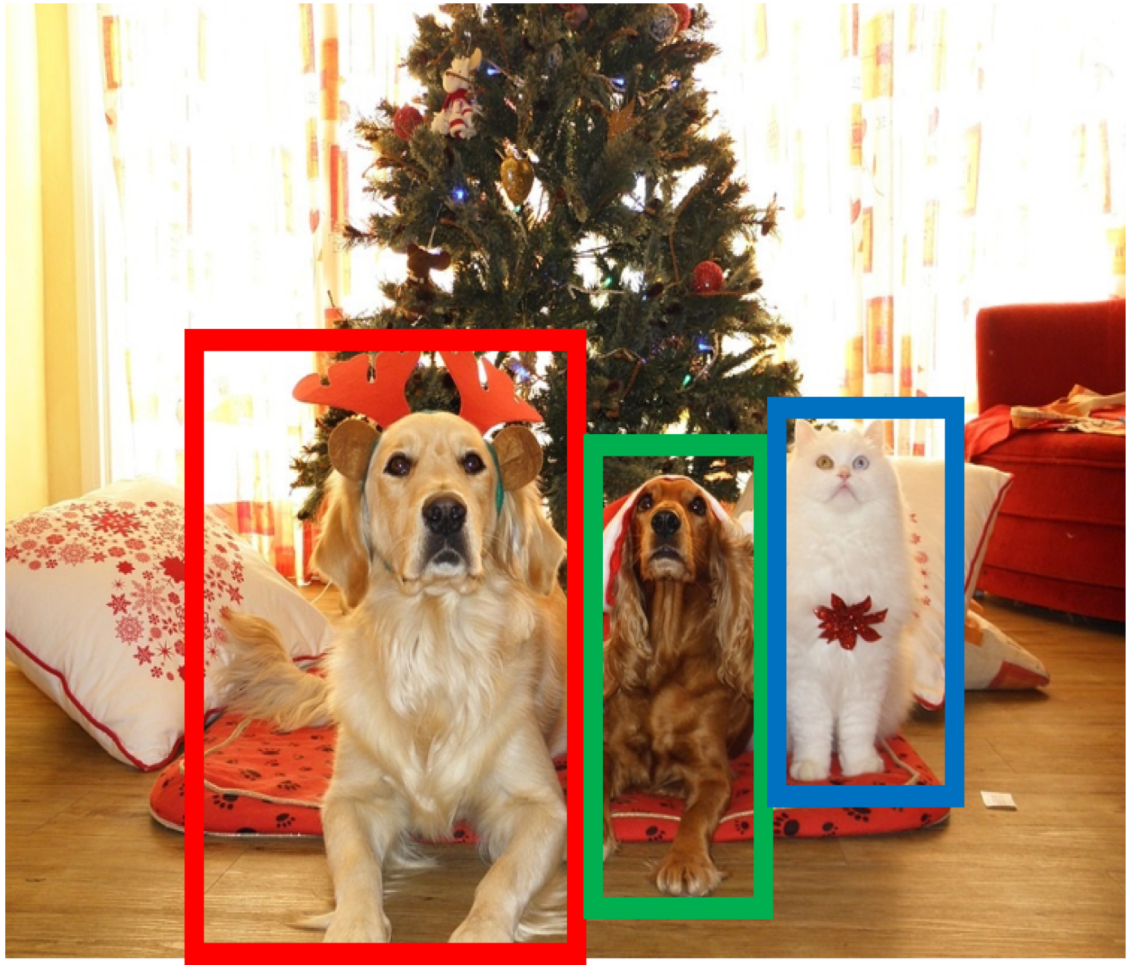
Output: A set of detected objects;
For each object predict:

1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)



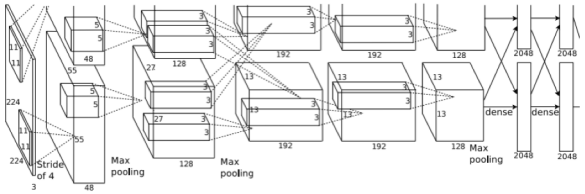
Object Detection: Challenges

- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of output**: Need to predict ”what” (category label) as well as “where” (bounding box)
- **Large images**: Classification works at 224x224; need higher resolution for detection, often ~800x600



Detecting a single object





Vector:
4096

This image is CC0 public domain

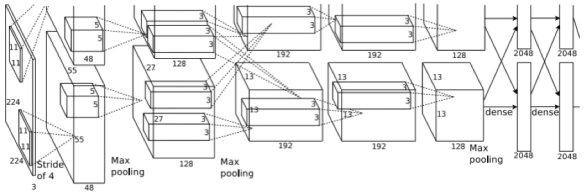
Detecting a single object

"What"

Correct label:
Cat



This image is CC0 public domain



Fully Connected:
4096 to 1000

Vector:
4096

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

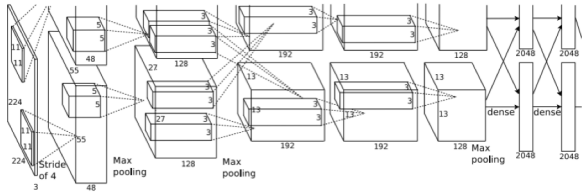
Detecting a single object

"What"

Correct label:
Cat



This image is CC0 public domain



Fully Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

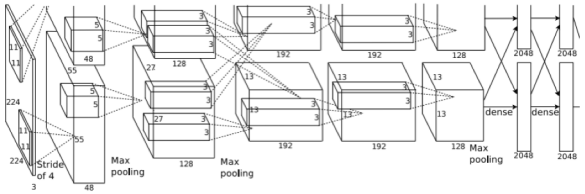
"Where"

Treat localization as a regression problem!

Detecting a single object



This image is CC0 public domain



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label:
Cat

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

Weighted Sum

Loss

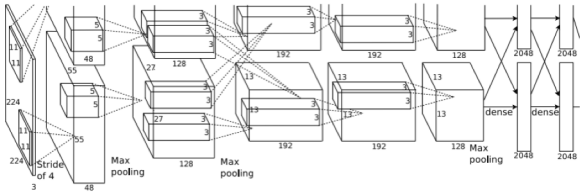
"What"

"Where"

Detecting a single object



This image is CC0 public domain



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

"What"

Correct label:
Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

"Where"

L2 Loss

Correct box:
(x', y', w', h')

Weighted Sum

Loss

Multitask Loss

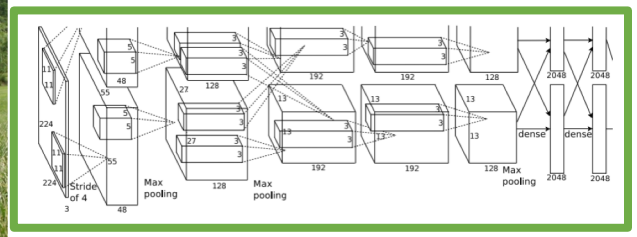
Detecting a single object

"What"

Often pretrained on ImageNet (Transfer learning)



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax Loss

Multitask Loss

Weighted Sum

Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

Treat localization as a regression problem!

"Where"

Detecting a single object

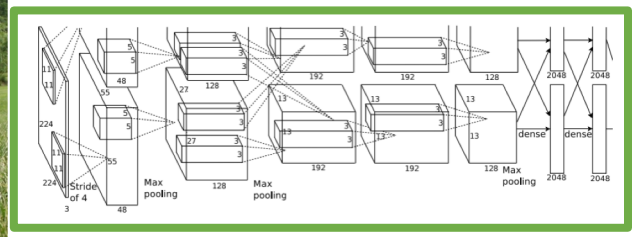
"What"

Correct label:
Cat

Often pretrained on ImageNet (Transfer learning)



This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Multitask Loss

Weighted Sum

Loss

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

Treat localization as a regression problem!

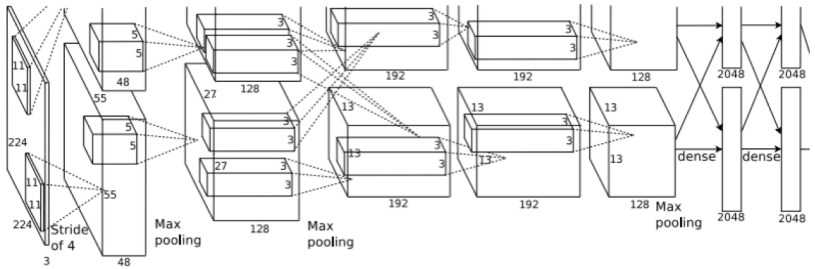
Problem: Images can have more than one object!

"Where"

Detecting Multiple Objects

Need different numbers of outputs per image

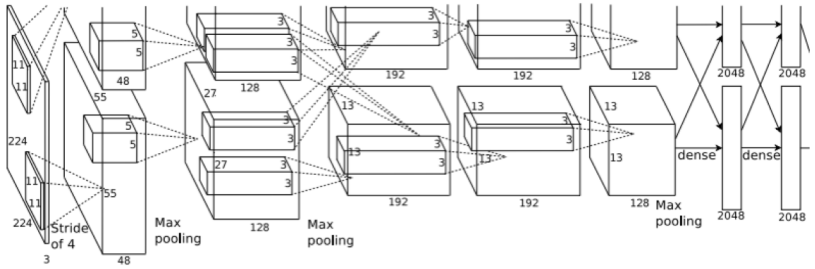




CAT: (x, y, w, h)

4 numbers

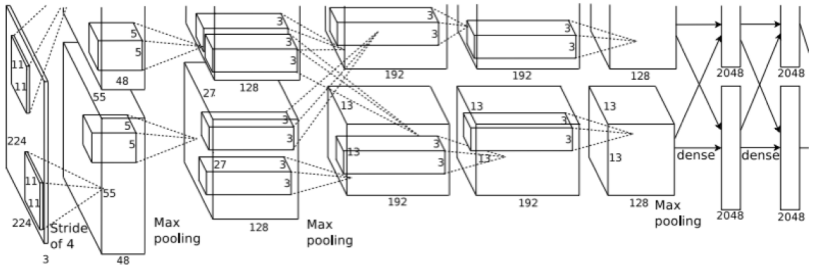




DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers





DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

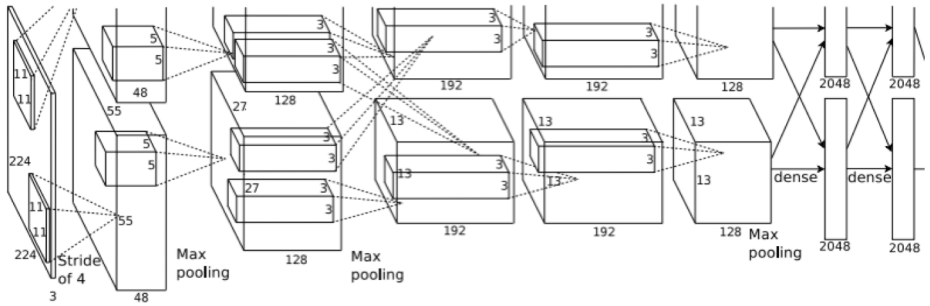
Many numbers!

Duck image is free to use under the Pixabay license

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

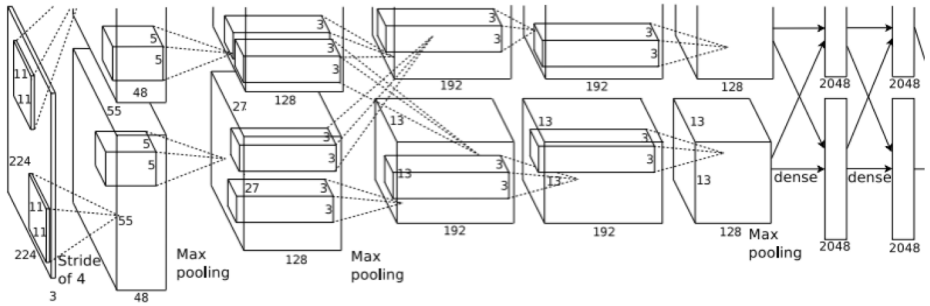


Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

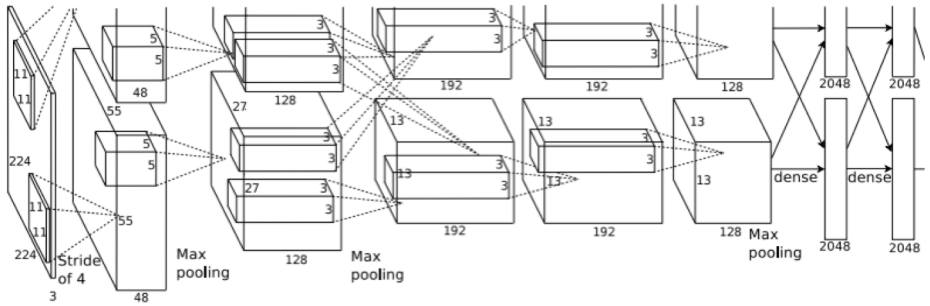


Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

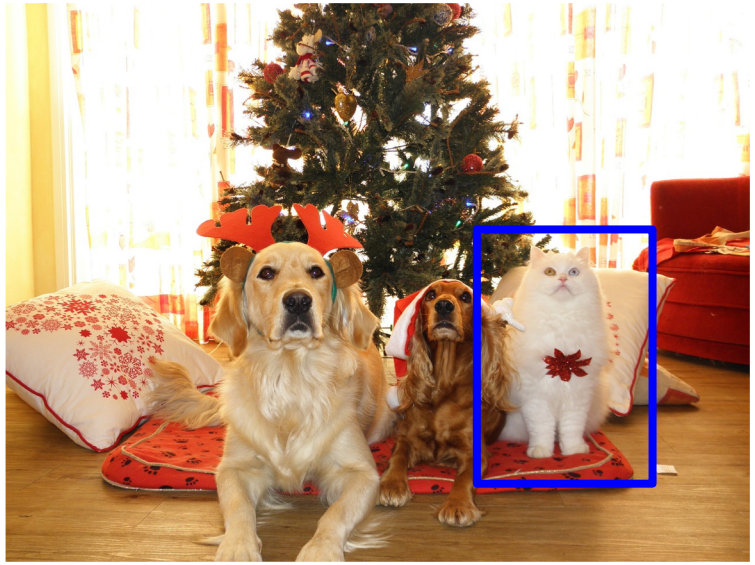


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

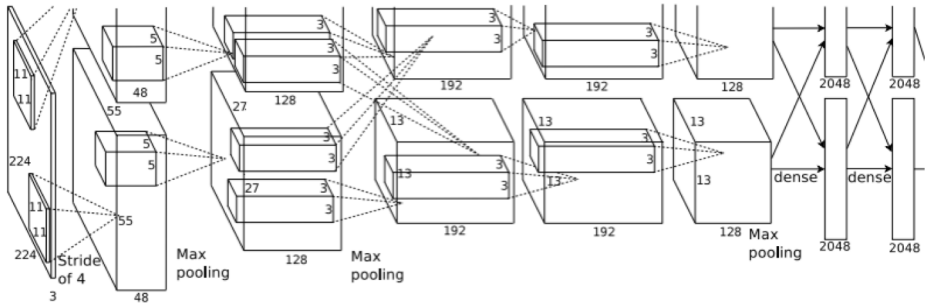


Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

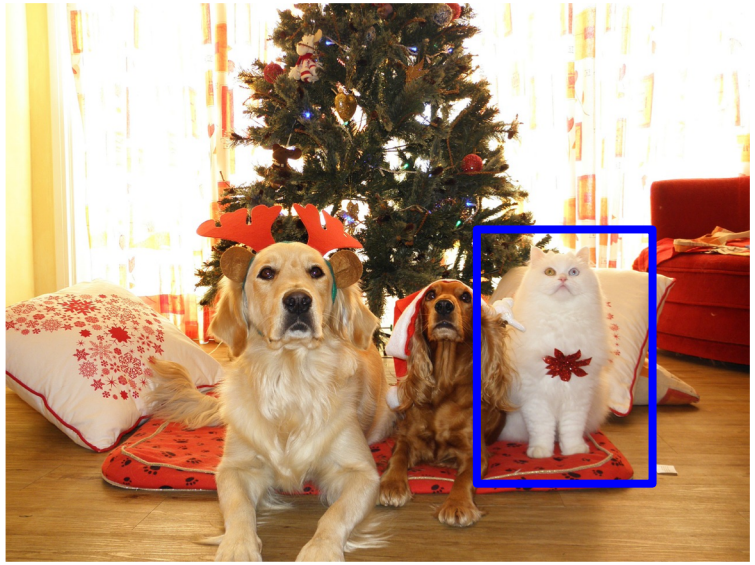


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

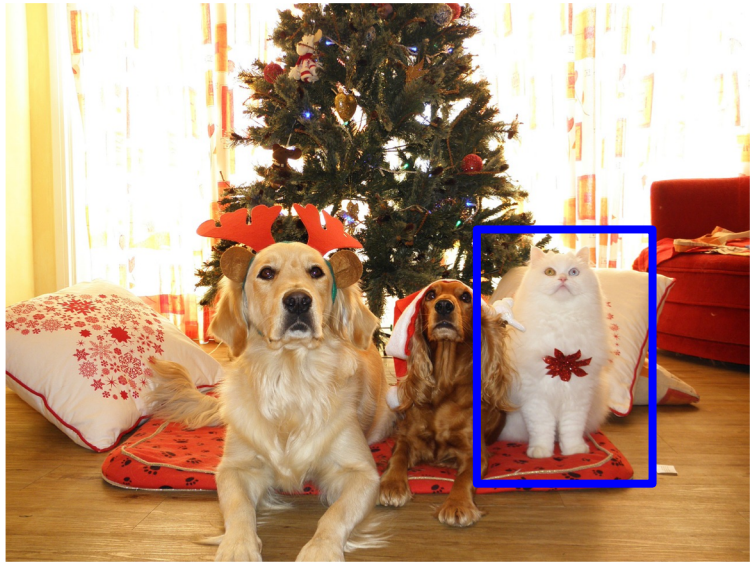
Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

800 x 600 image has ~58M boxes! No way we can evaluate them all

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

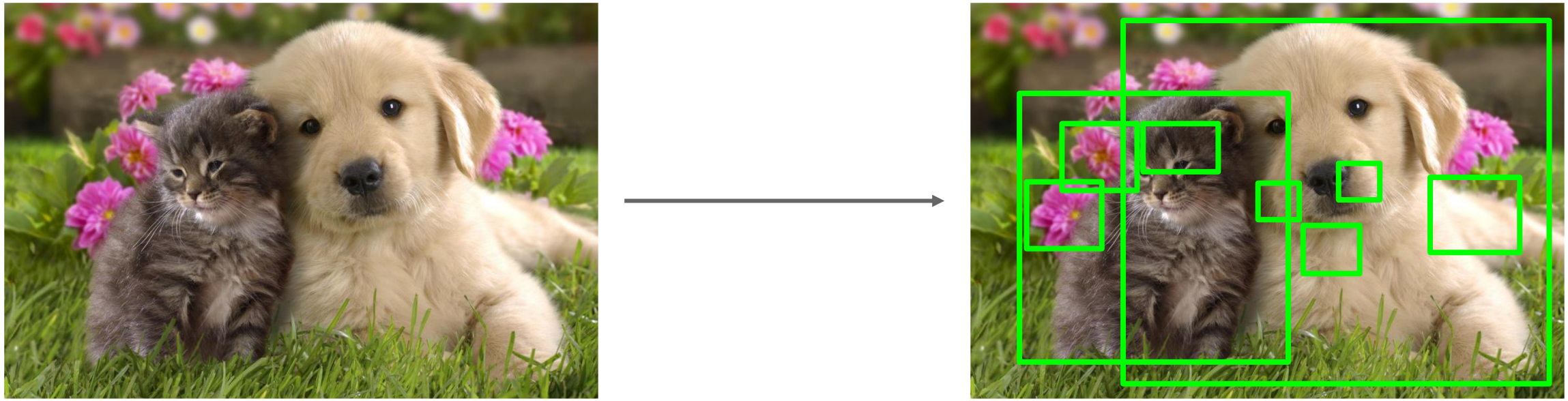
Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

R-CNN: Region-Based CNN

Input image



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN

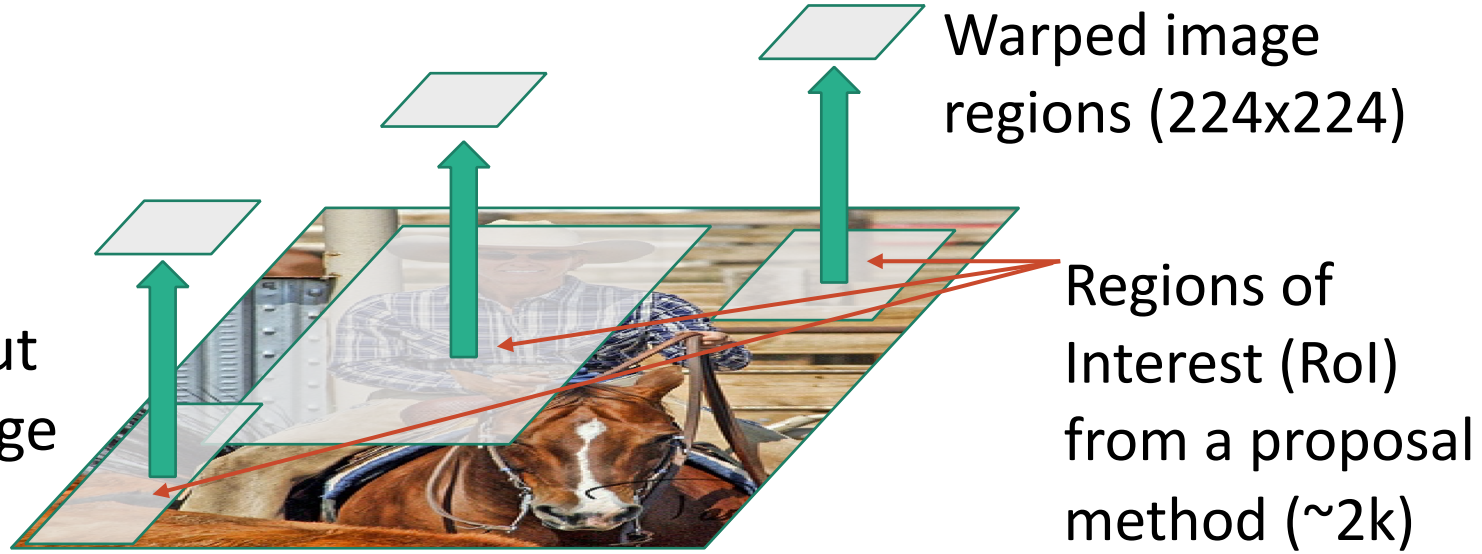
Input image



Regions of Interest (RoI) from a proposal method (~2k)

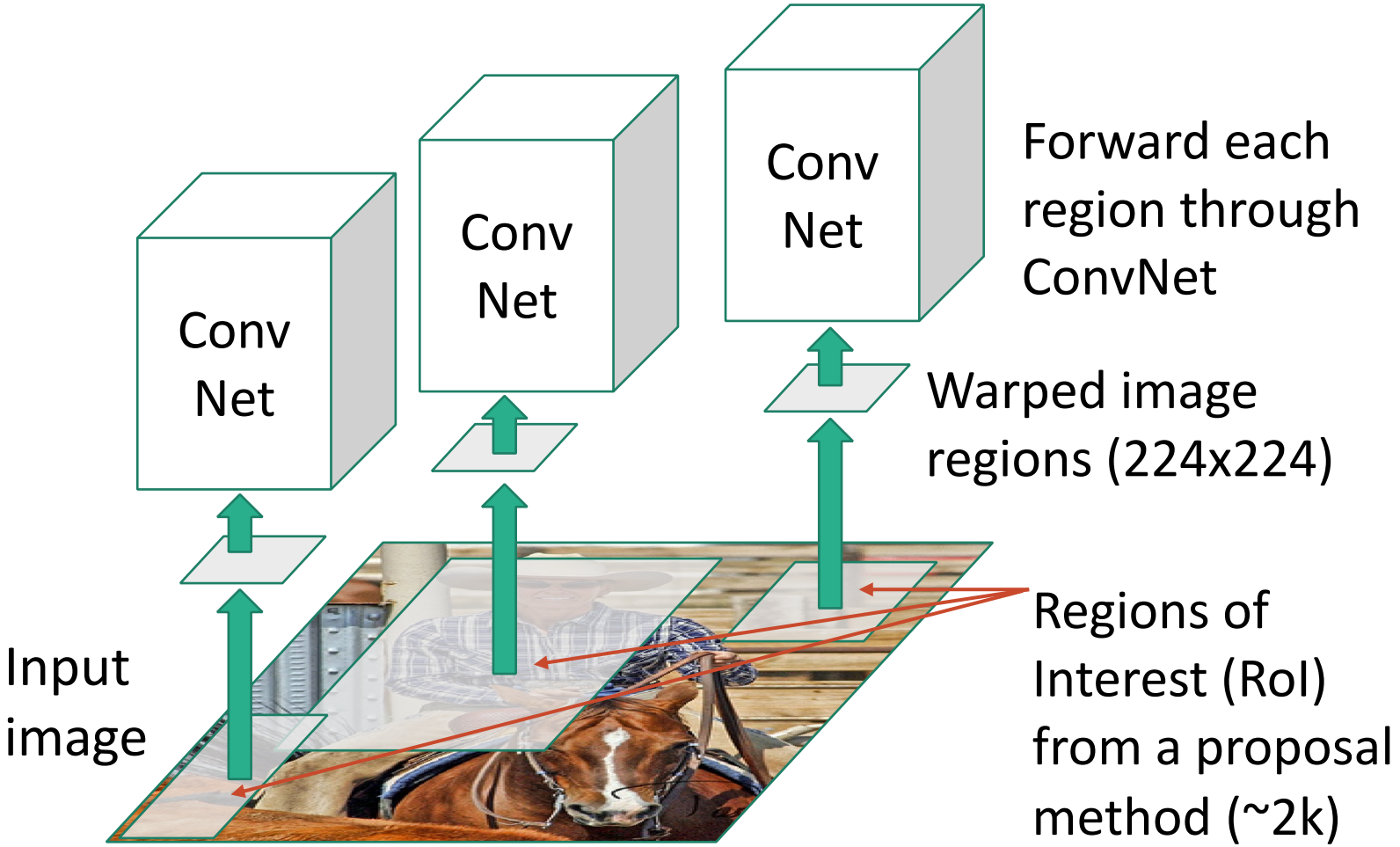
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

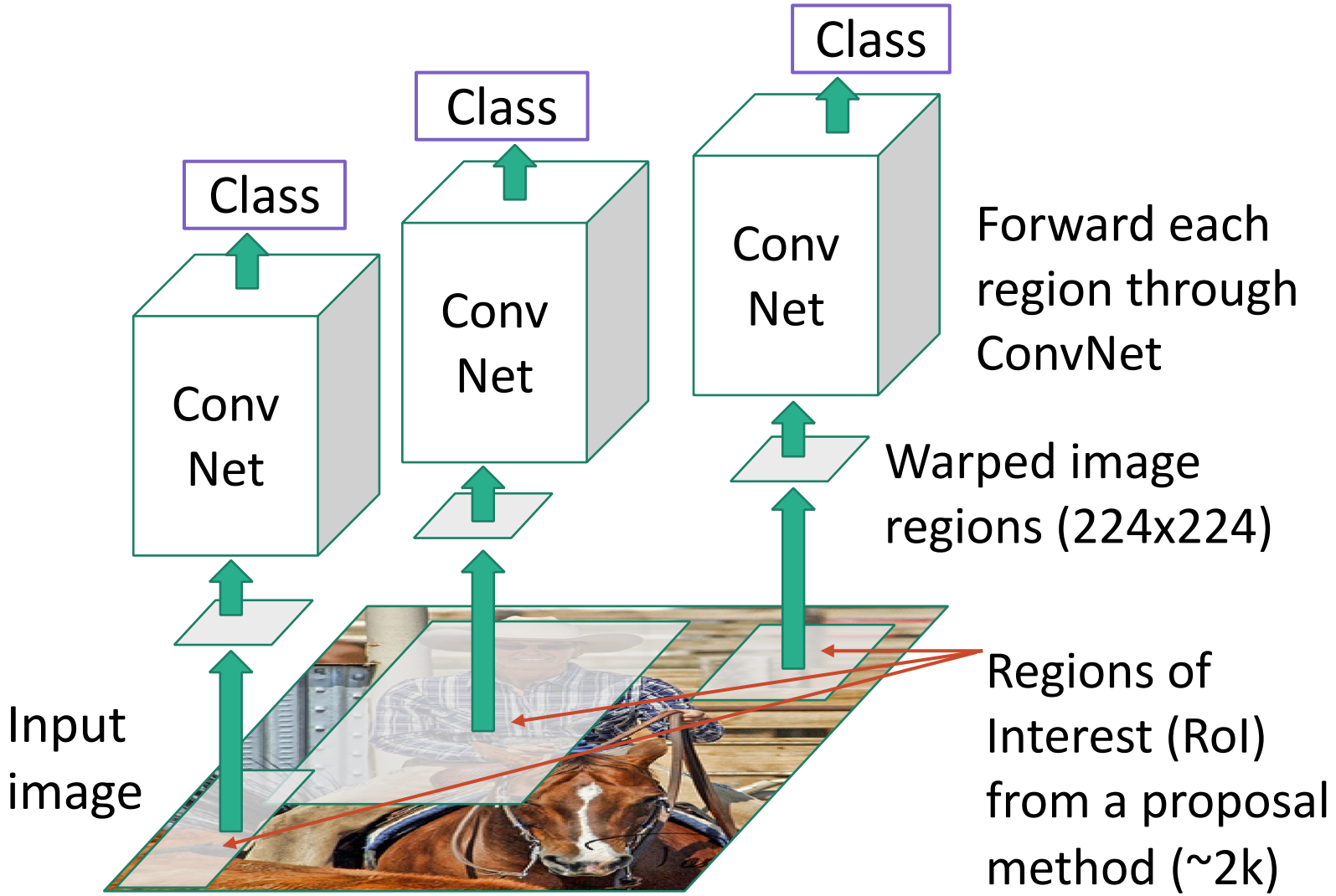
R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN

Classify each region

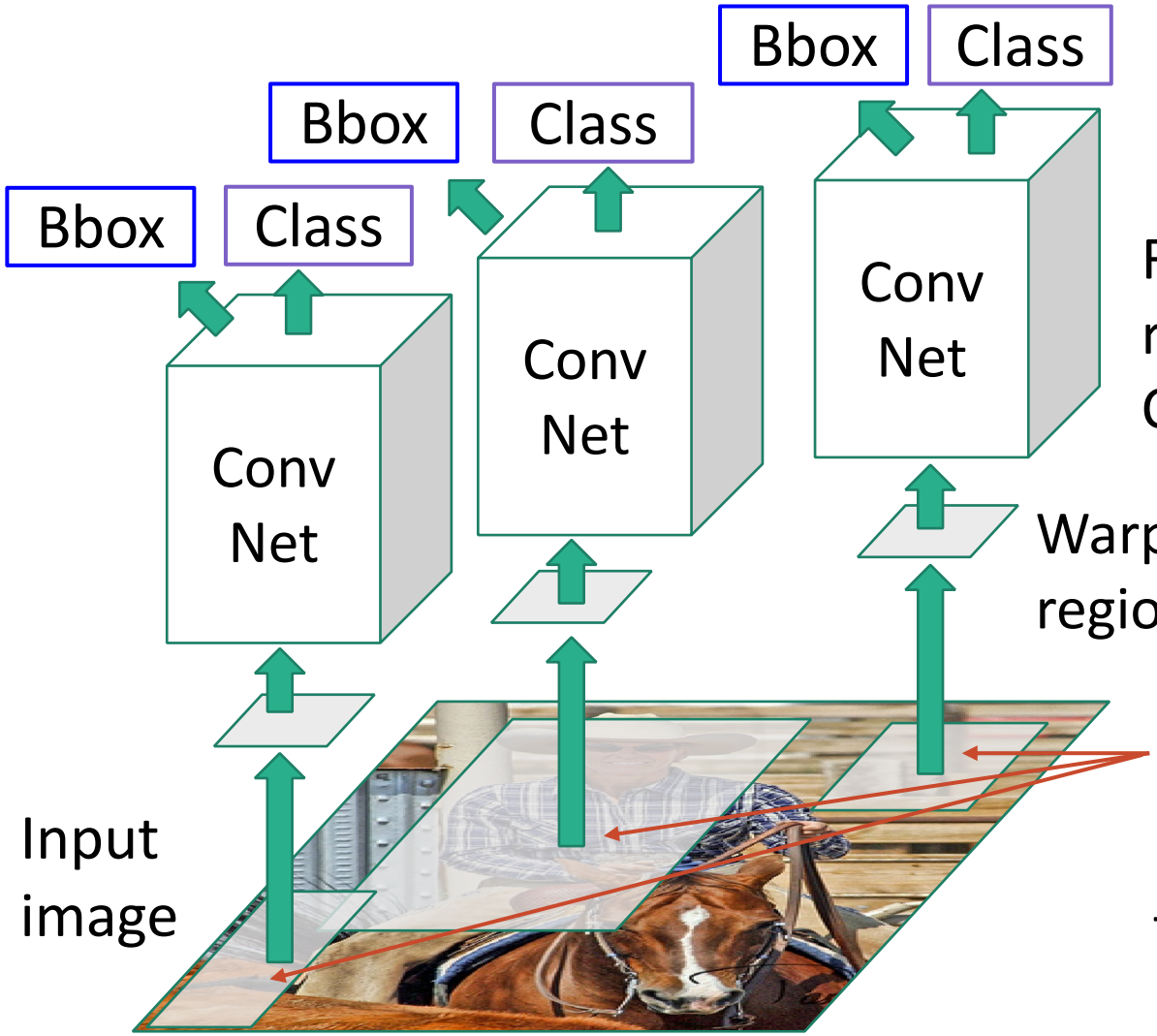


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)



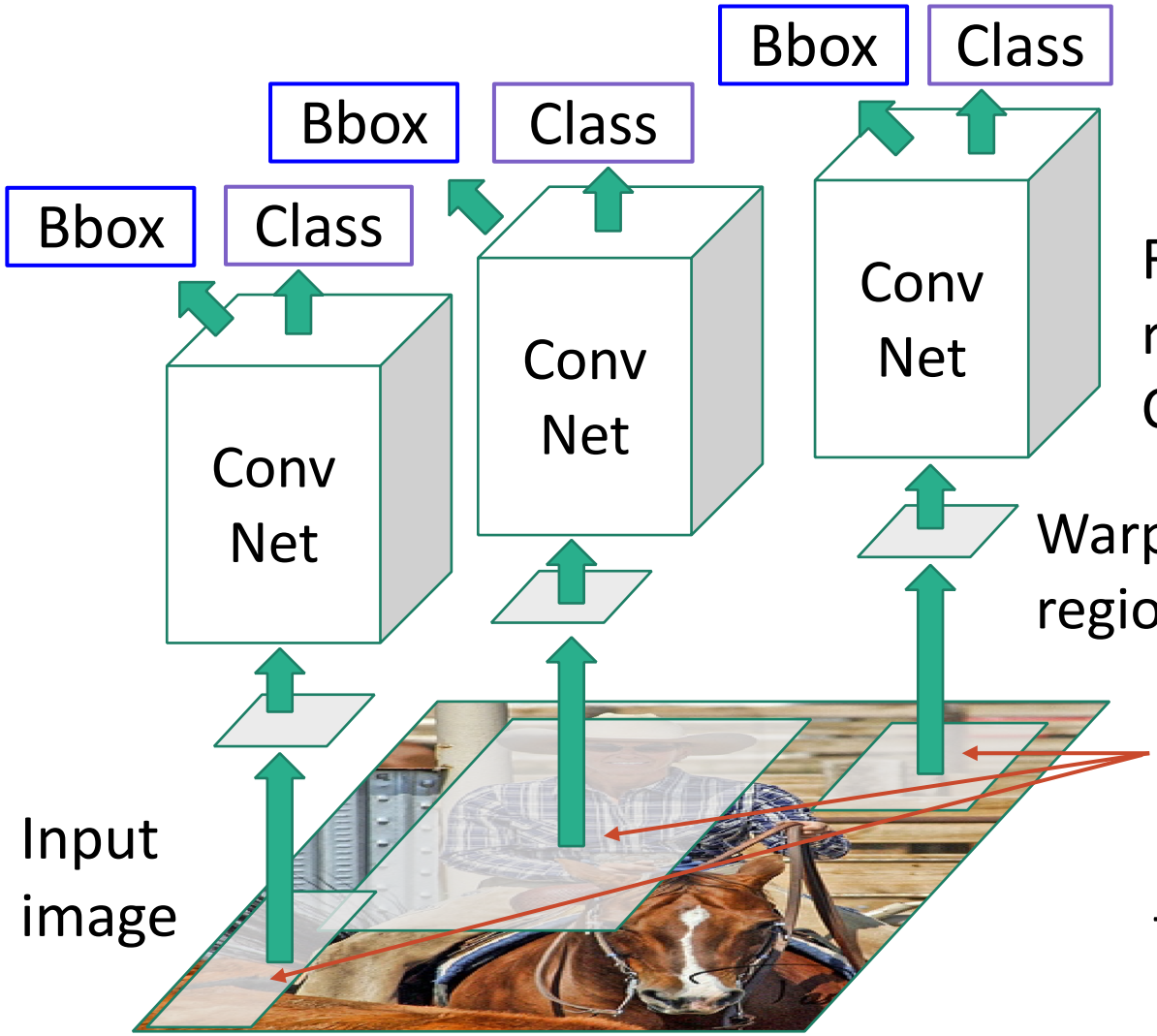
Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN



Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

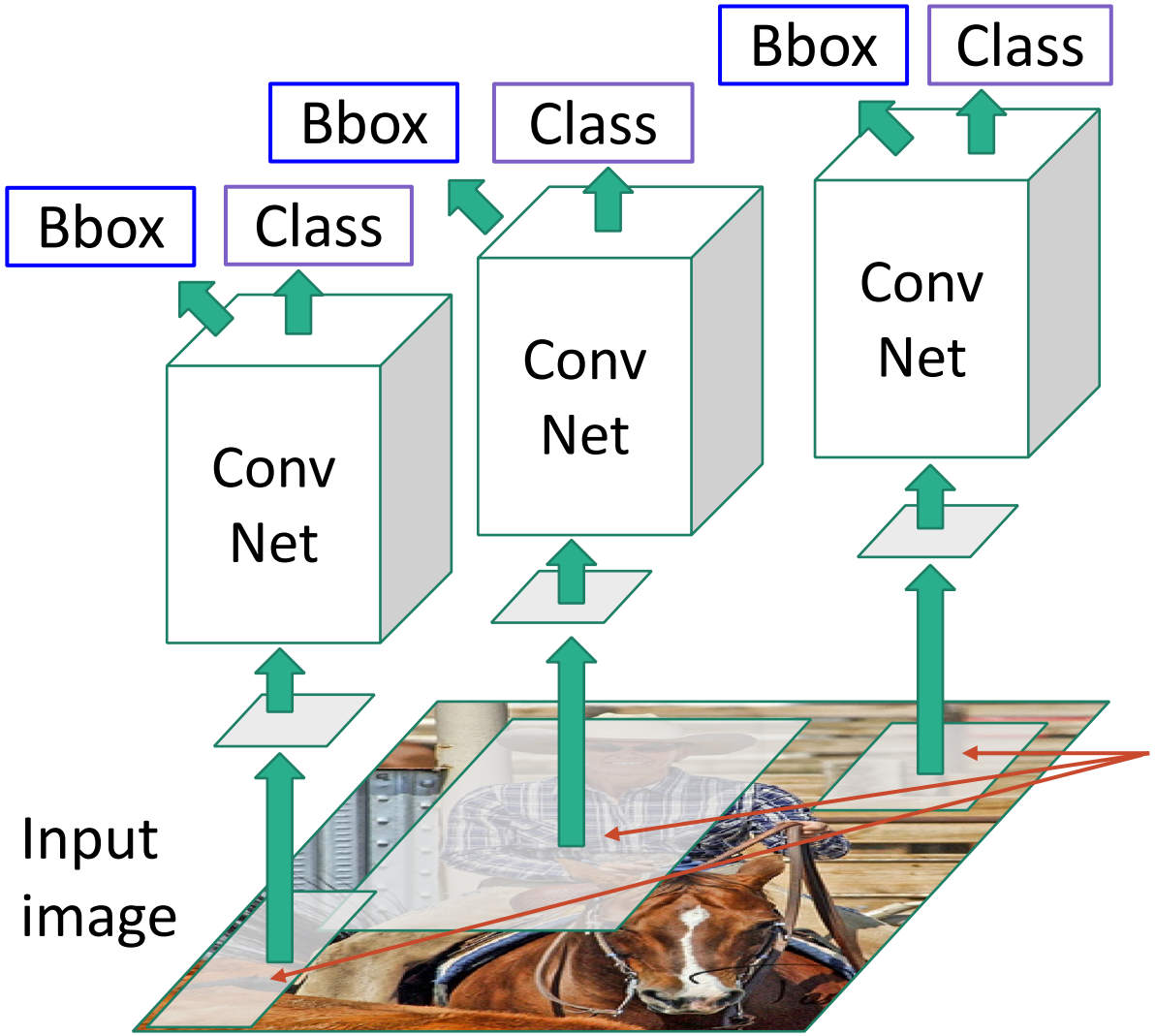
Region proposal: (p_x, p_y, p_h, p_w)
Transform: (t_x, t_y, t_h, t_w)
Output box: (b_x, b_y, b_h, b_w)

Translate relative to box size:
$b_x = p_x + p_w t_x \qquad b_y = p_y + p_h t_y$

Log-space scale transform:
$b_w = p_w \exp(t_w) \qquad b_h = p_h \exp(t_h)$

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Test-time



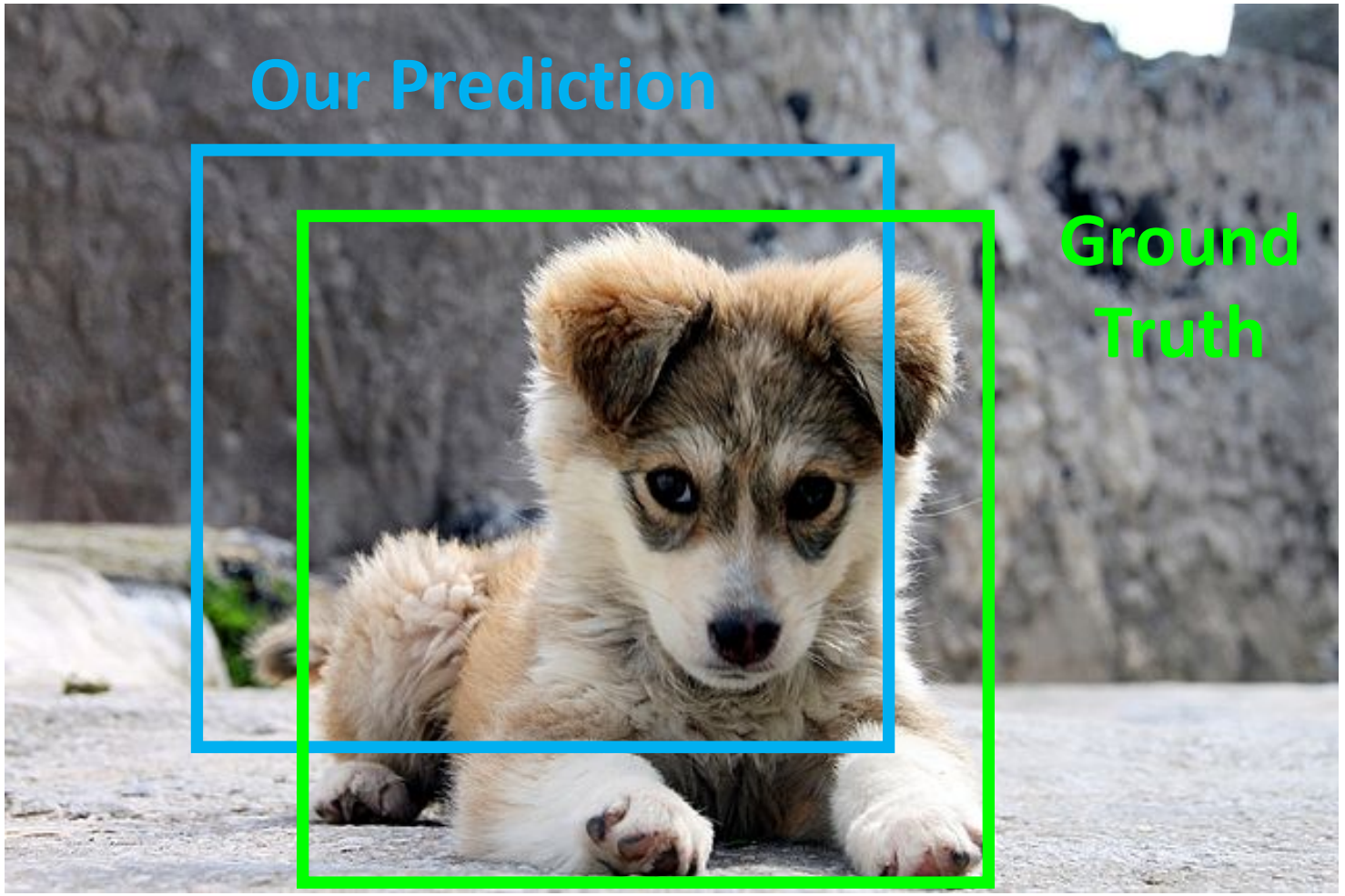
Input: Single RGB Image

1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict class scores and bbox transform
3. Use scores to select a subset of region proposals to output
   (Many choices here: threshold on background, or per-category? Or take top K proposals per image?)
4. Compare with ground-truth boxes

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?



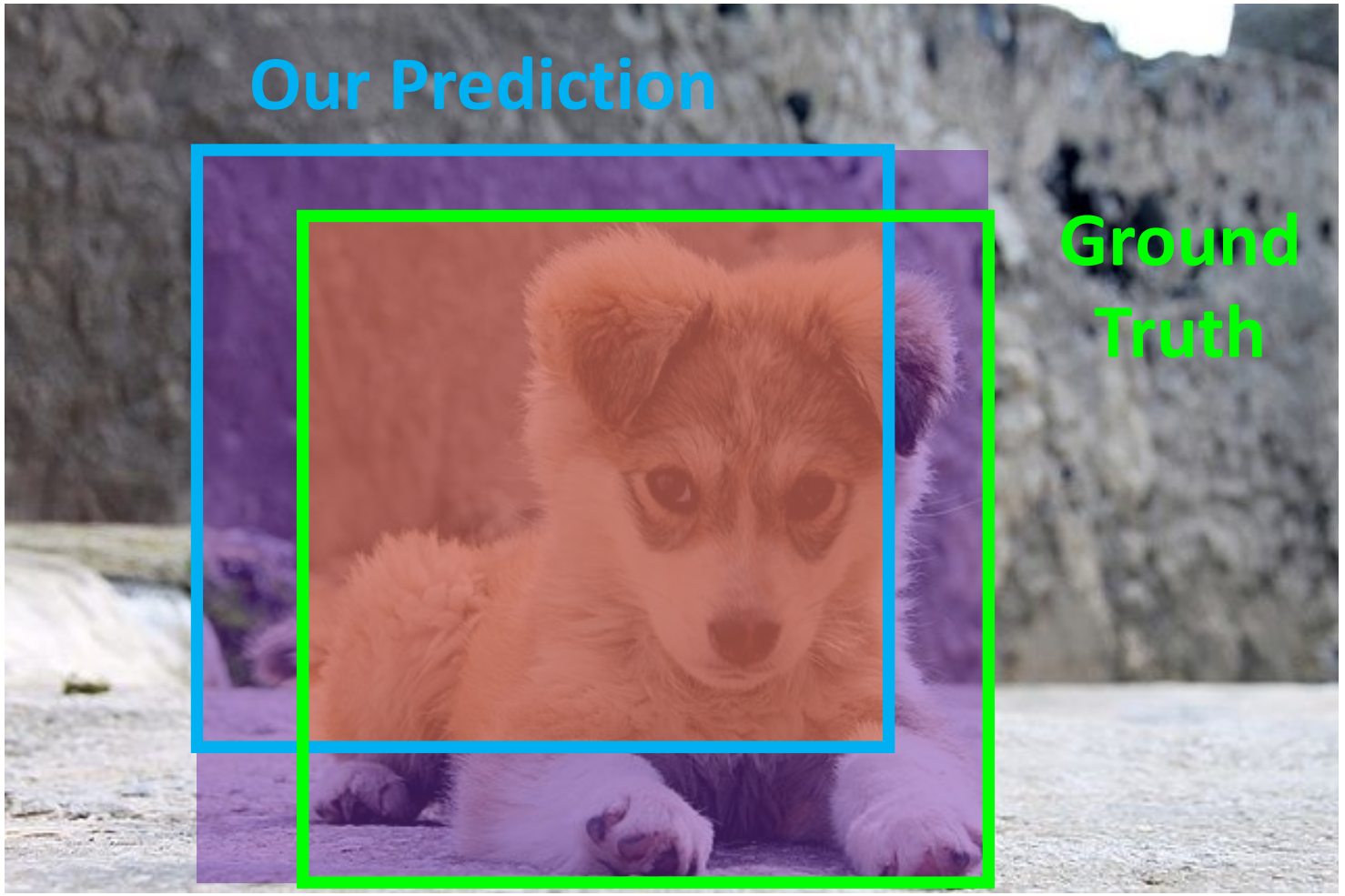
Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

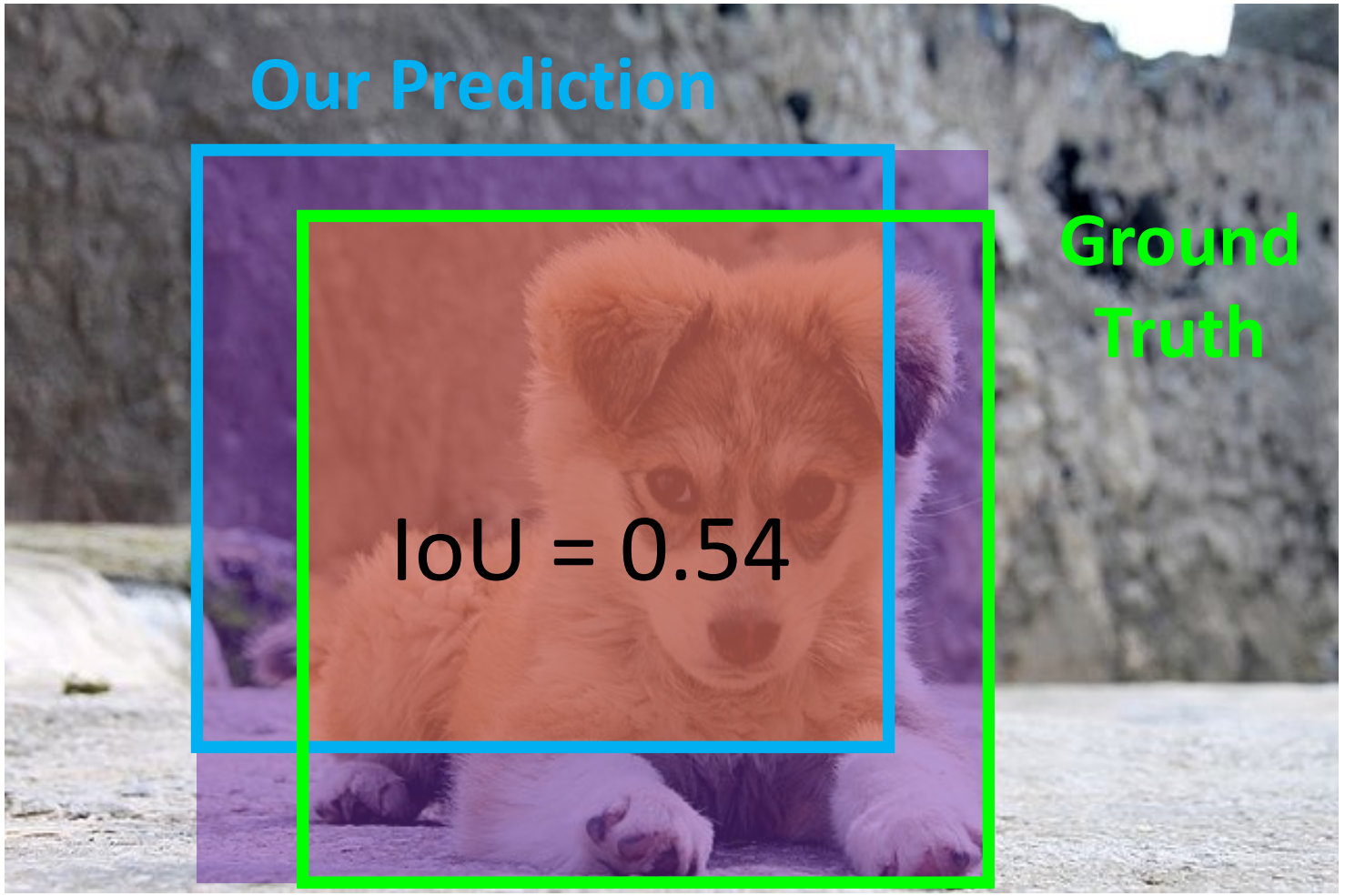
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent"



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

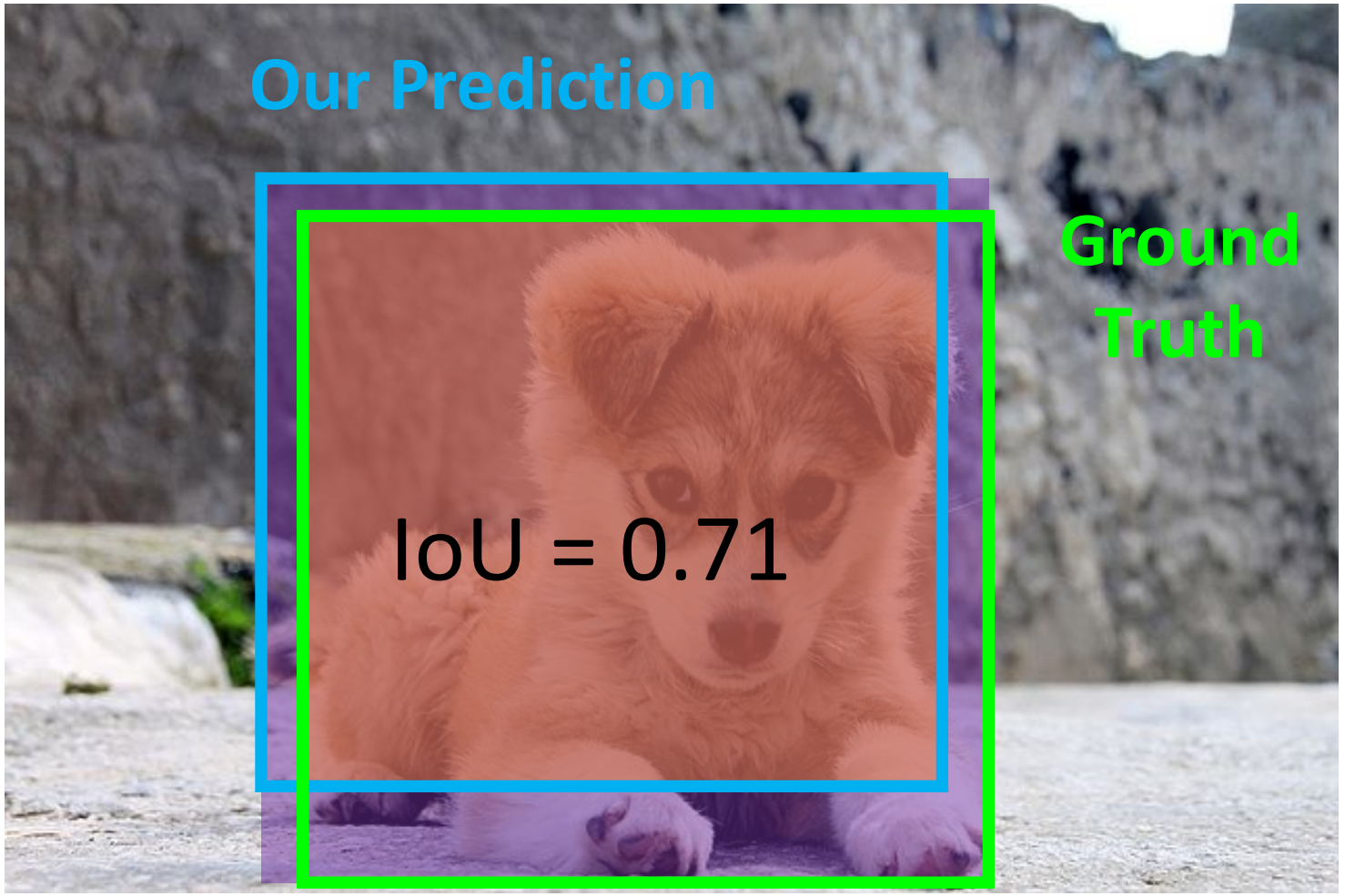
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

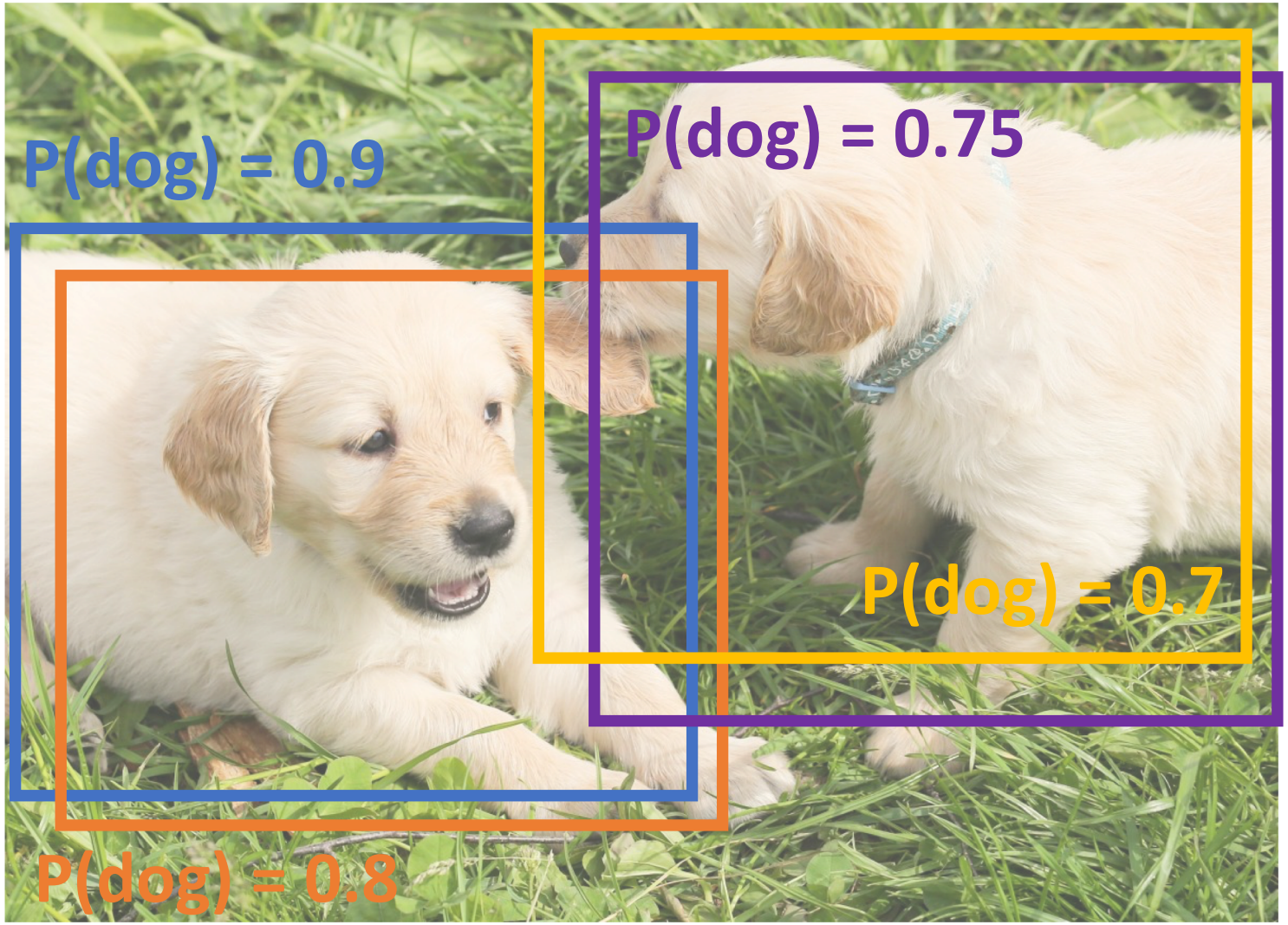
IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",
IoU > 0.9 is "almost perfect"



Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Overlapping Boxes

Problem: Object detectors often output many overlapping detections:



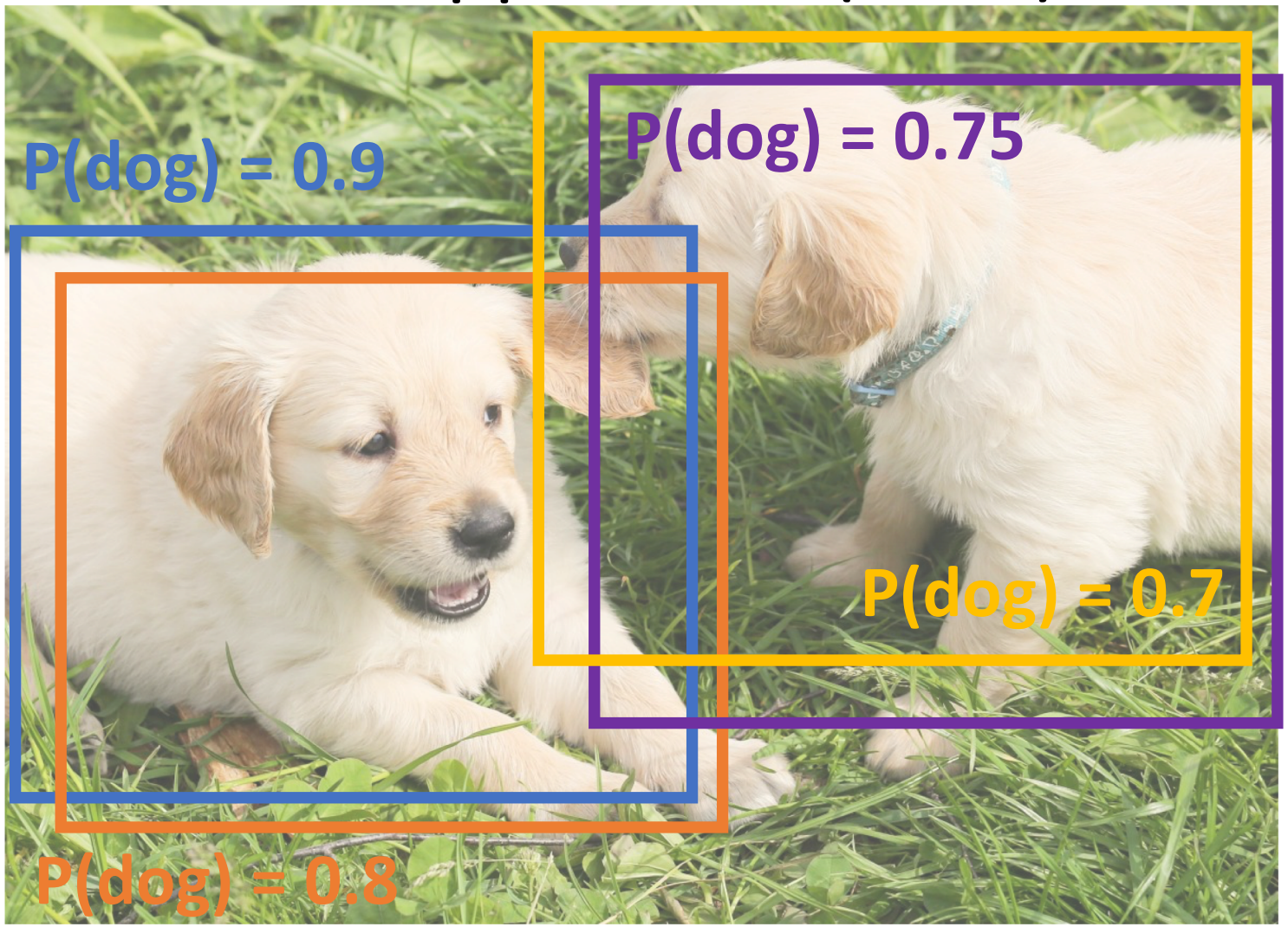
Puppy image is CC0 Public Domain

Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Puppy image is CC0 Public Domain

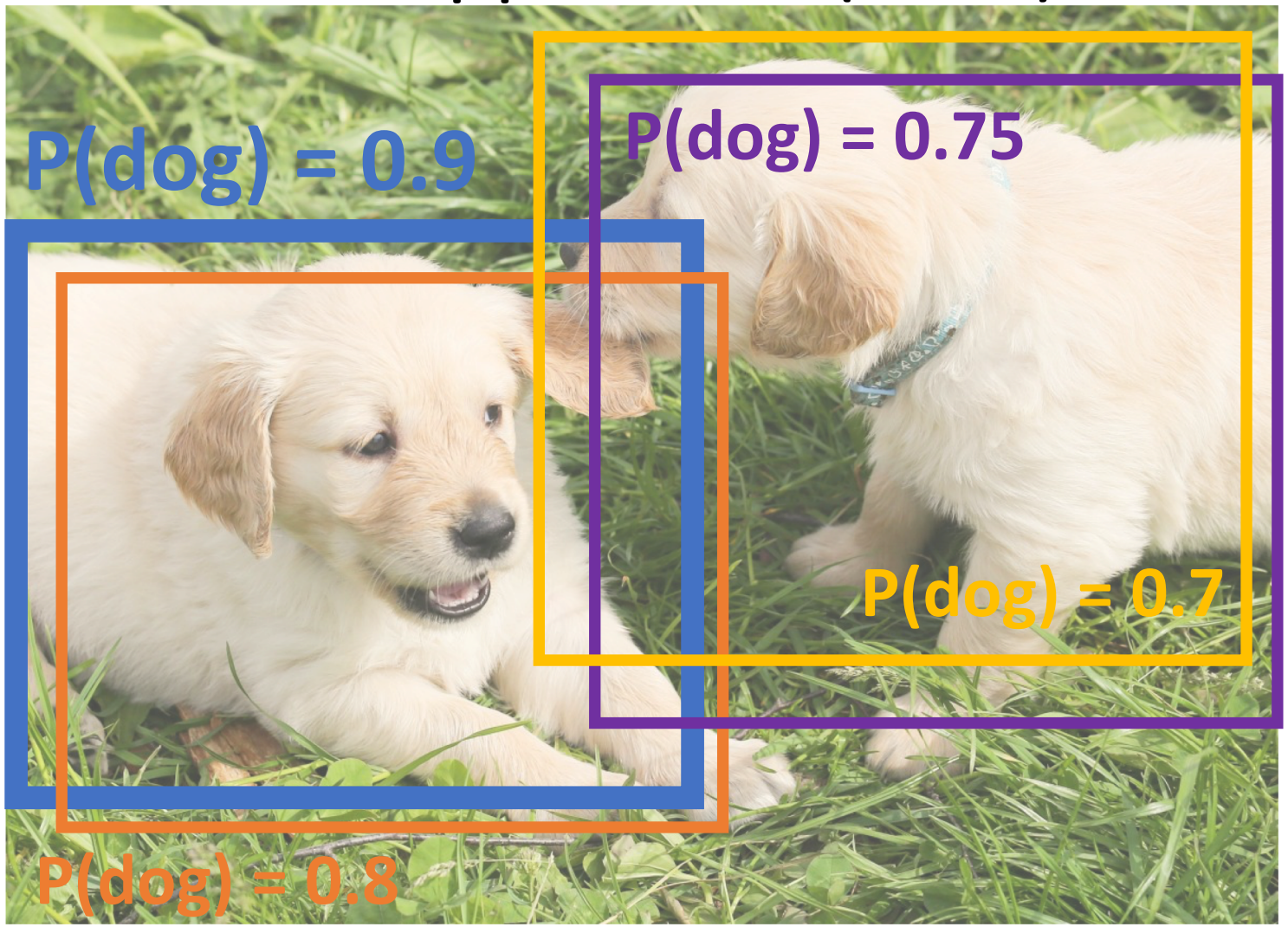
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



Puppy image is CC0 Public Domain

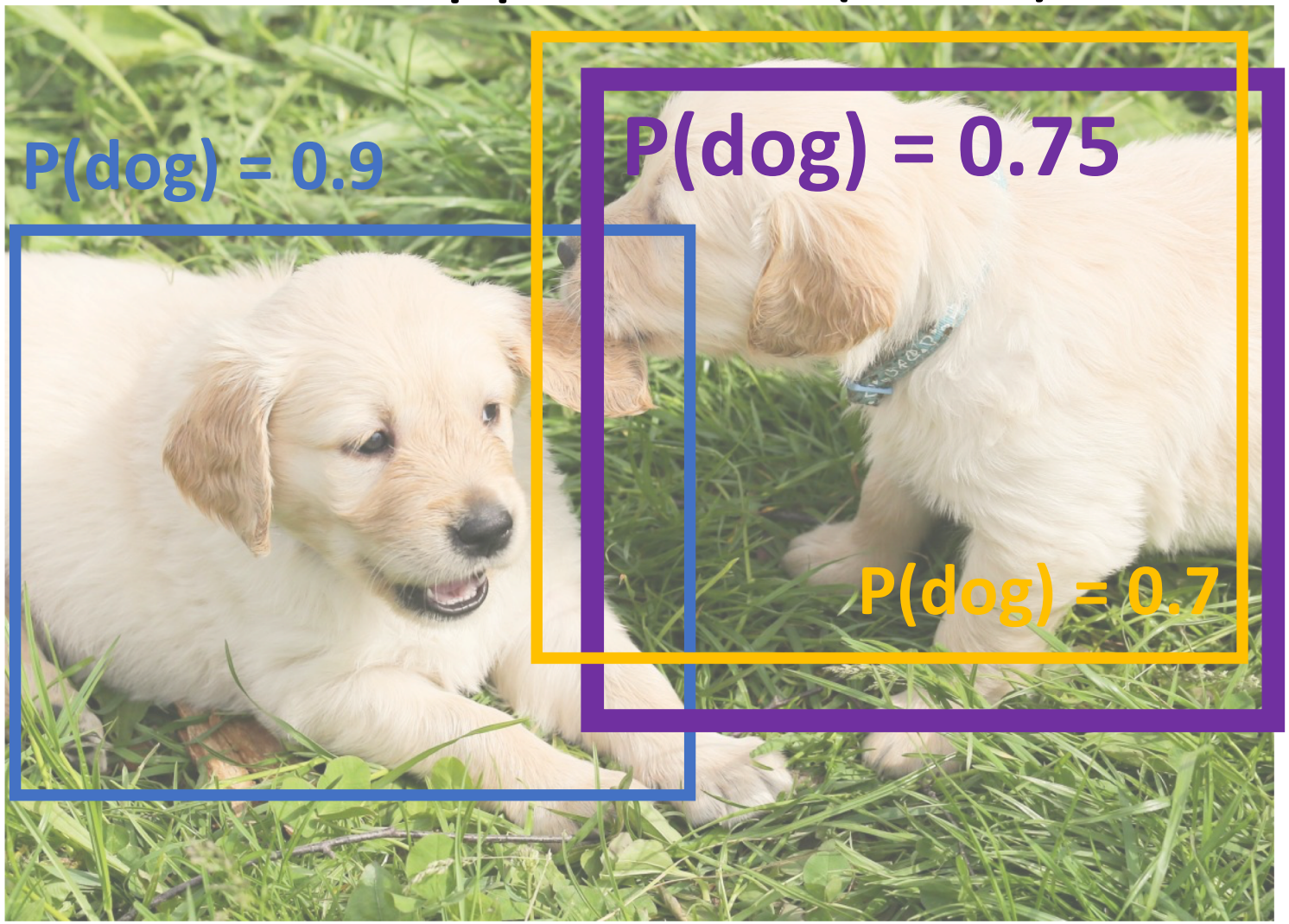
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.74**



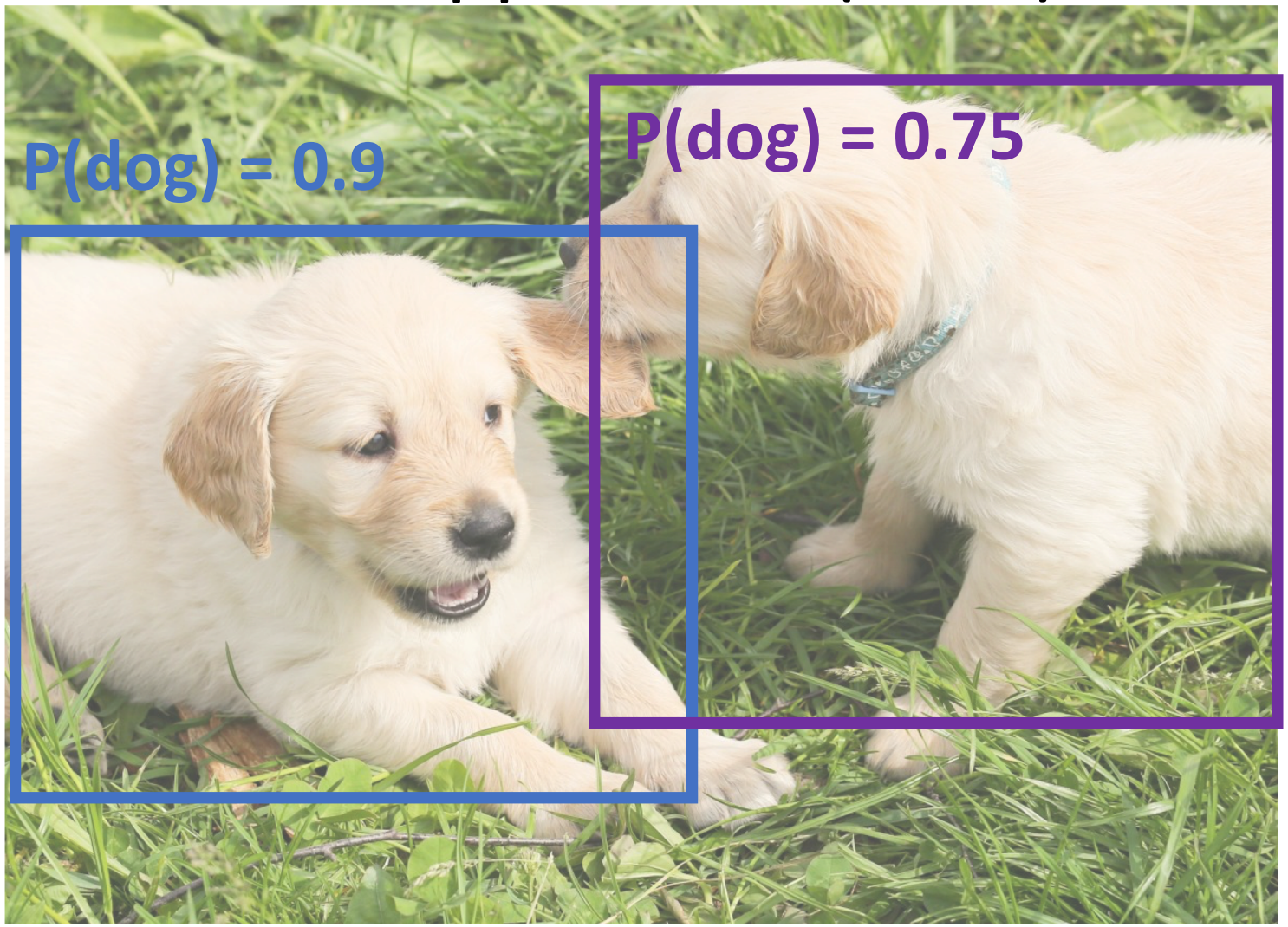
Puppy image is CC0 Public Domain

Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Puppy image is CC0 Public Domain

Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Problem: NMS may eliminate ”good” boxes when objects are highly overlapping… no good solution =(



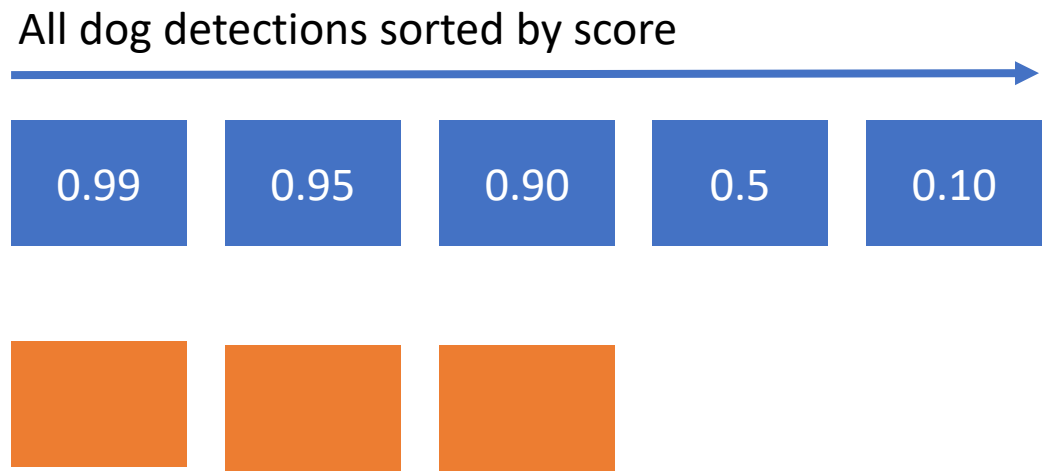
Crowd image is free for commercial use under the Pixabay license

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve

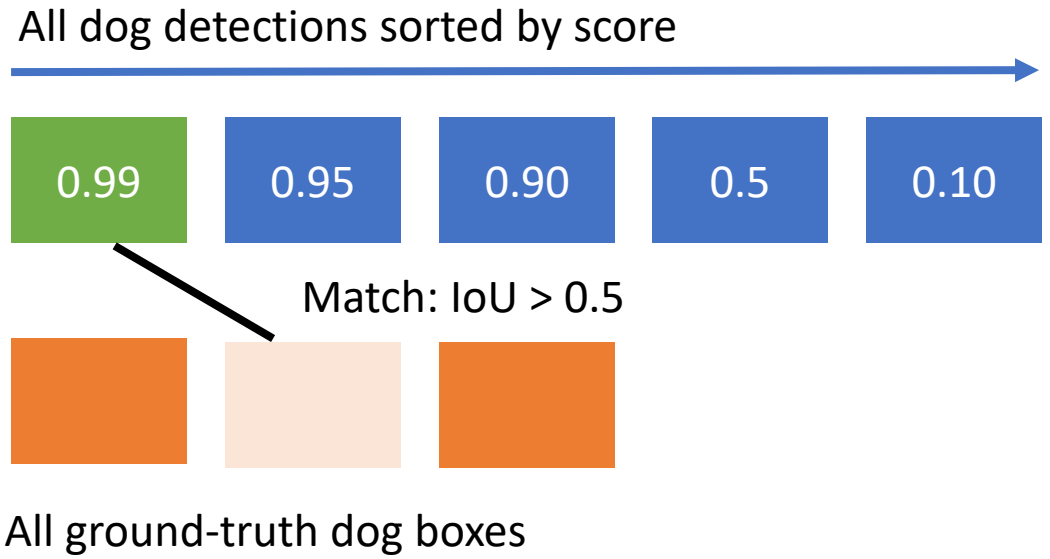
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)



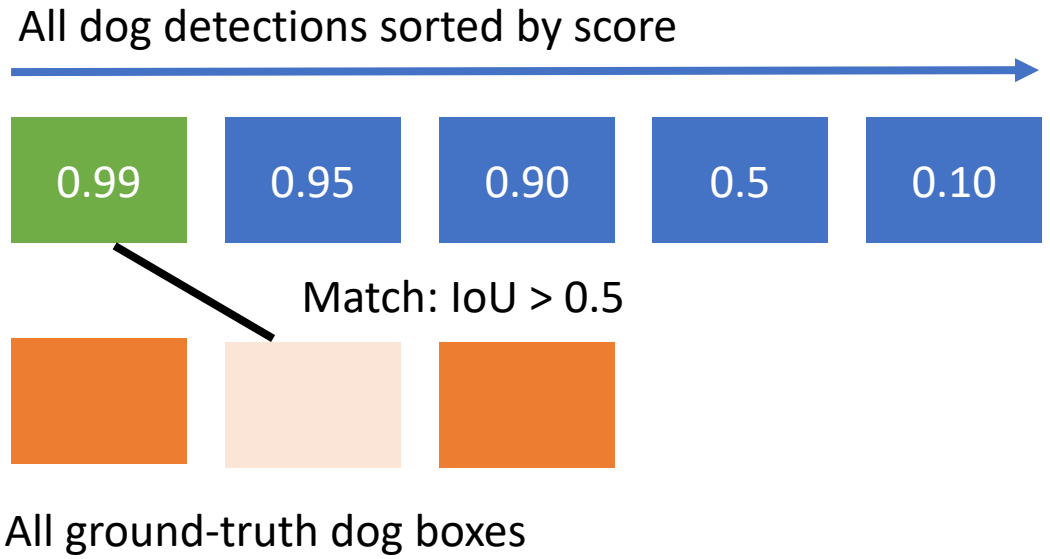
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative

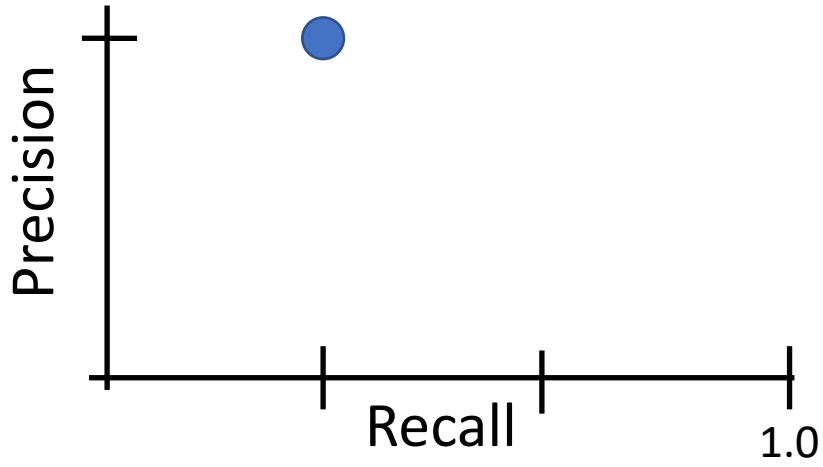


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

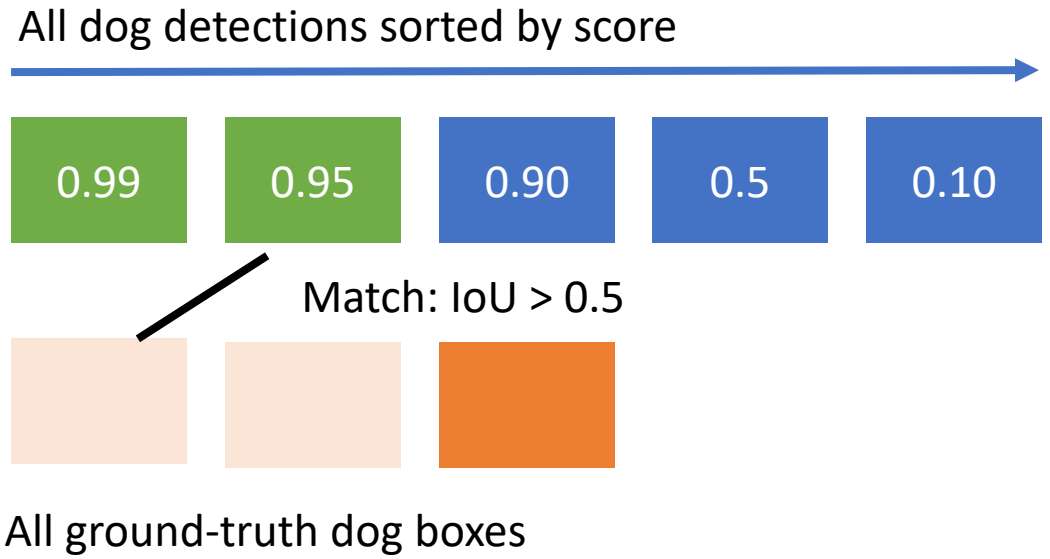


Precision = 1/1 = 1.0
Recall = 1/3 = 0.33

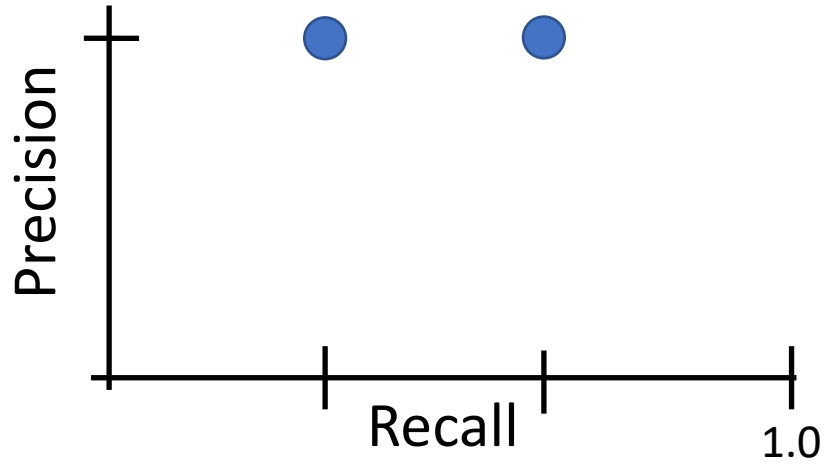


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

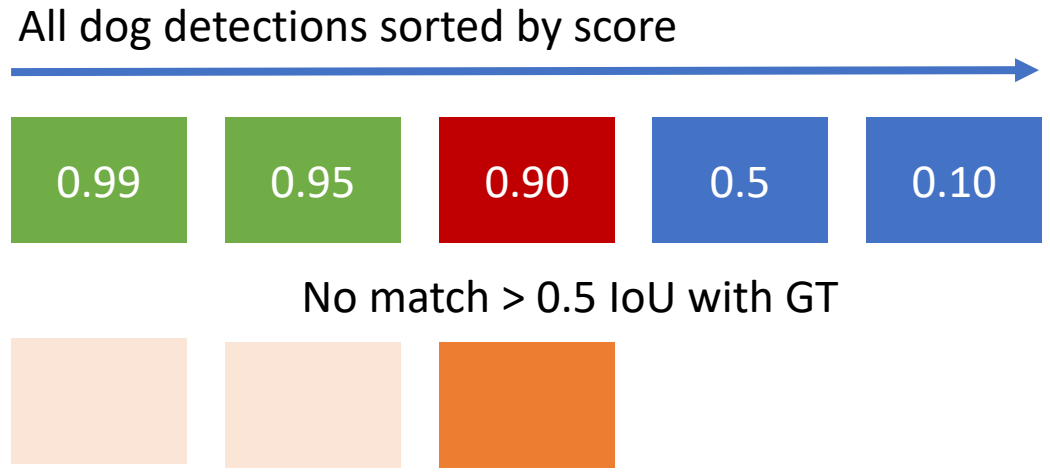


Precision = 2/2 = 1.0
Recall = 2/3 = 0.67



Evaluating Object Detectors: Mean Average Precision (mAP)

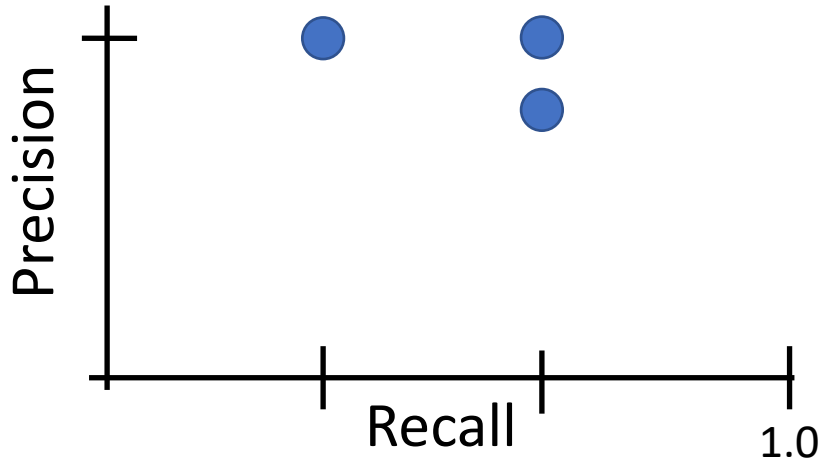
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



All ground-truth dog boxes

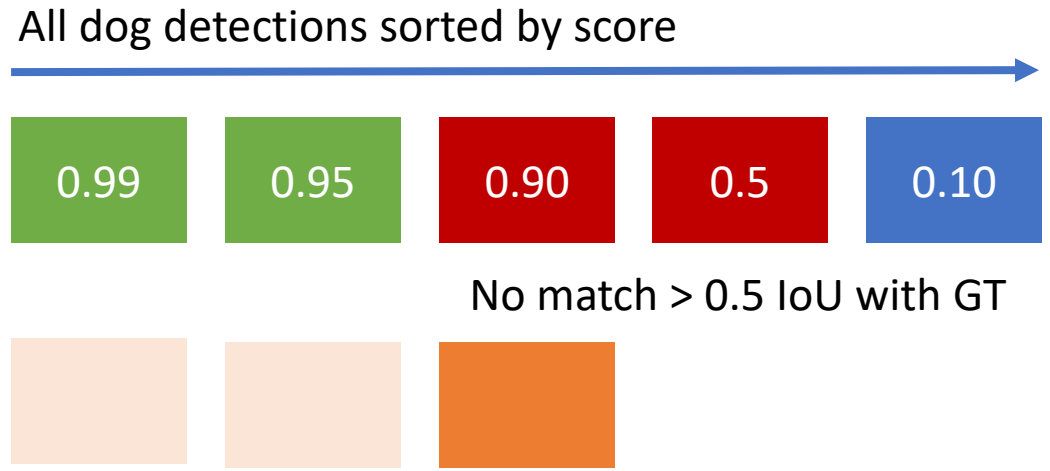
Precision = 2/3 = 0.67

Recall = 2/3 = 0.67

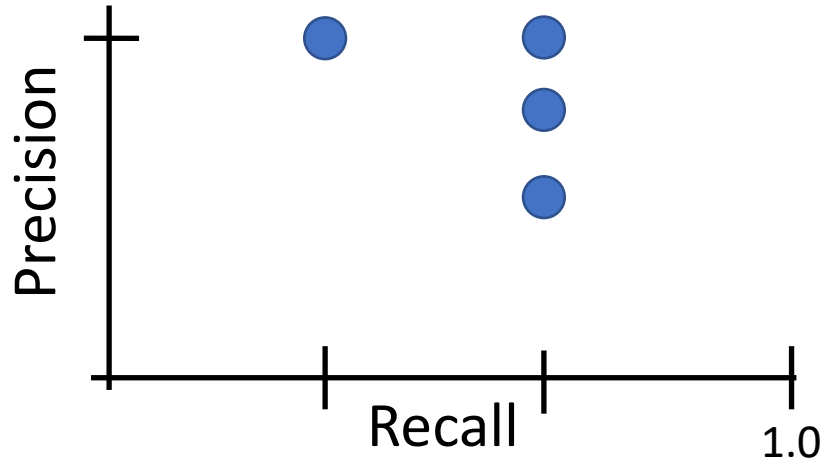


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

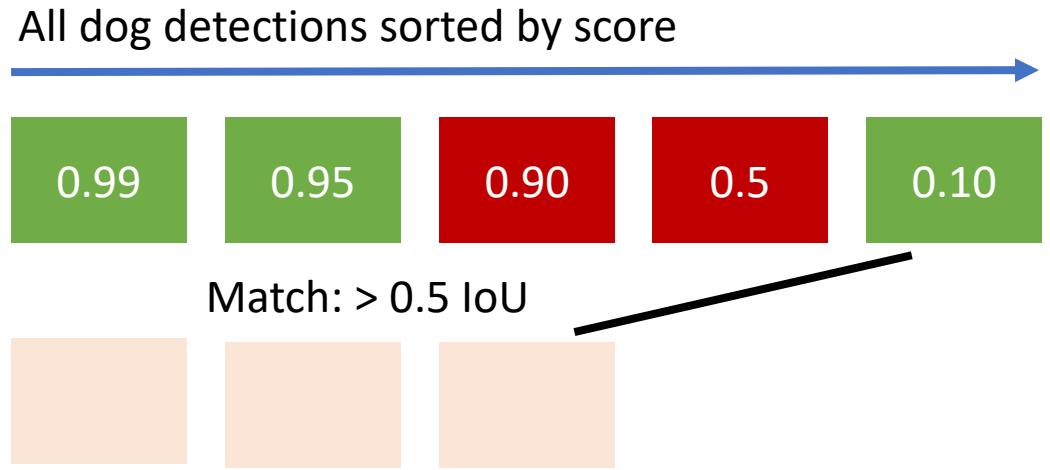


Precision = 2/4 = 0.5
Recall = 2/3 = 0.67

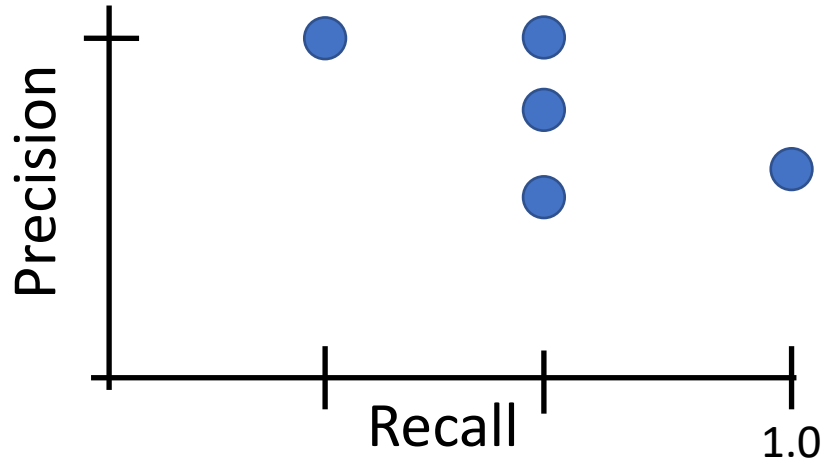


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



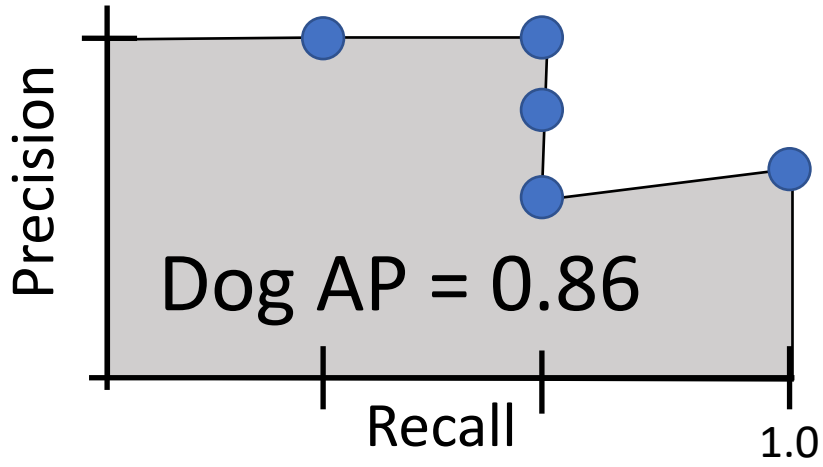
Precision = 3/5 = 0.6
Recall = 3/3 = 1.0



Evaluating Object Detectors: Mean Average Precision (mAP)



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

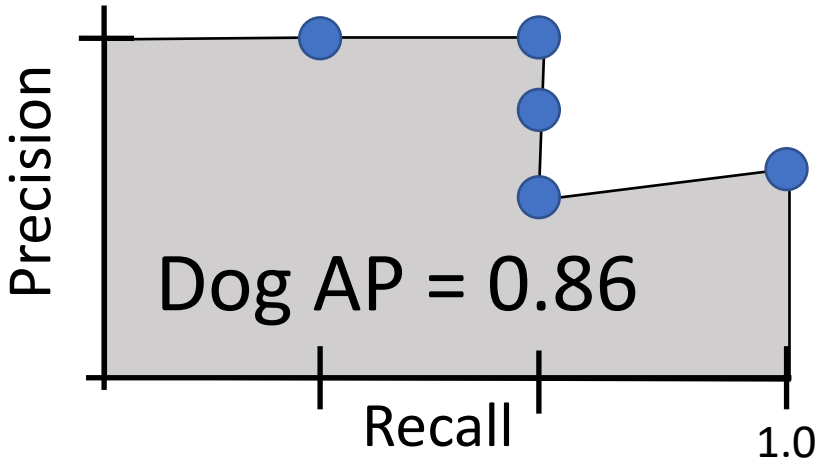


Evaluating Object Detectors: Mean Average Precision (mAP)



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

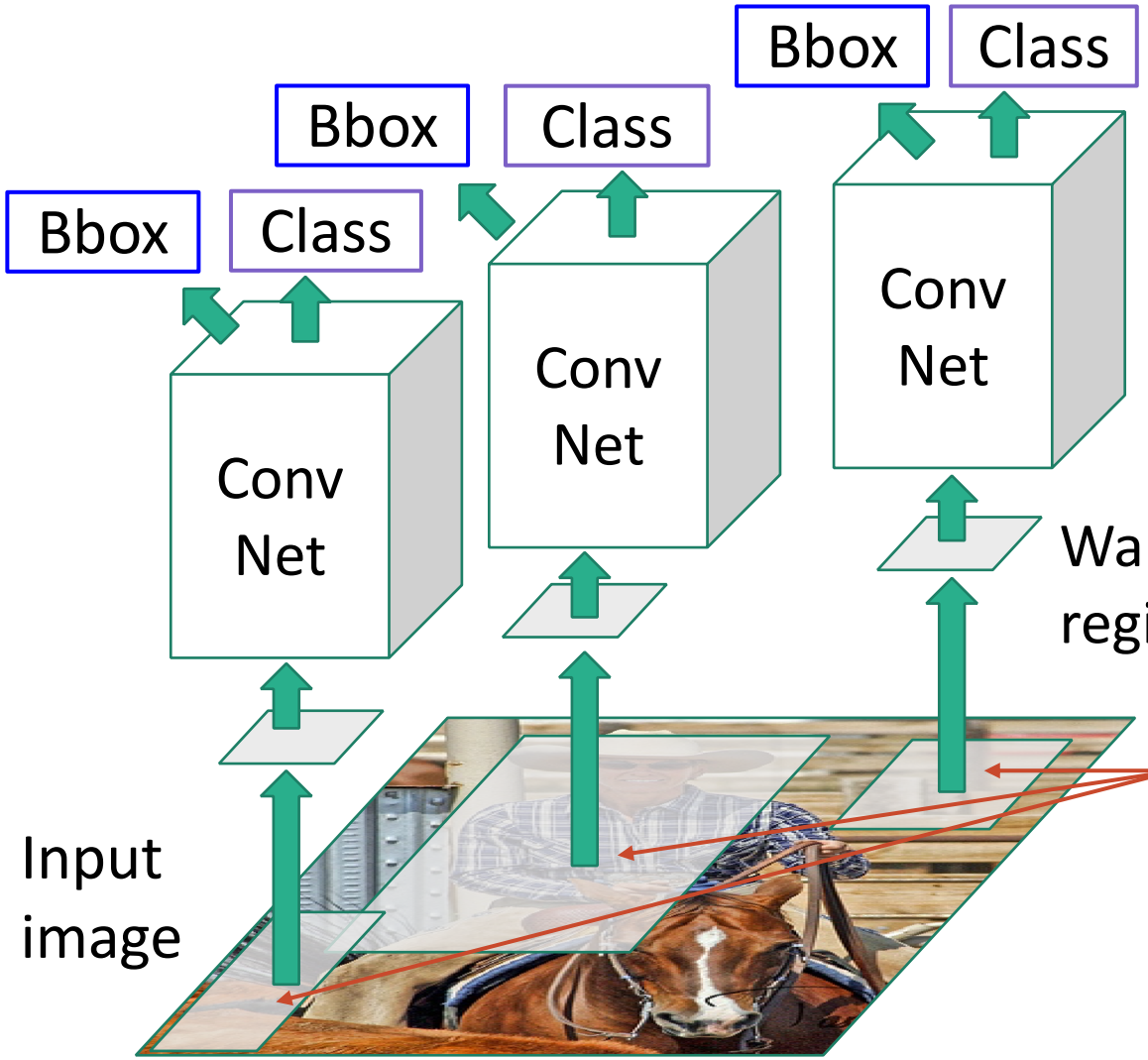
mAP@0.95 = 0.2

COCO mAP = 0.4

R-CNN: Region-Based CNN

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)



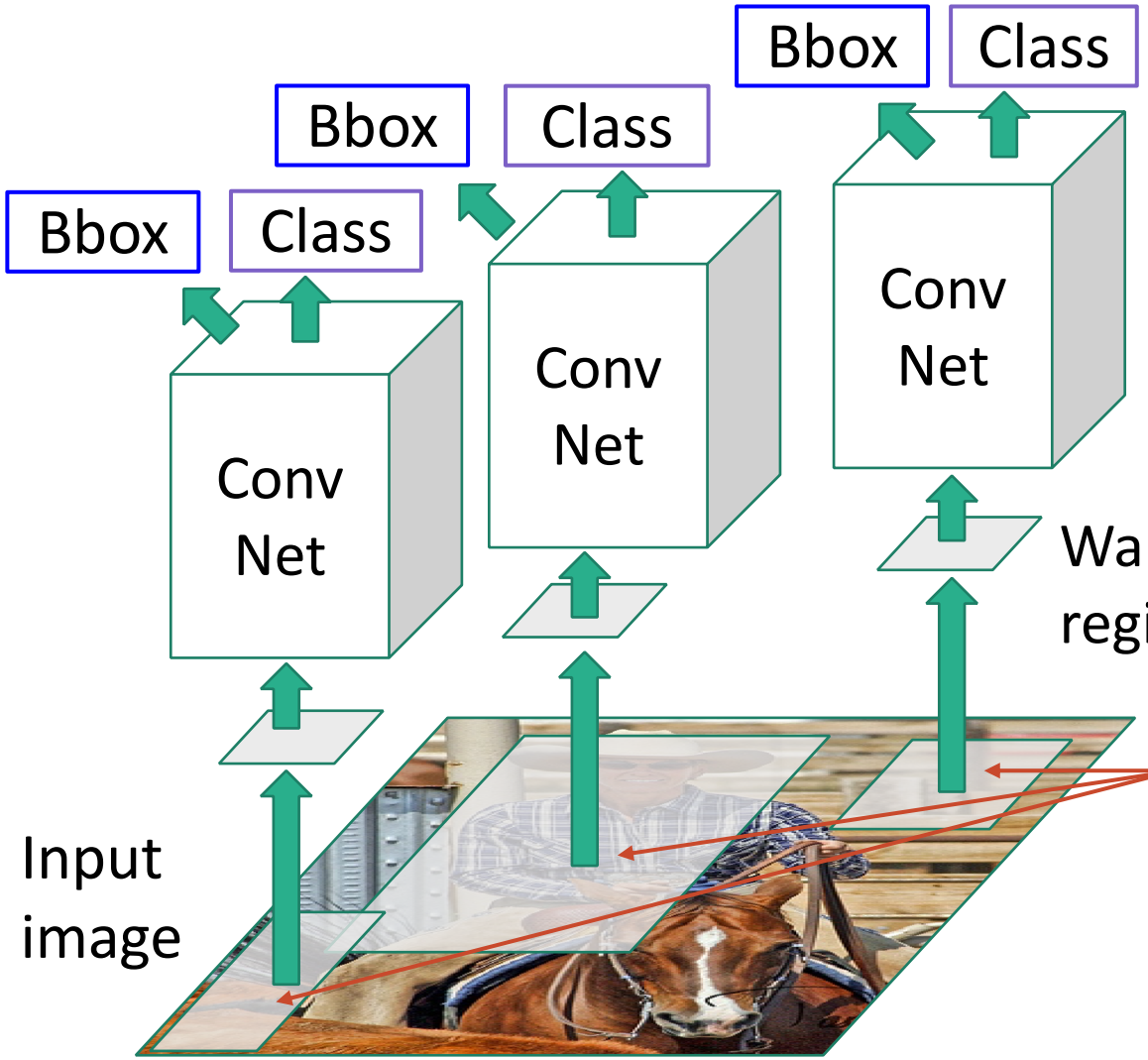
Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

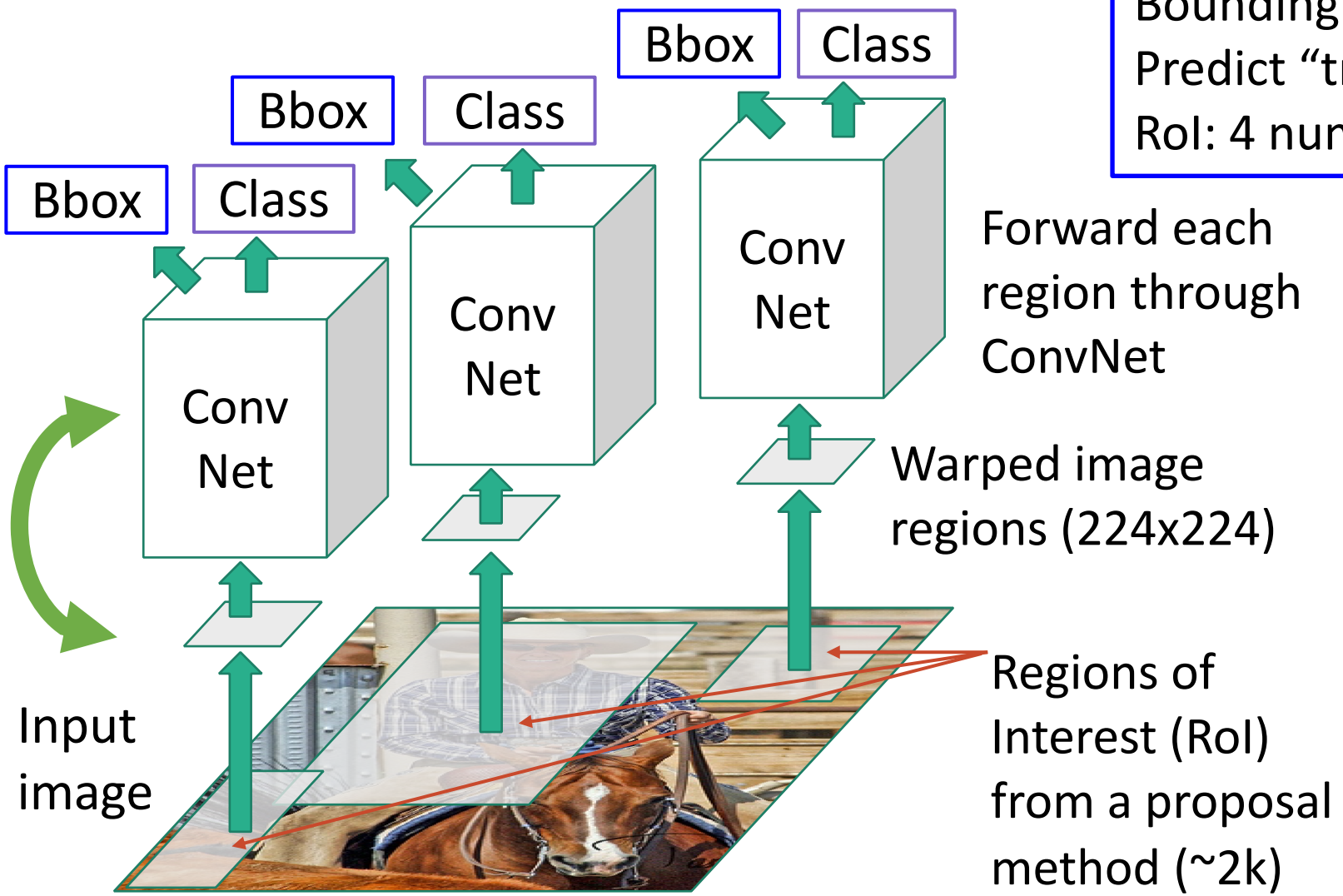
Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow!
Need to do ~2k forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Region-Based CNN



Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

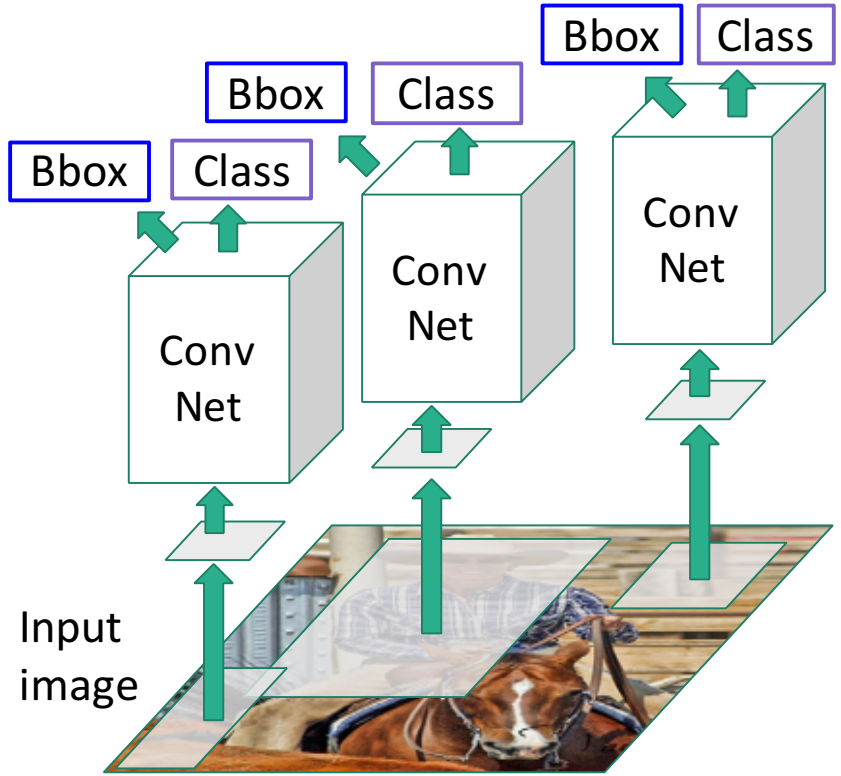
Problem: Very slow! Need to do ~2k forward passes for each image!

Solution: Run CNN *before* warping!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

“Slow” R-CNN

Process each region independently



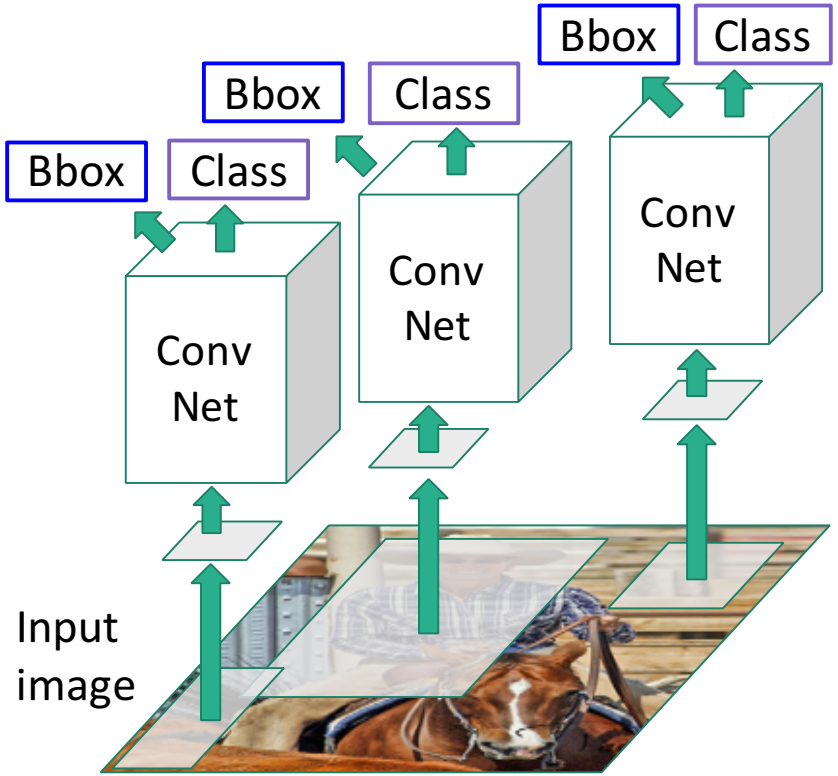
Fast R-CNN



Input image

"Slow" R-CNN

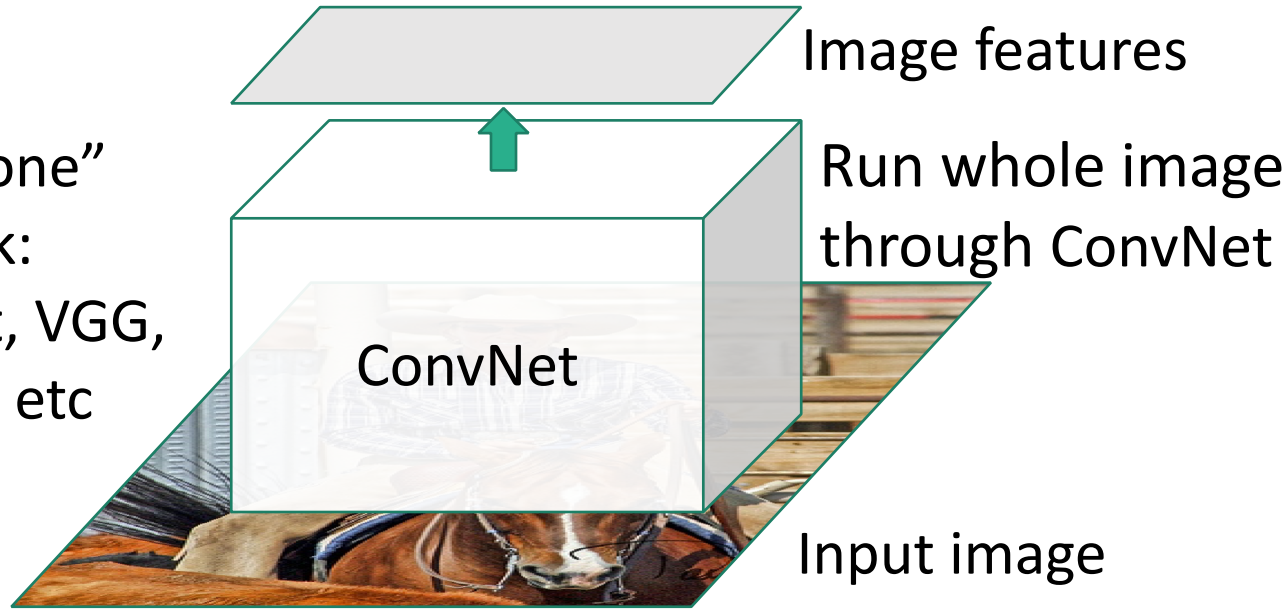
Process each region independently



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

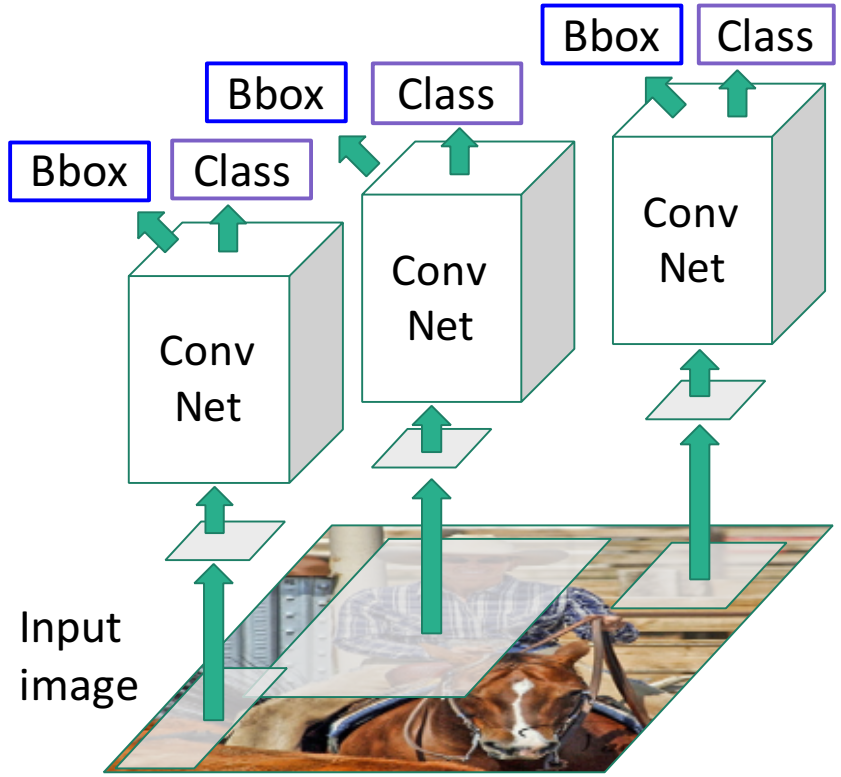
Fast R-CNN

"Backbone" network: AlexNet, VGG, ResNet, etc



"Slow" R-CNN

Process each region independently

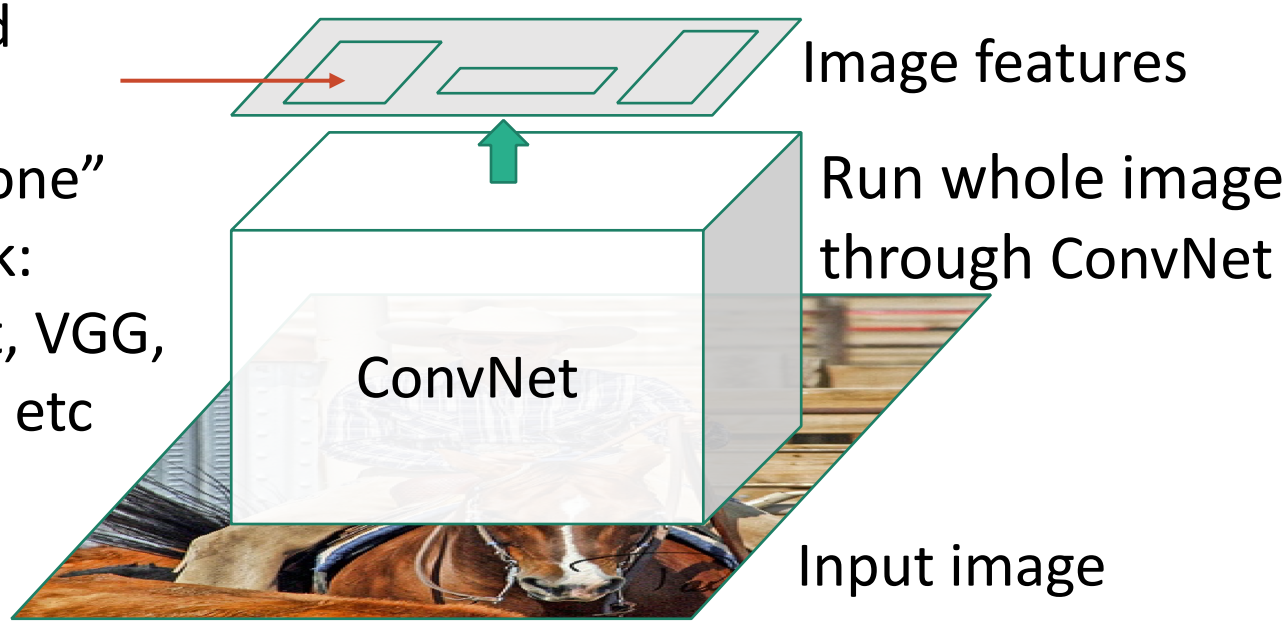


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

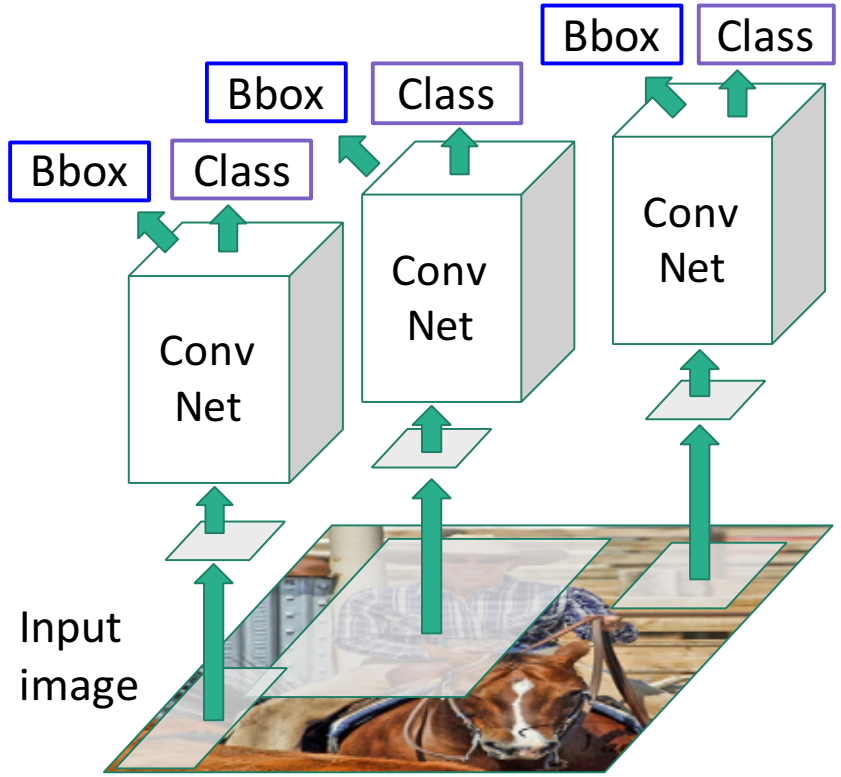
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



"Slow" R-CNN

Process each region independently

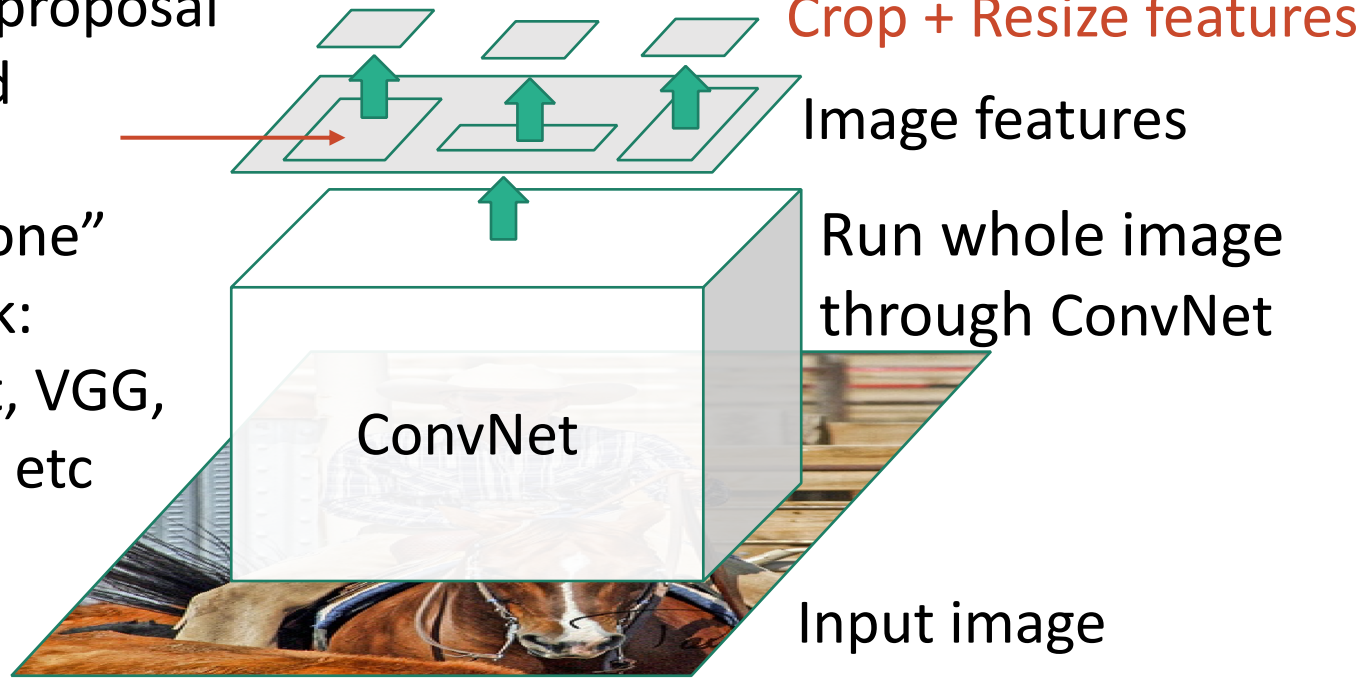


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

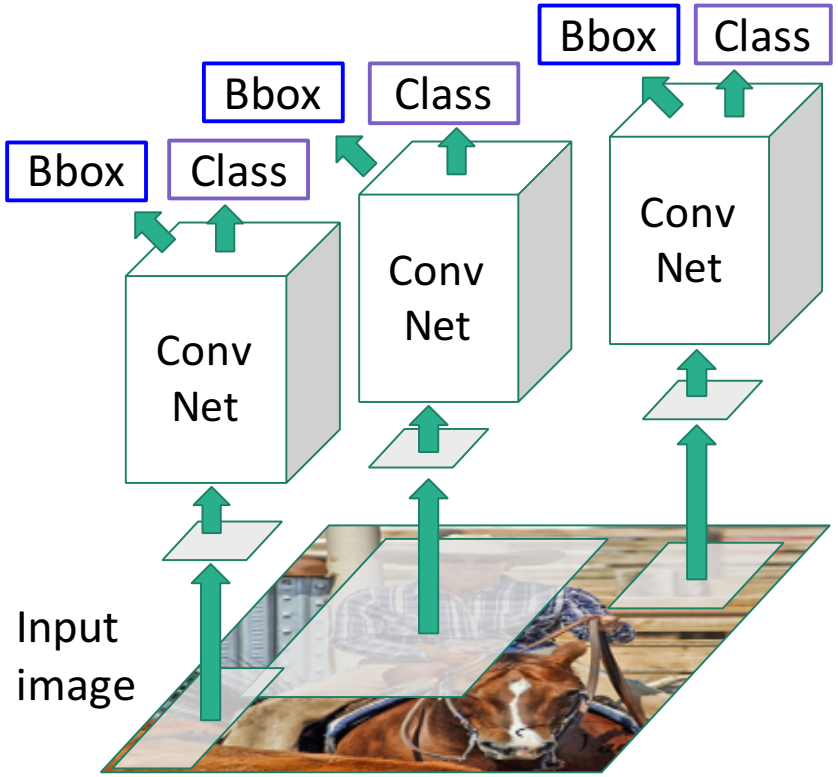
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



"Slow" R-CNN

Process each region independently

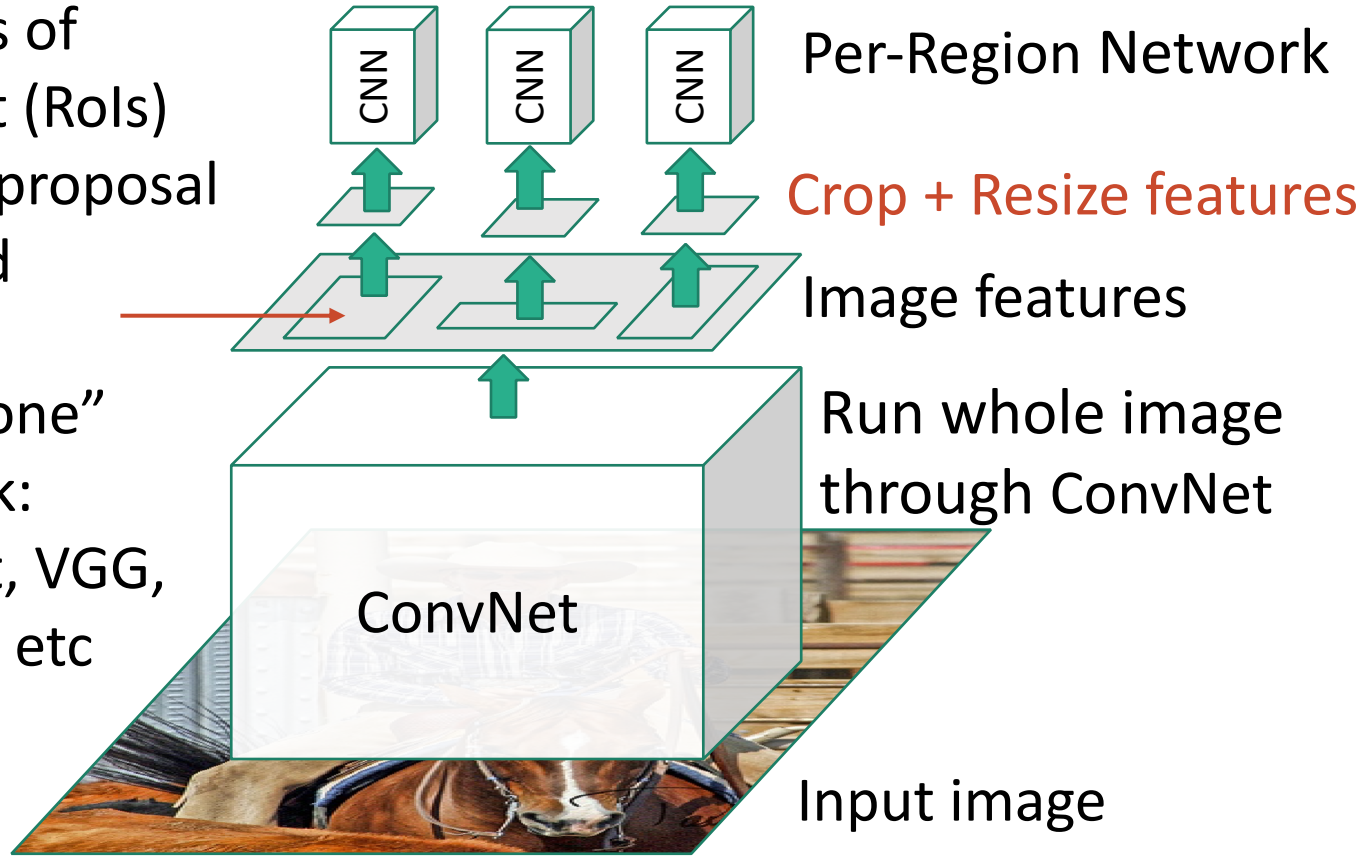


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

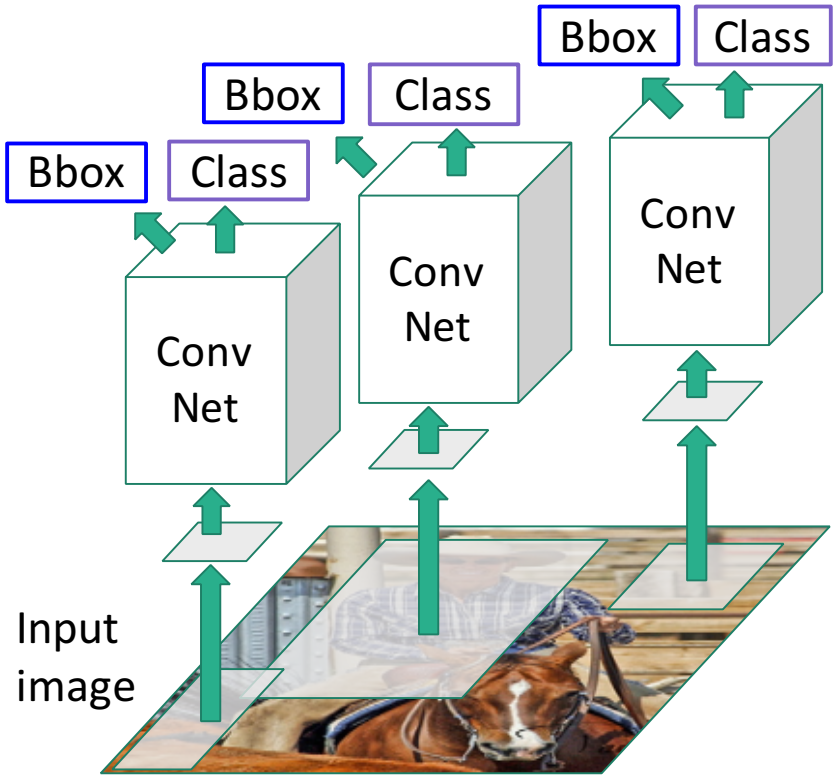
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



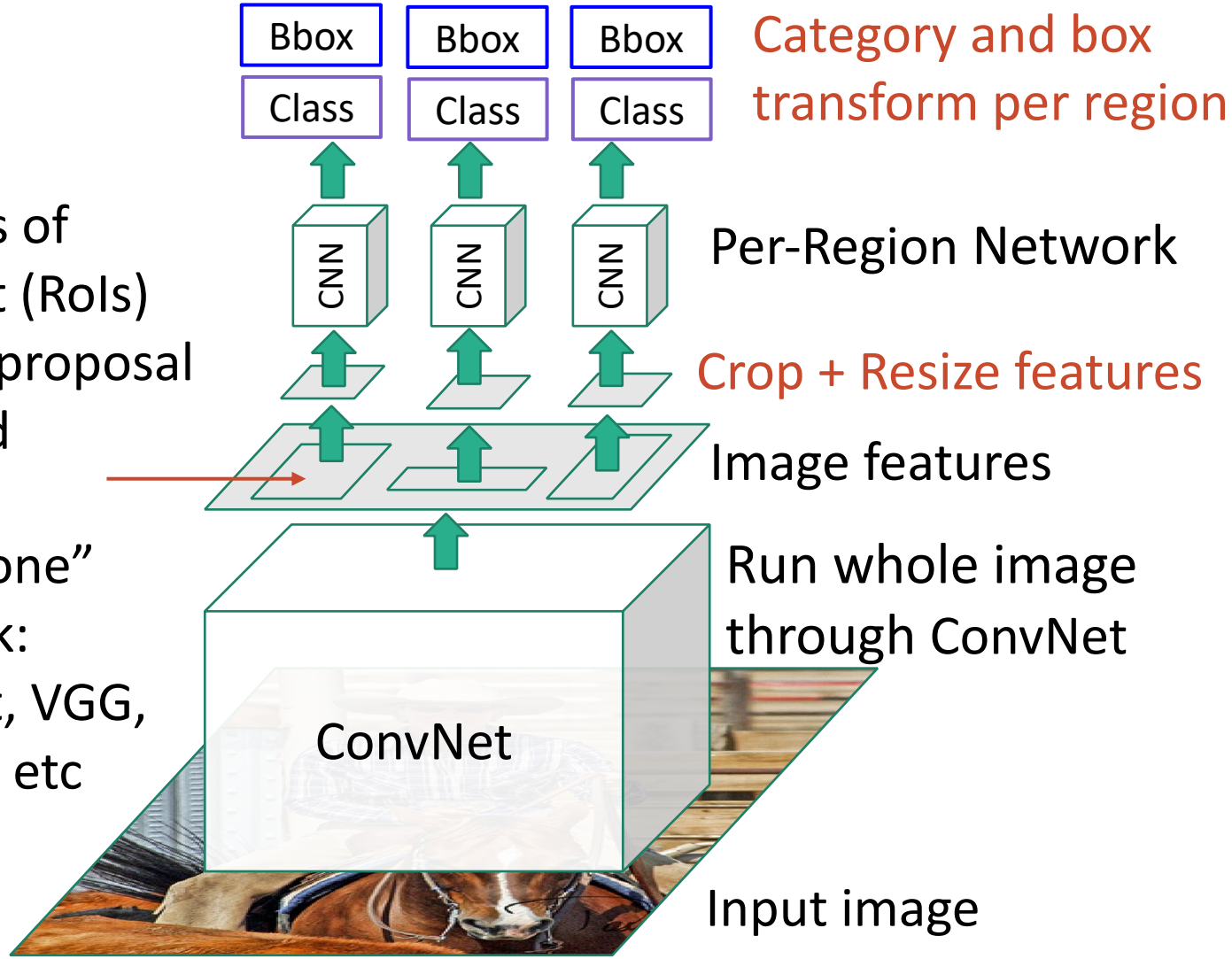
"Slow" R-CNN

Process each region independently



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

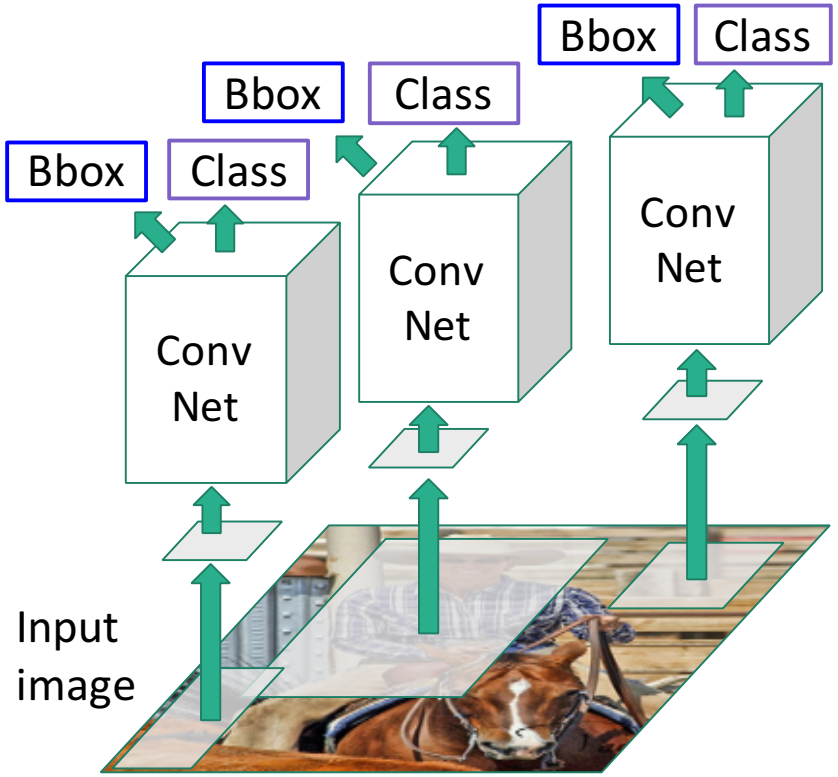


Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

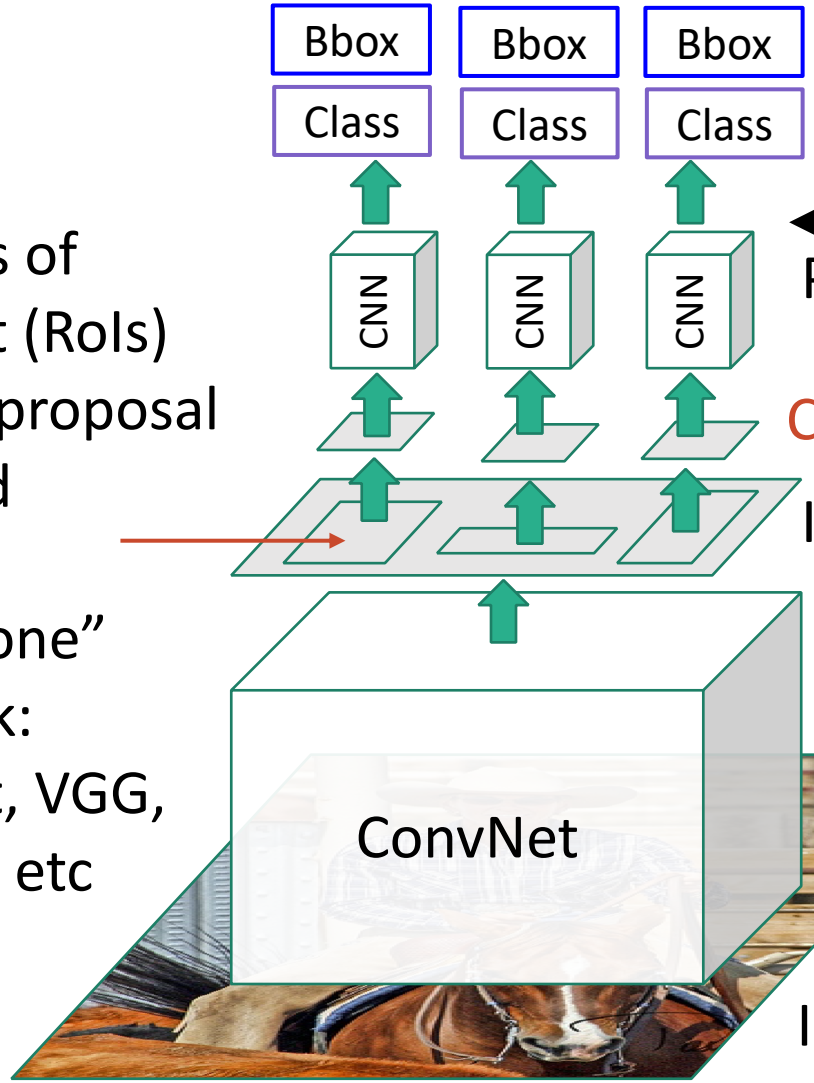
"Slow" R-CNN

Process each region independently



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



Category and box transform per region

Per-Region network is relatively lightweight

Per-Region Network

Regions of Interest (RoIs) from a proposal method

Crop + Resize features

Image features

"Backbone" network: AlexNet, VGG, ResNet, etc

Run whole image through ConvNet

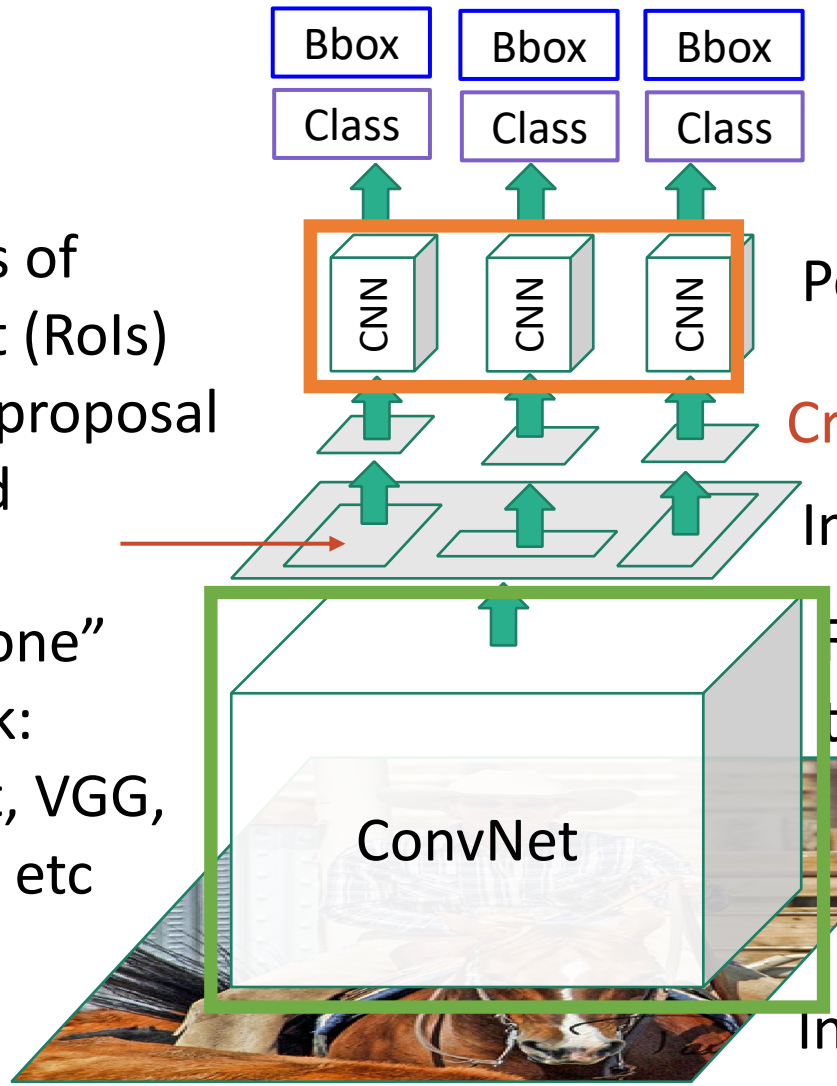
ConvNet

Most of the computation happens in backbone network; this saves work for overlapping region proposals

Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



Category and box transform per region

Per-Region Network

Crop + Resize features

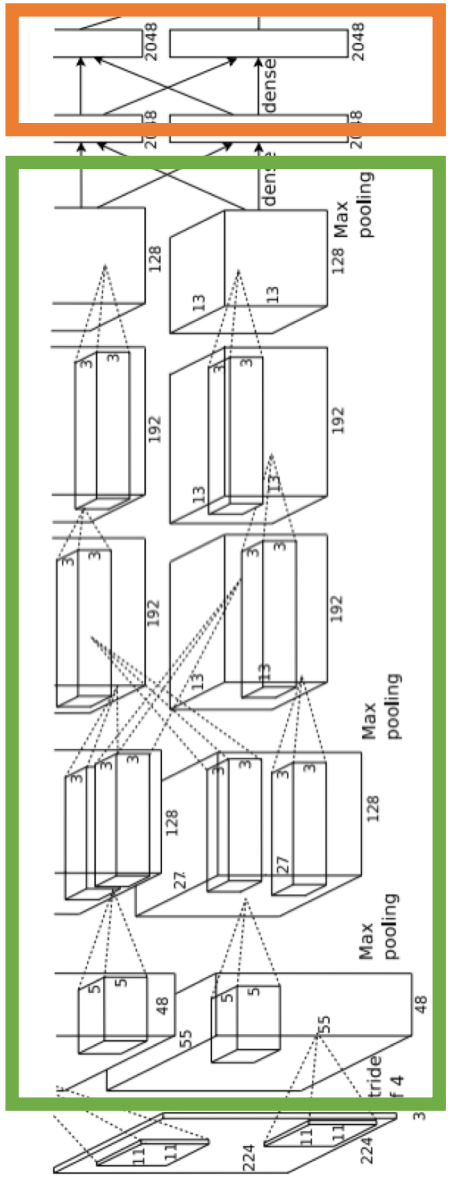
Image features

Run whole image through ConvNet

Input image

Regions of Interest (RoIs) from a proposal method

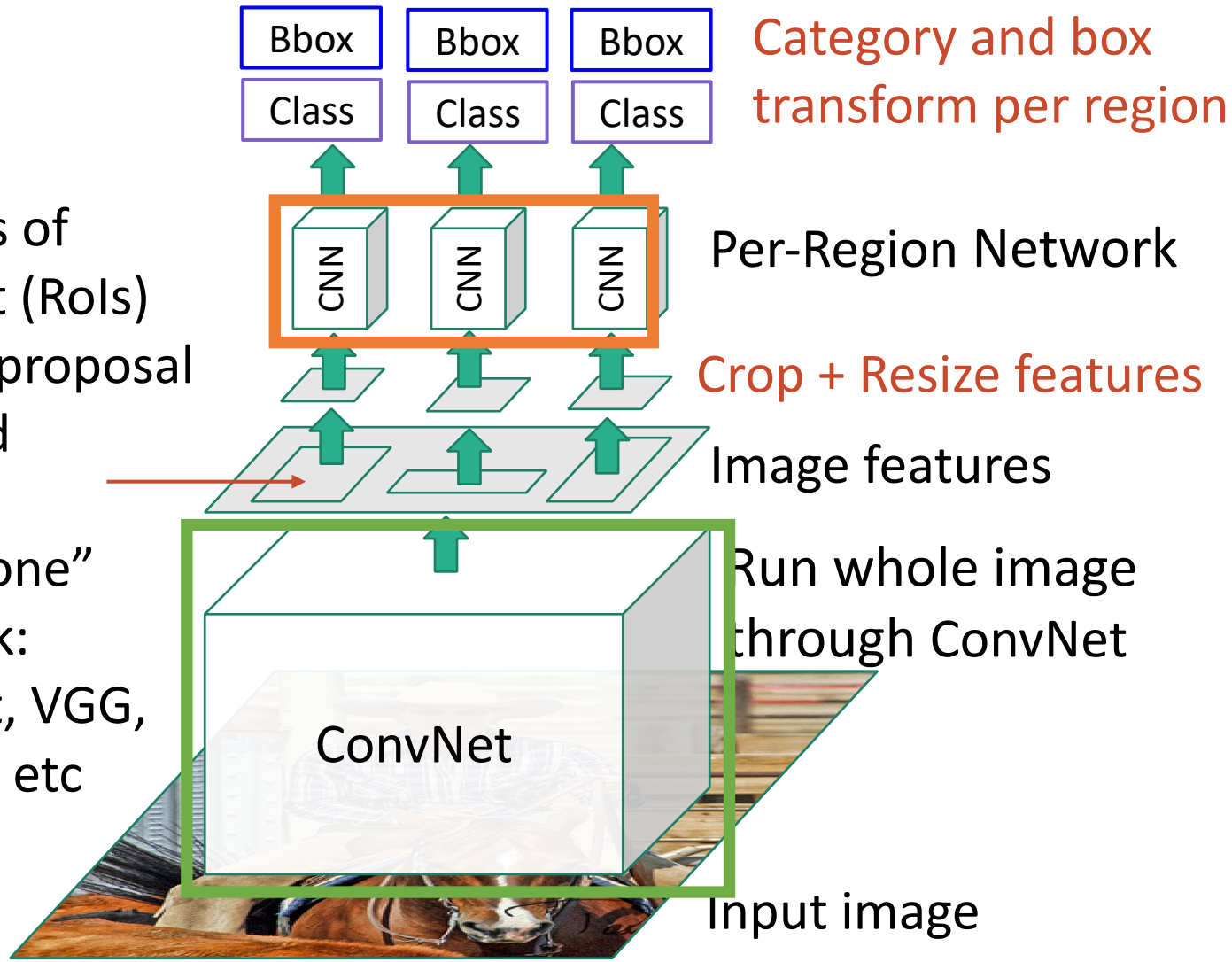
"Backbone" network: AlexNet, VGG, ResNet, etc



Example: When using AlexNet for detection, five conv layers are used for backbone and two FC layers are used for per-region network

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



Category and box transform per region

Per-Region Network

Crop + Resize features

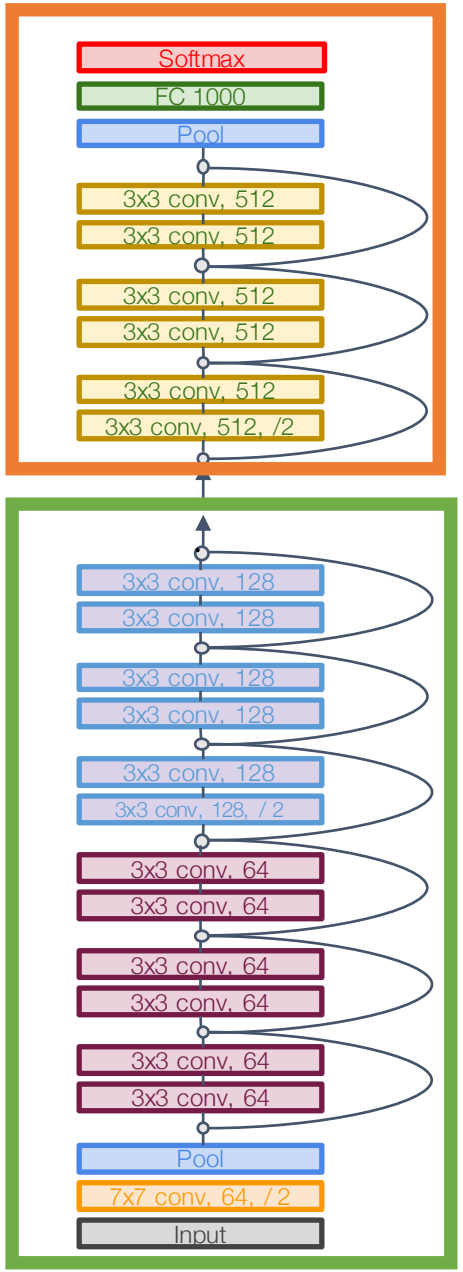
Image features

Run whole image through ConvNet

Input image

Regions of Interest (RoIs) from a proposal method

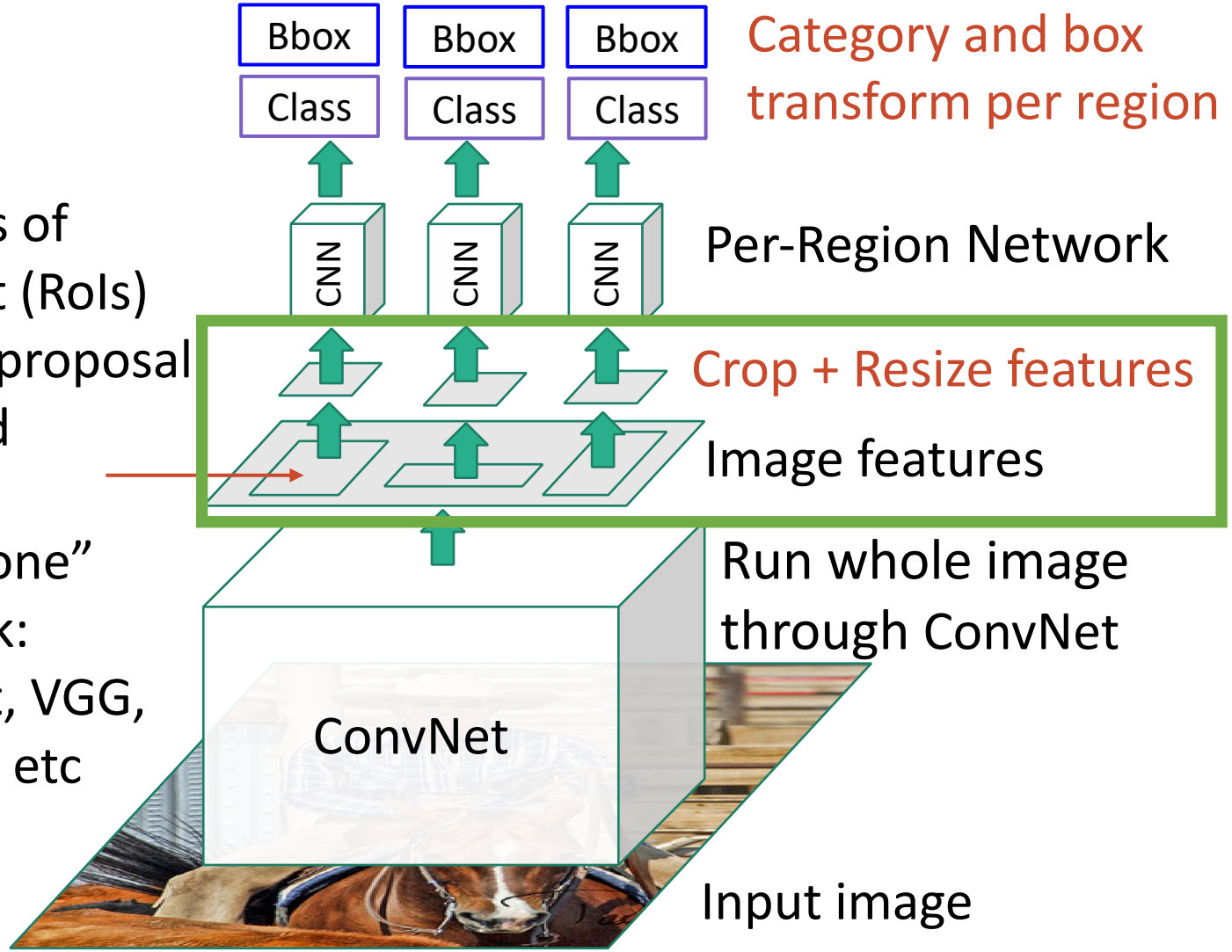
"Backbone" network: AlexNet, VGG, ResNet, etc



Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



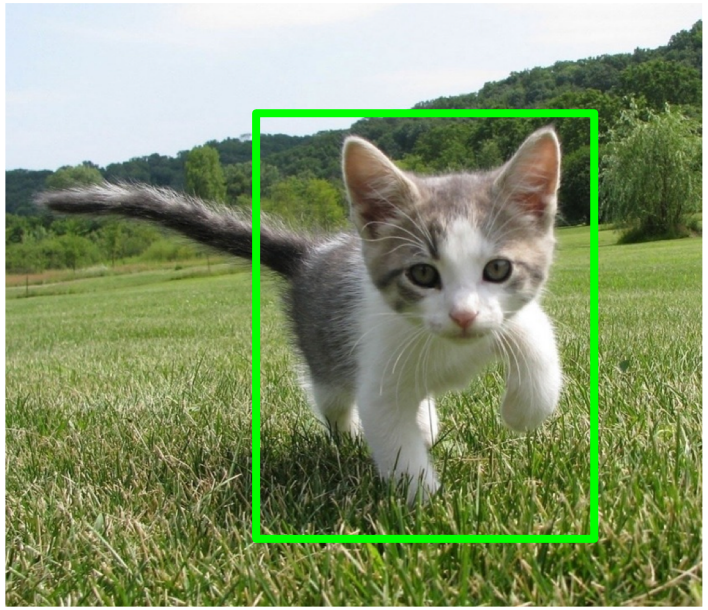
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

How to crop features?

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

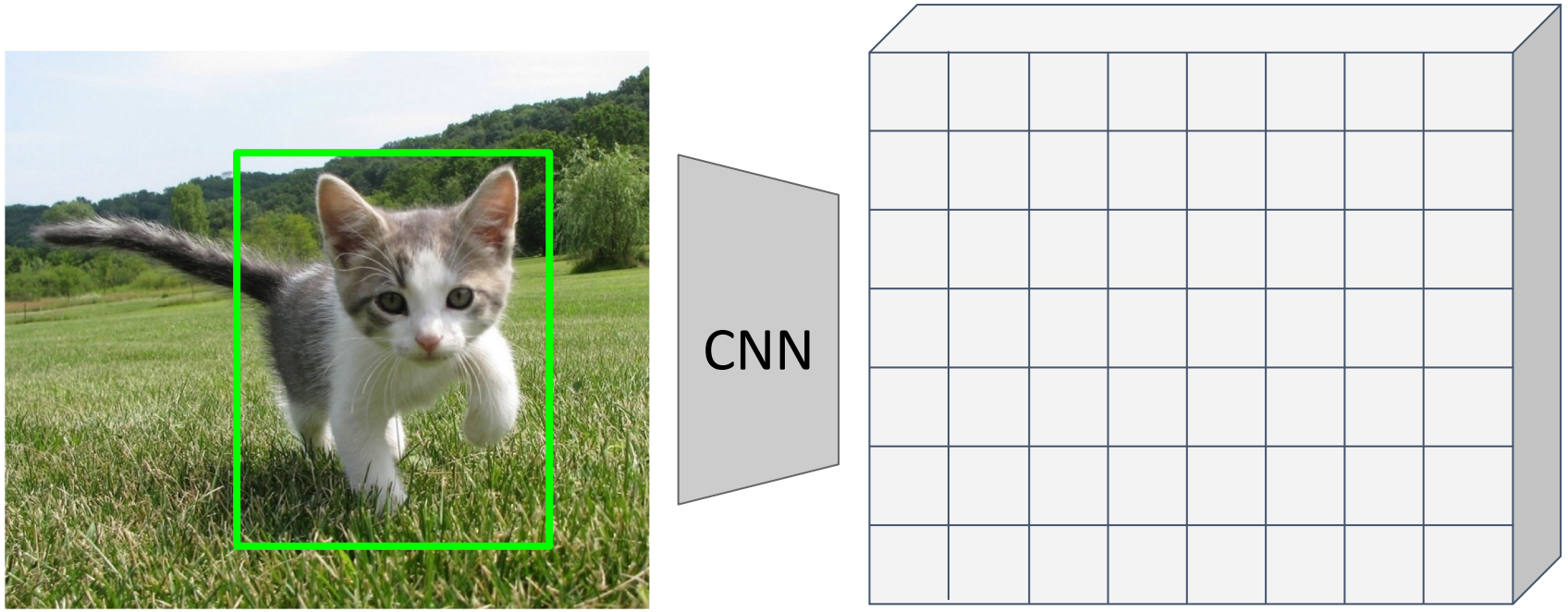
Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool

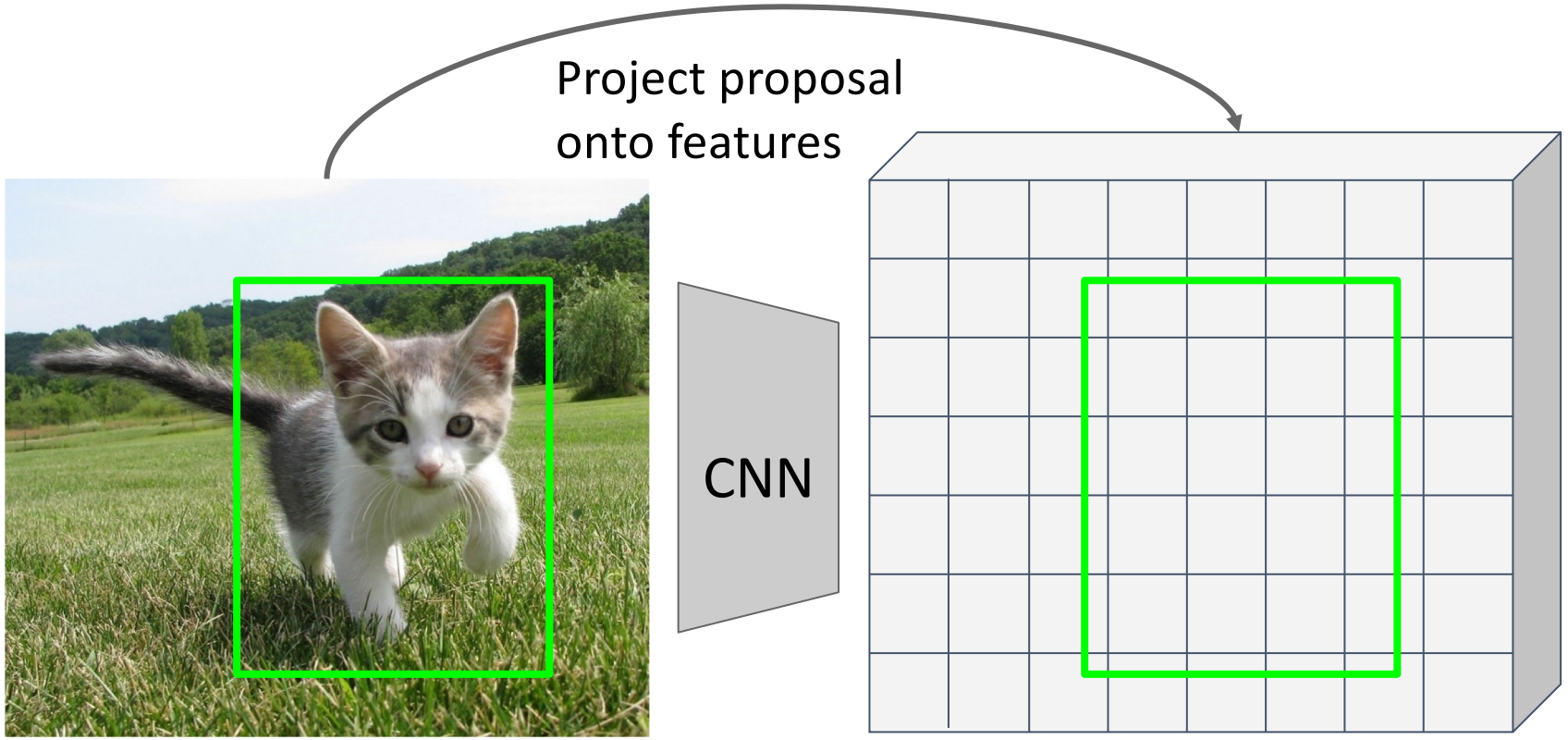


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

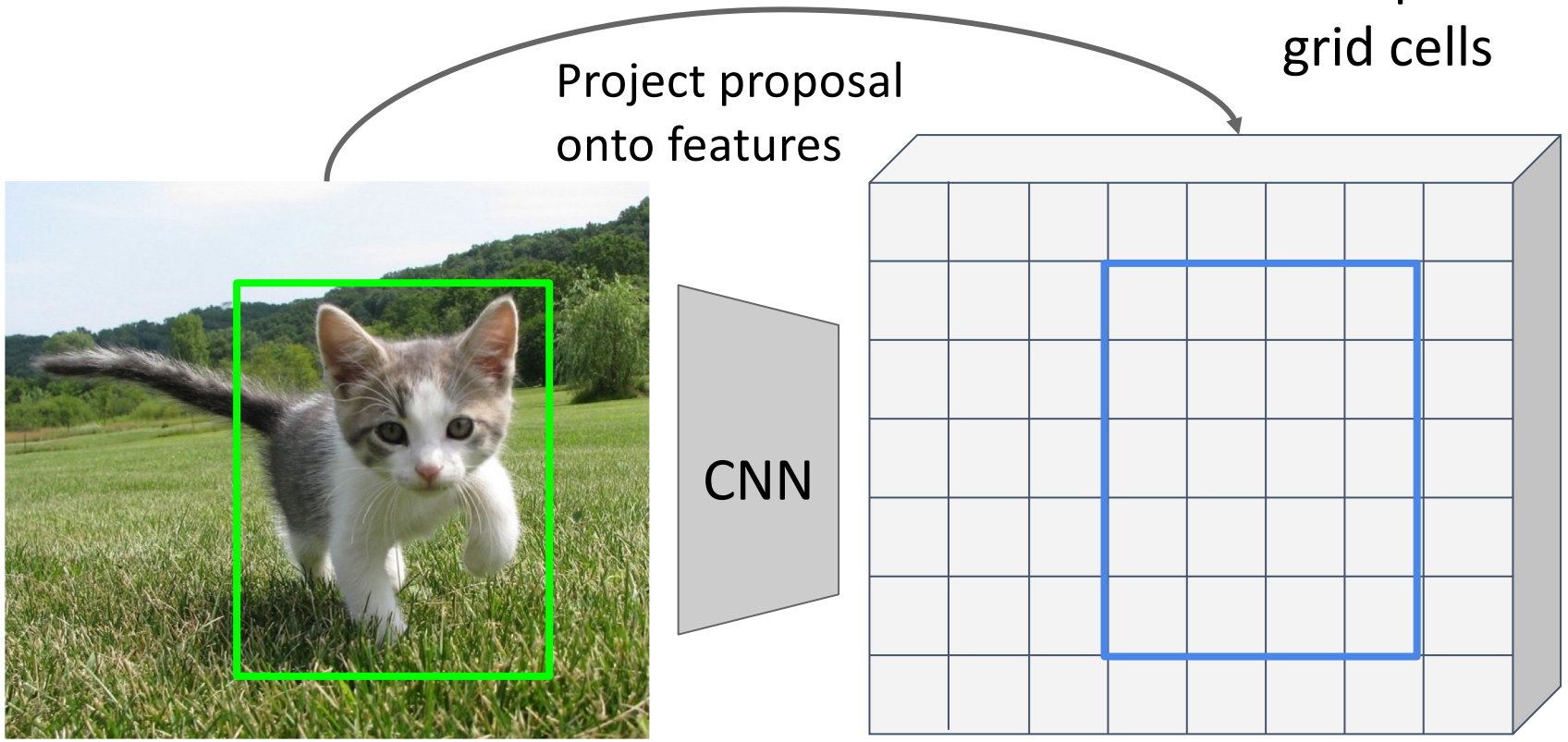
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



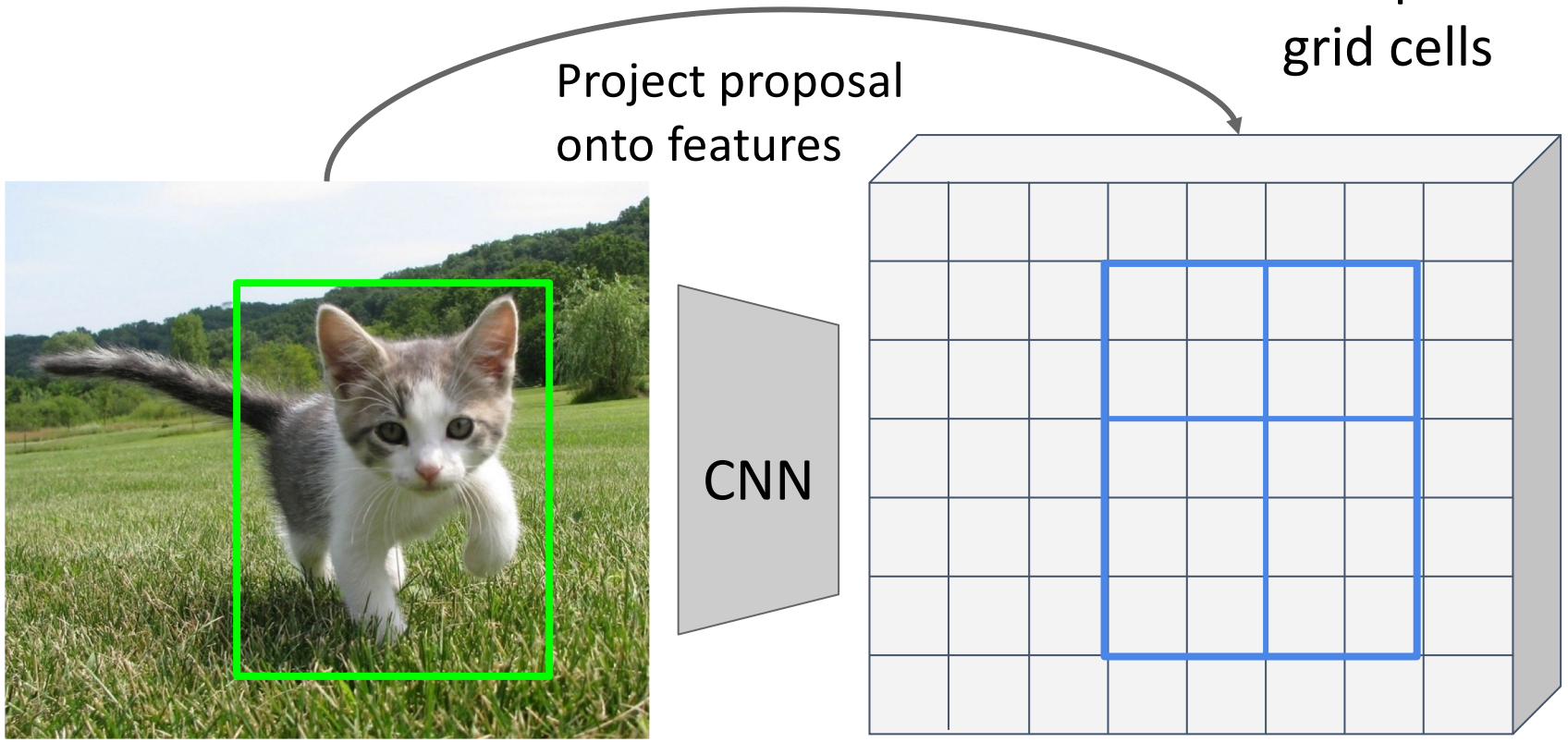
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



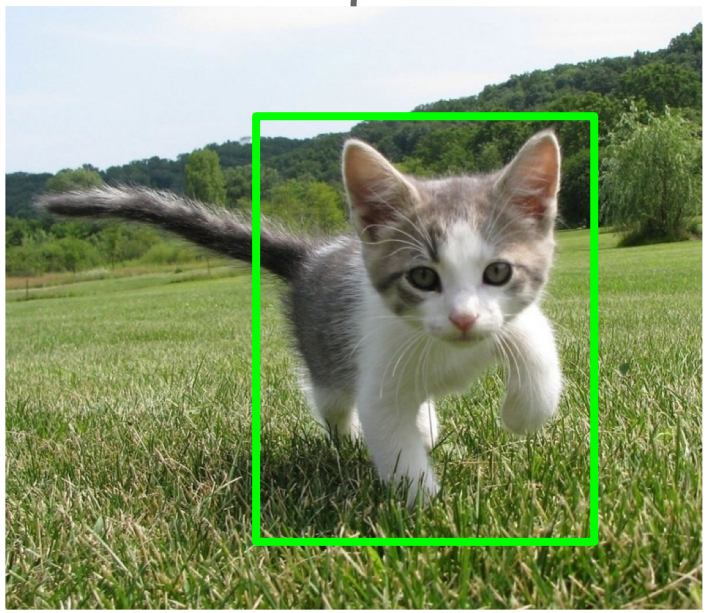
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

CNN

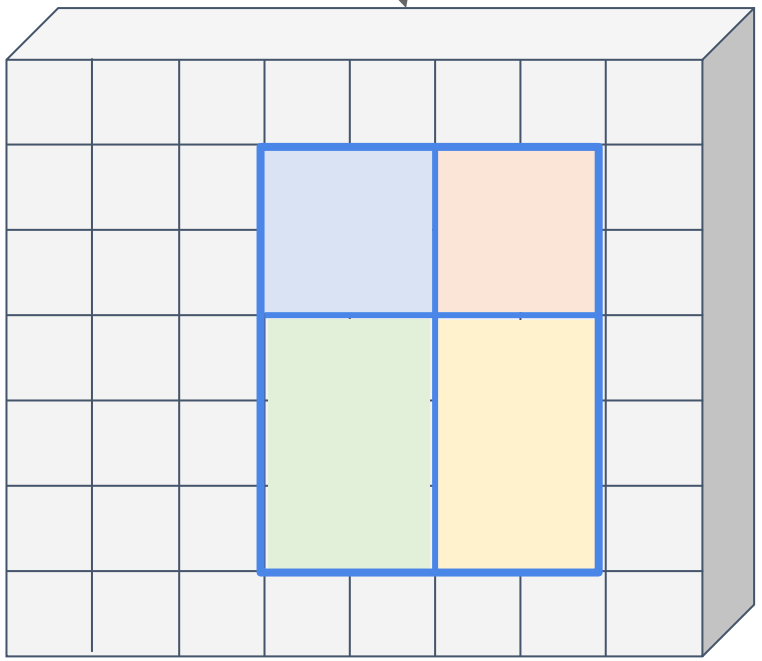
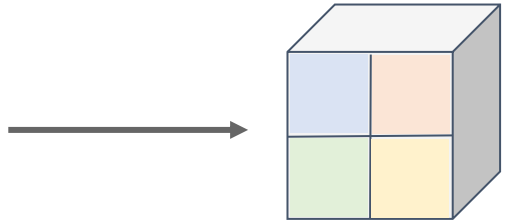


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

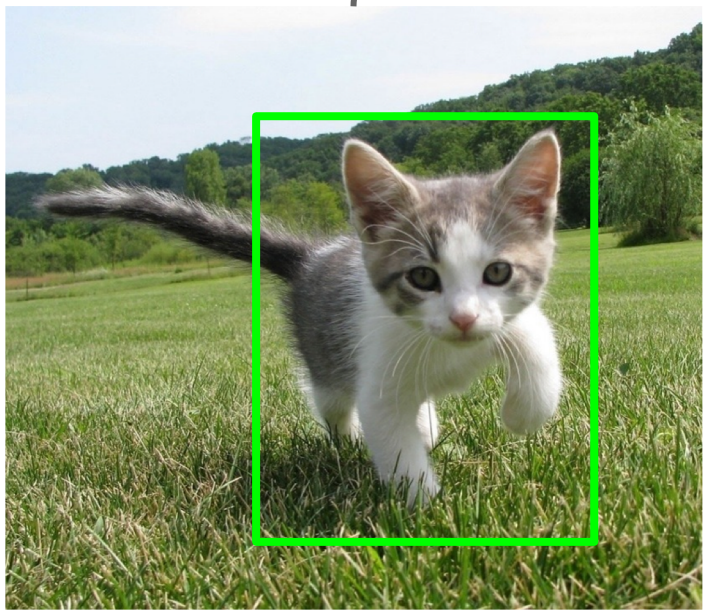


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

CNN

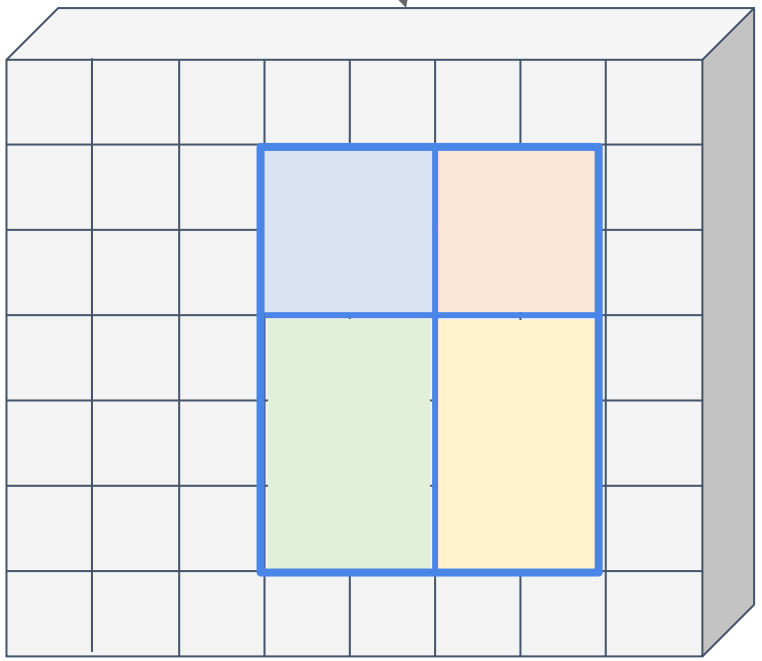
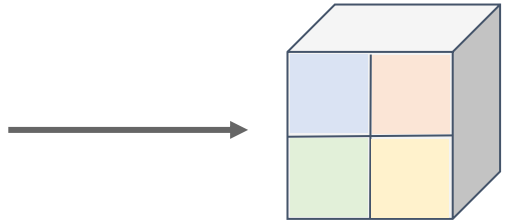


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

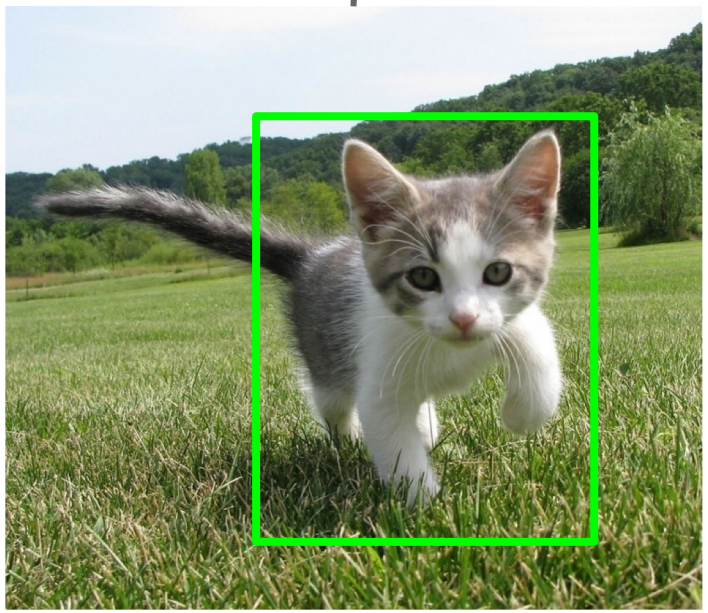


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

Project proposal onto features

CNN

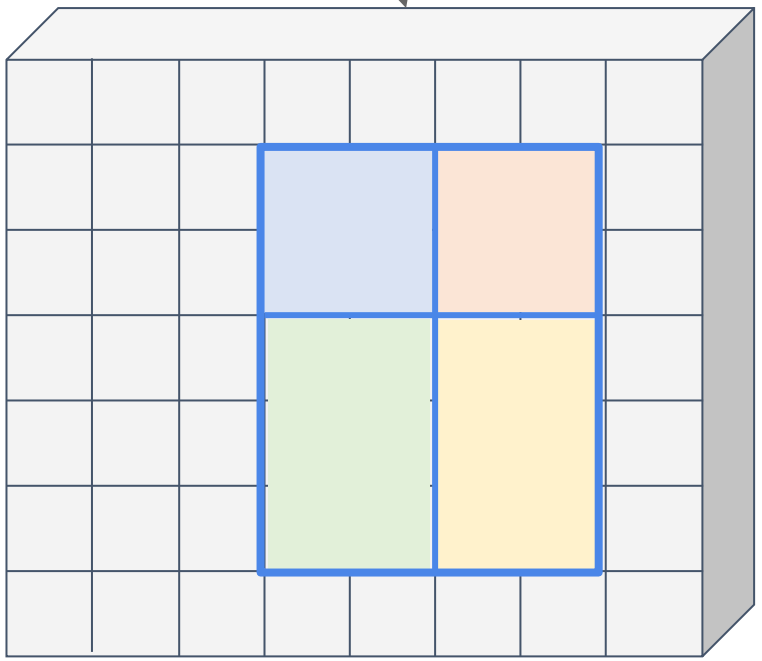
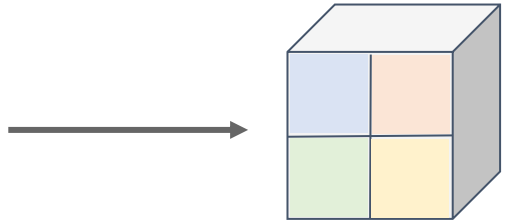


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion



Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Problem: Slight misalignment due to snapping; different-sized subregions is weird

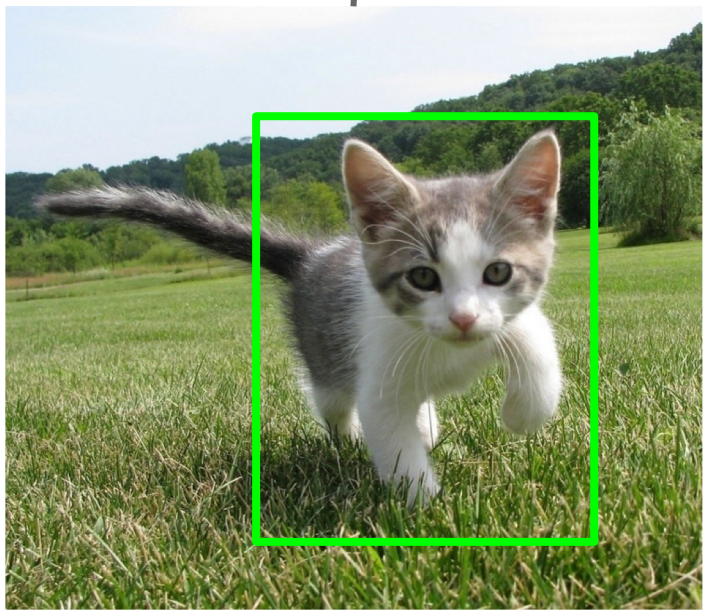
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Align

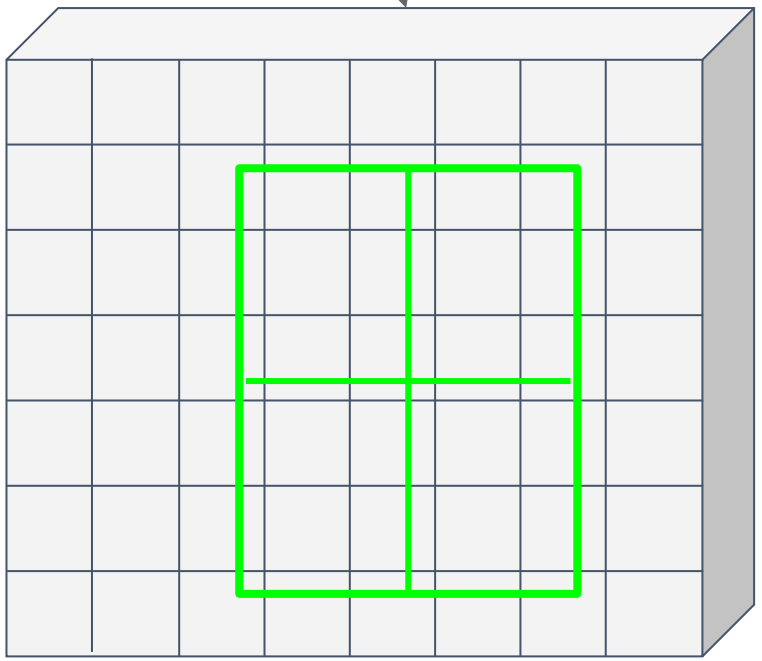
Divide into equal-sized subregions
(may not be aligned to grid!)

No “snapping”!

Project proposal
onto features



CNN



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

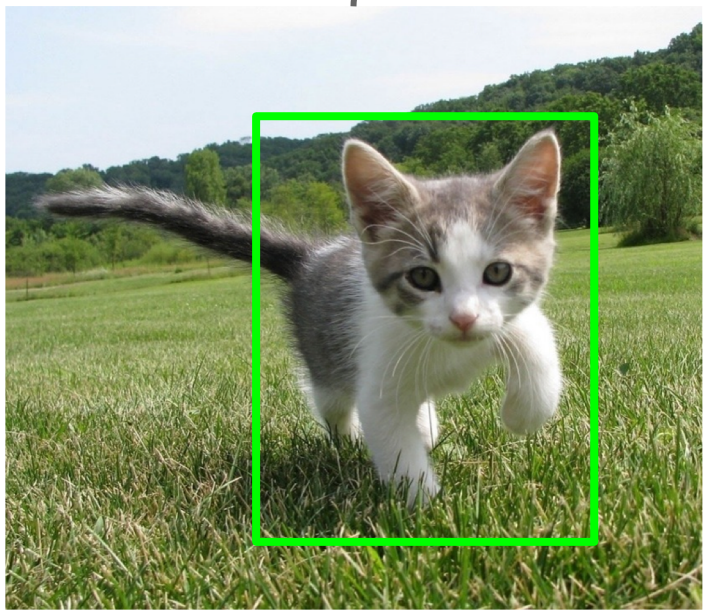
He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align

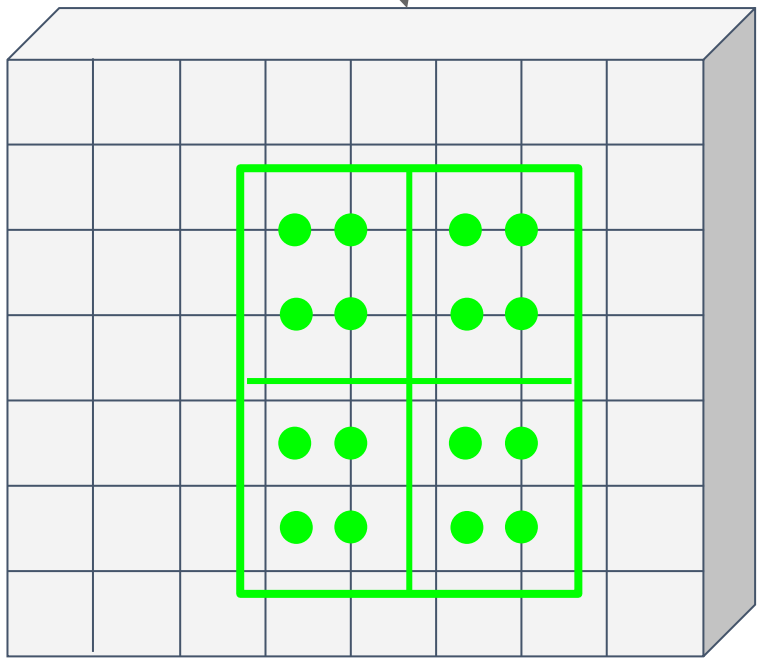
Divide into equal-sized subregions
(may not be aligned to grid!)

No “snapping”!

Project proposal
onto features



CNN



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

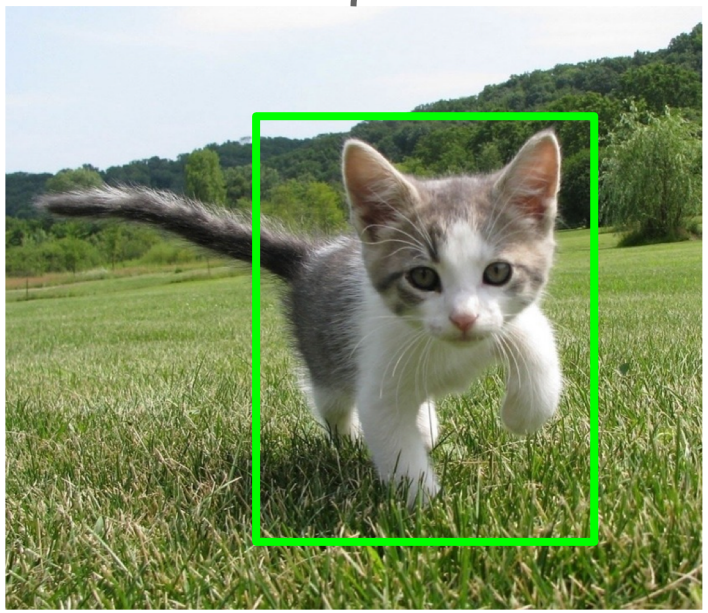
He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align

Divide into equal-sized subregions
(may not be aligned to grid!)

No "snapping"!

Project proposal
onto features



Input Image
(e.g. 3 x 640 x 480)

CNN

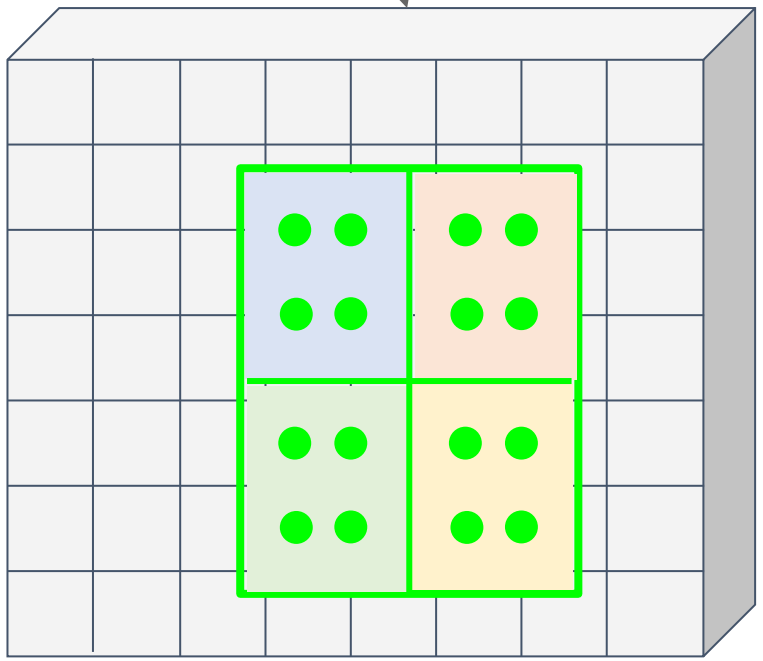
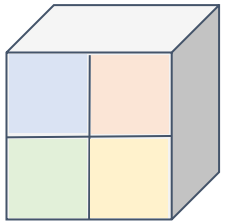


Image features
(e.g. 512 x 20 x 15)

Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation

After sampling, max-
pool in each subregion

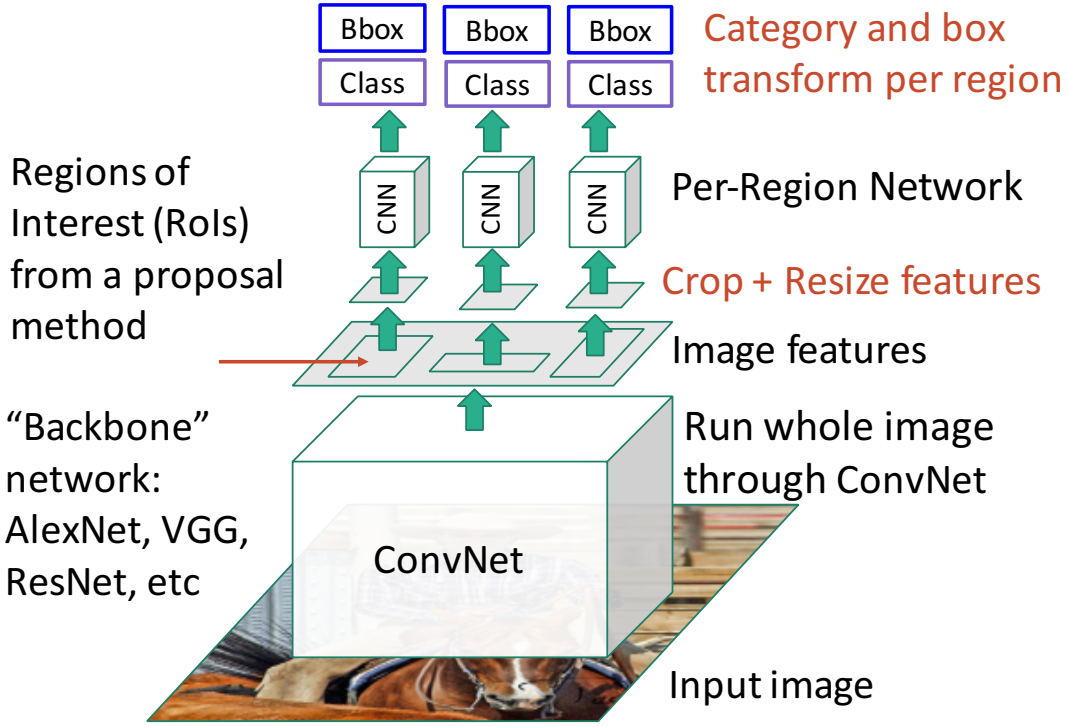


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

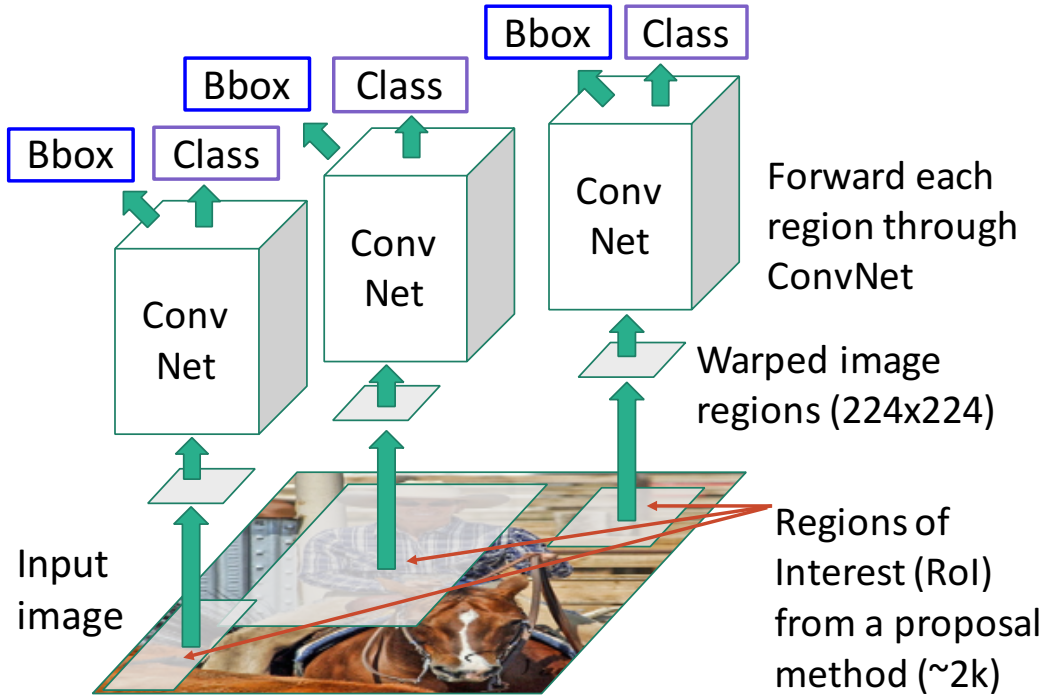
He et al, "Mask R-CNN", ICCV 2017

Fast R-CNN vs "Slow" R-CNN

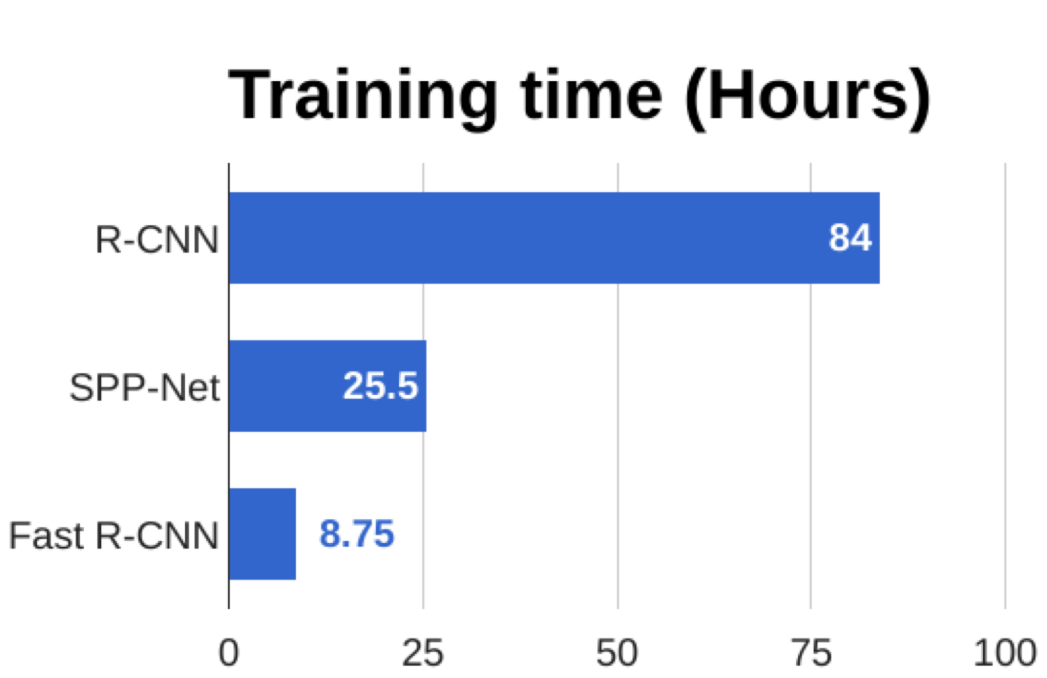
Fast R-CNN: Apply differentiable cropping to shared image features

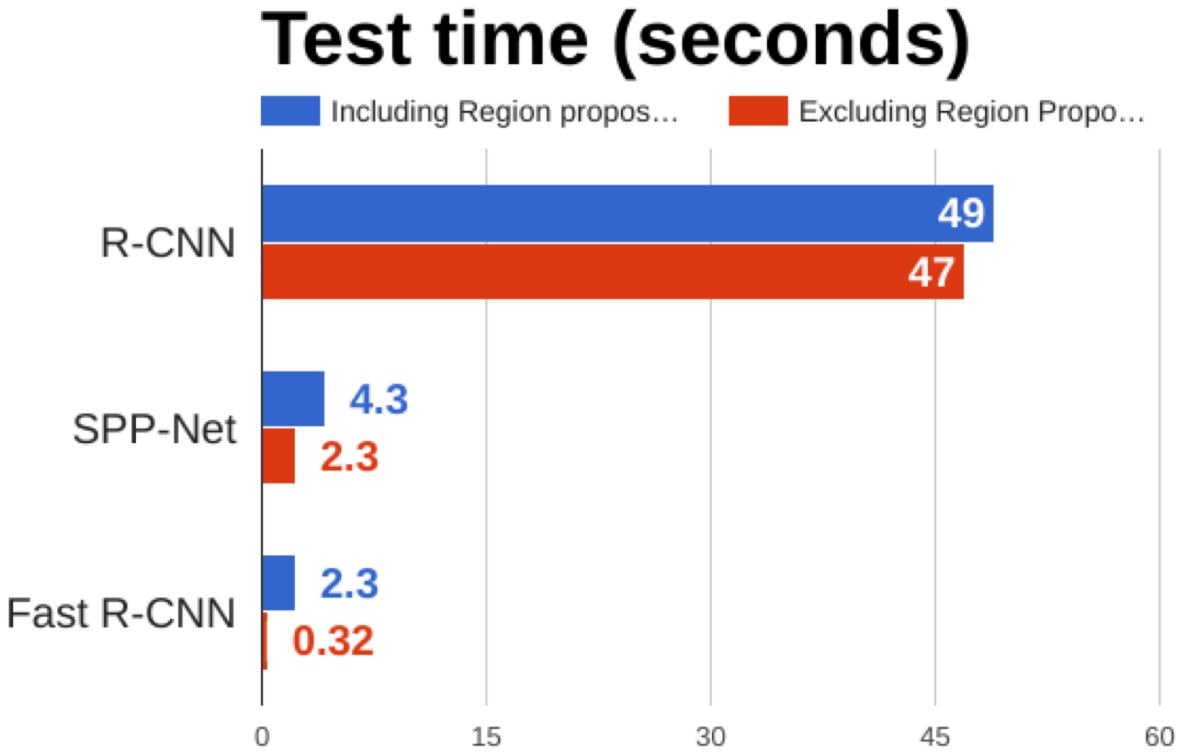


"Slow" R-CNN: Apply differentiable cropping to shared image features



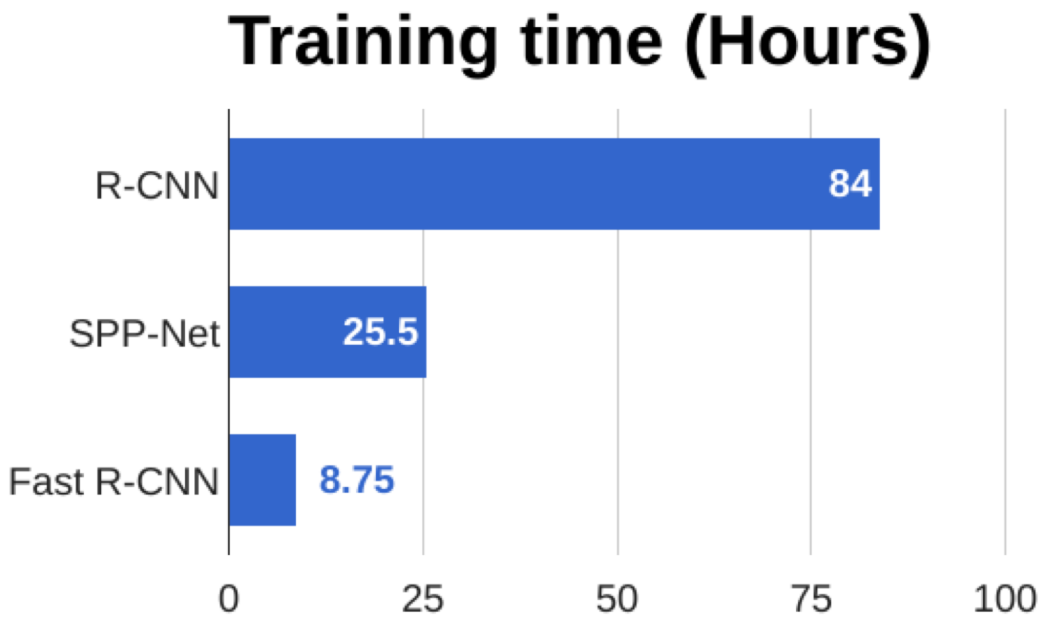
Fast R-CNN vs “Slow” R-CNN

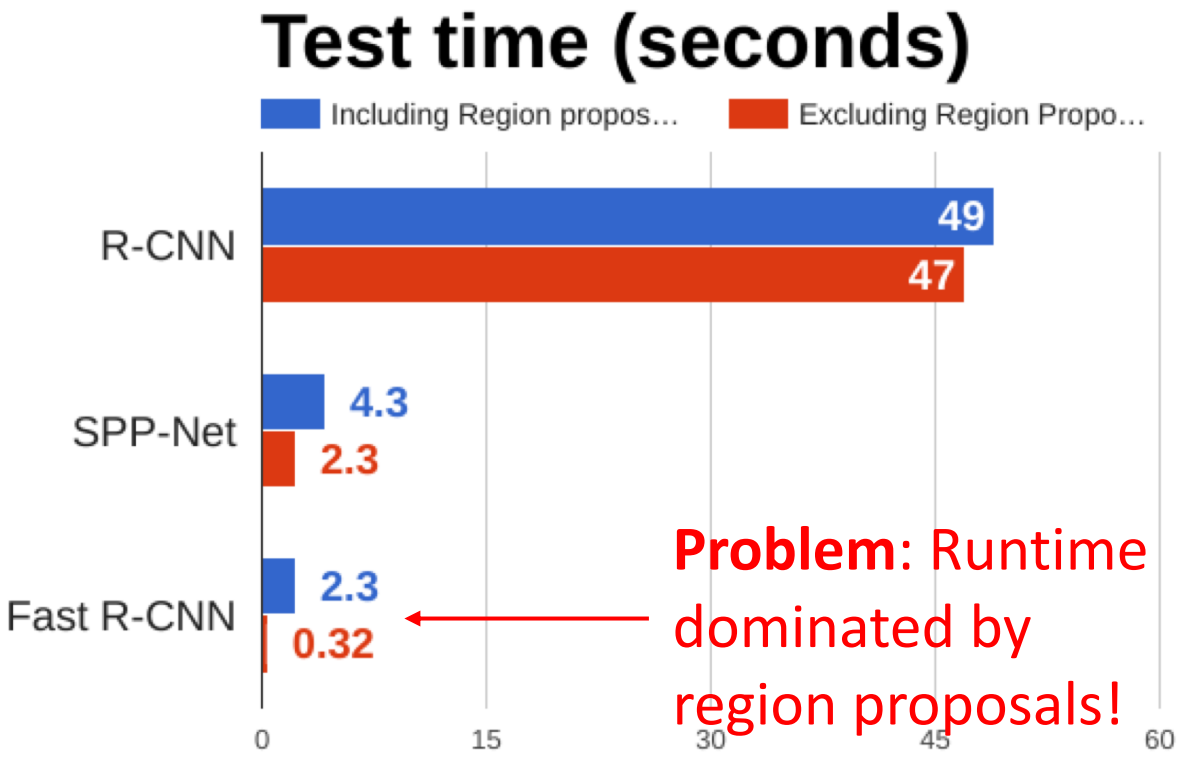




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

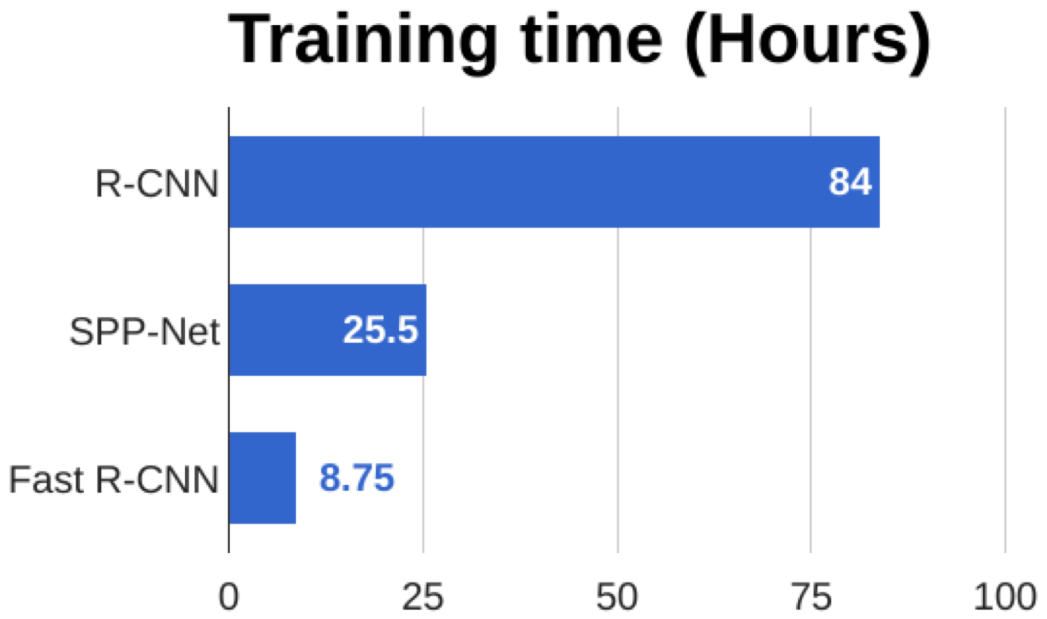
Fast R-CNN vs “Slow” R-CNN

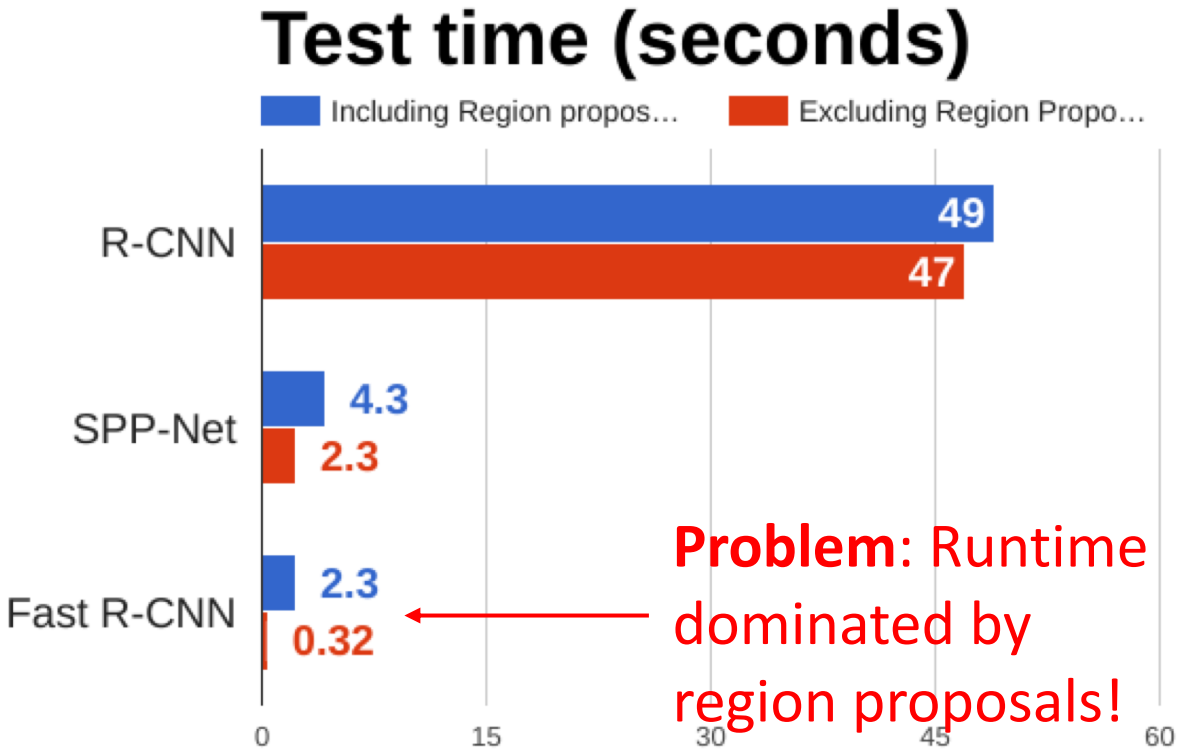




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs “Slow” R-CNN





Problem: Runtime dominated by region proposals!

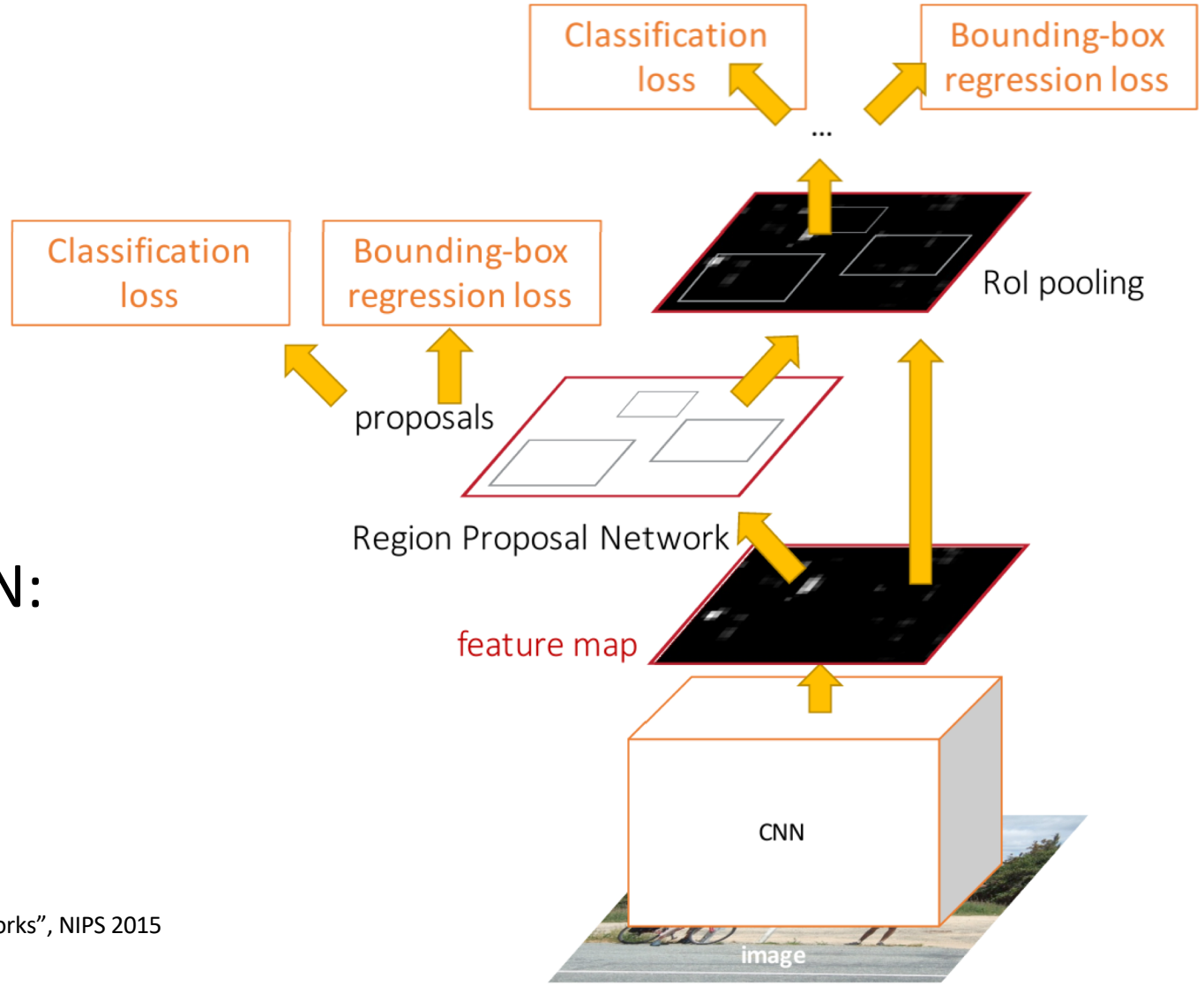
Recall: Region proposals computed by heuristic ”Selective Search” algorithm on CPU -- let’s learn them with a CNN instead!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Faster R-CNN: Learnable Region Proposals

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

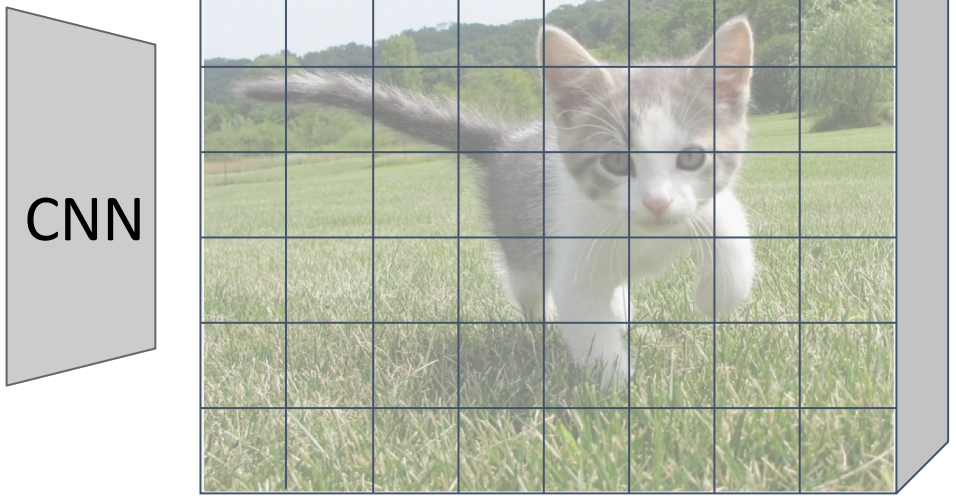


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network (RPN)

Imagine an **anchor box** of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

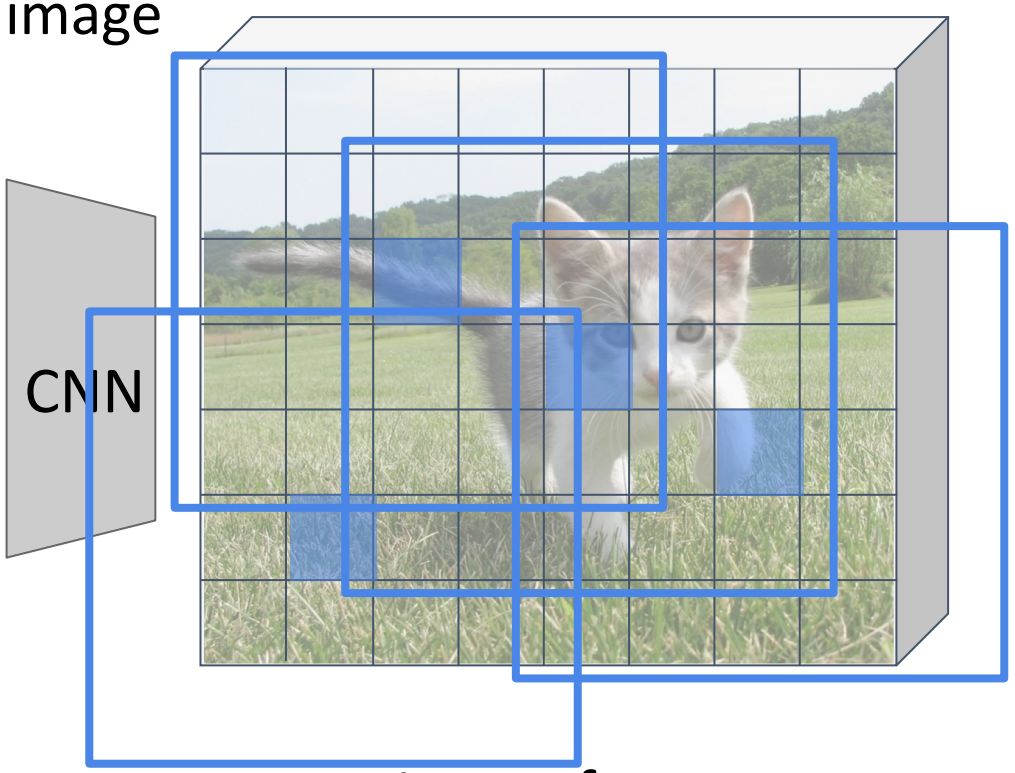


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network (RPN)

Imagine an anchor box of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

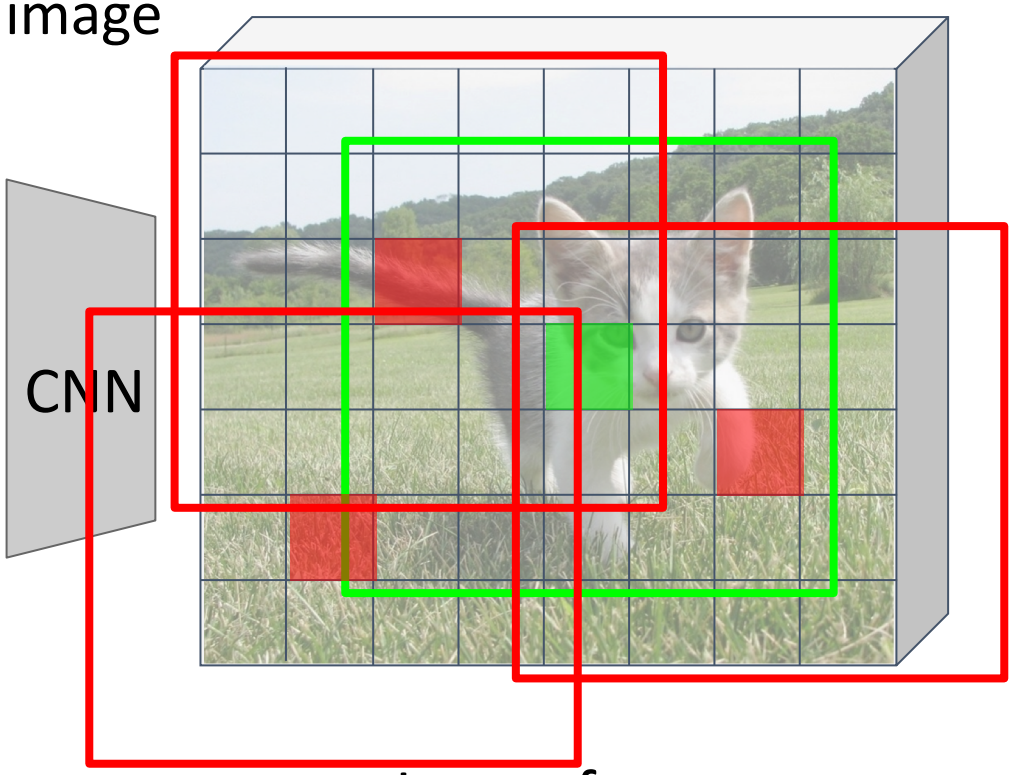
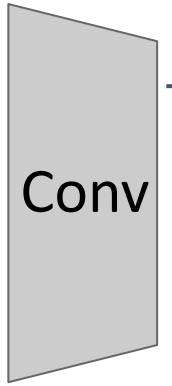


Image features
(e.g. 512 x 20 x 15)



Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (per-cell logistic regression, predict scores with conv layer)

Region Proposal Network (RPN)

Imagine an anchor box of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

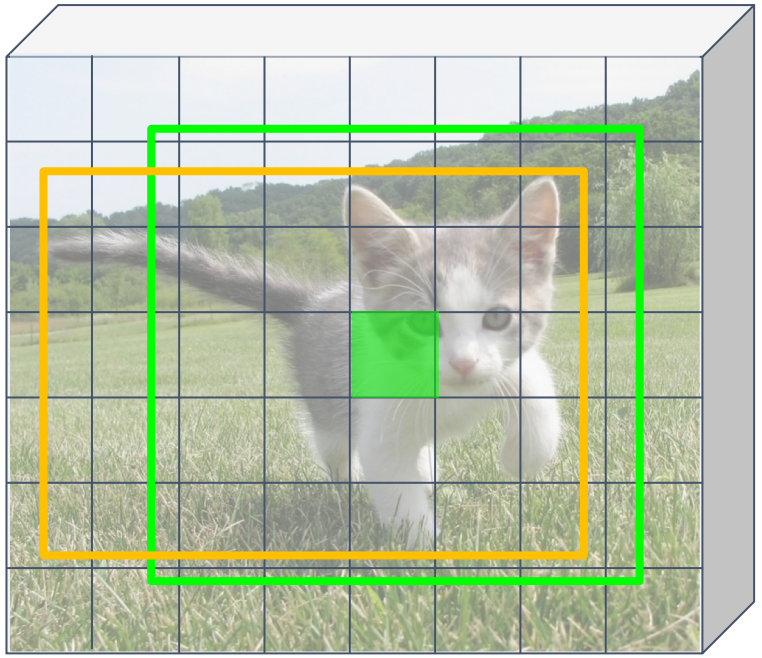


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

For positive boxes, also predict a box transform to regress from **anchor box** to **object box**

Region Proposal Network (RPN)

Problem: Anchor box may have the wrong size / shape

Solution: Use **K** different anchor boxes at each point!

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

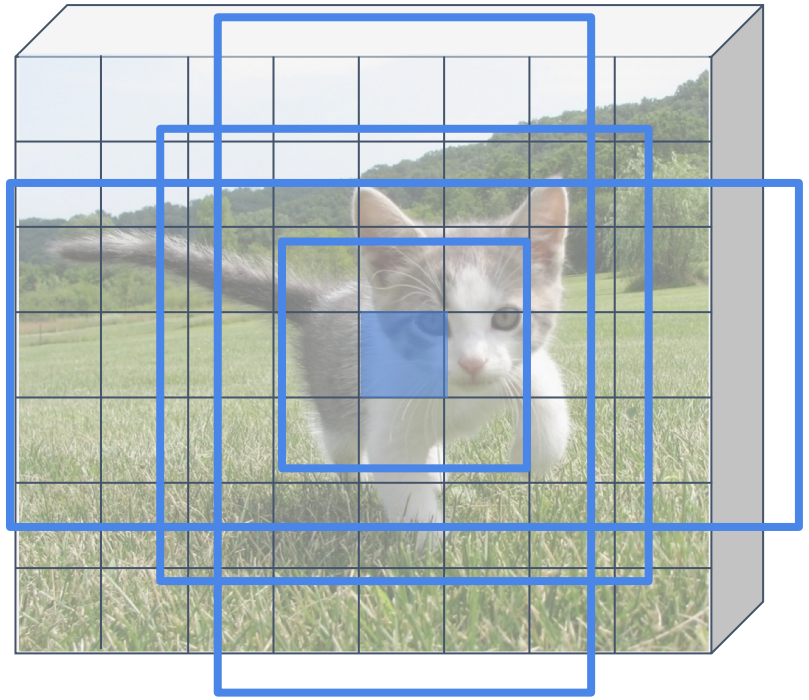


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
K x 20 x 15

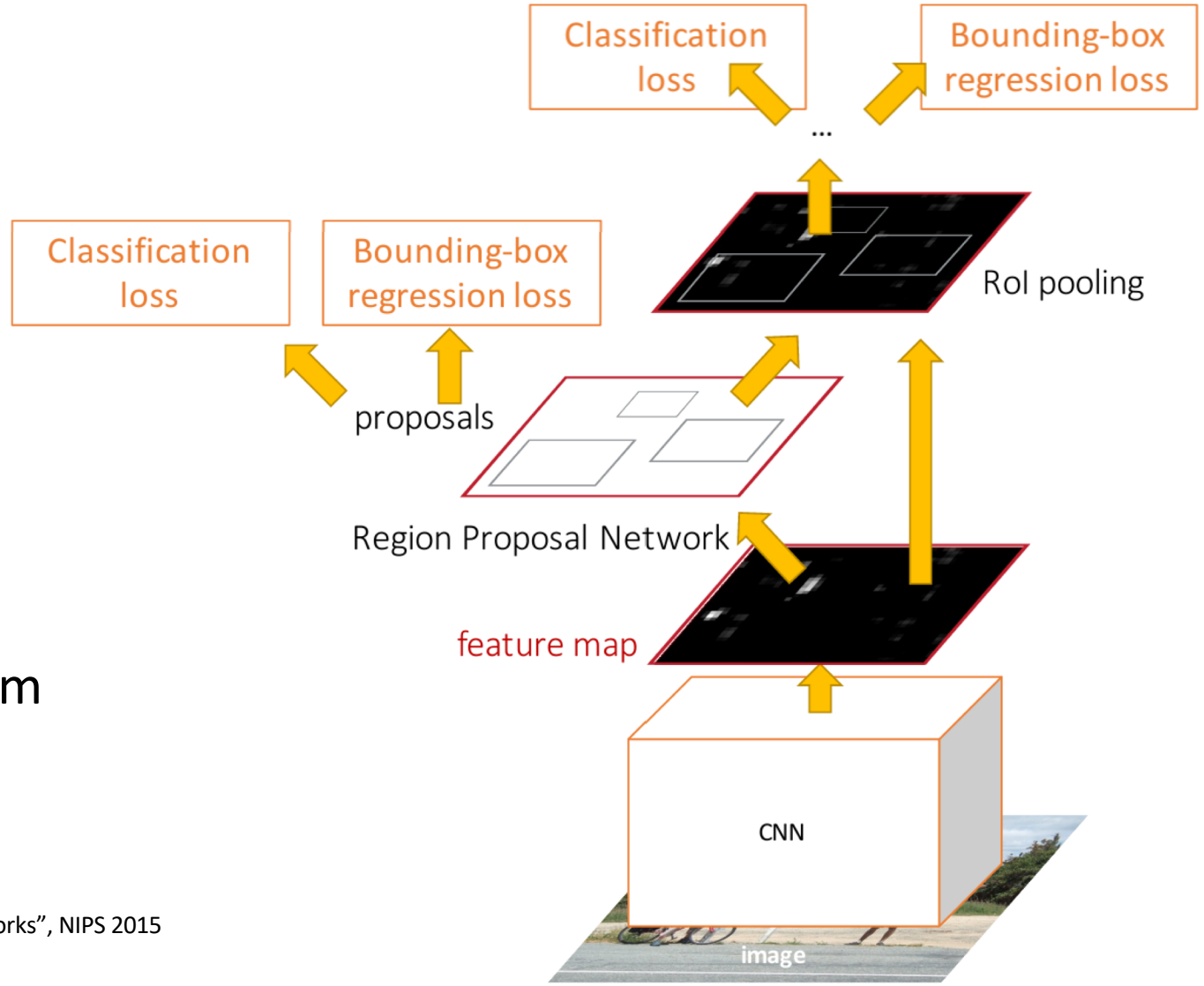
Box transforms
4K x 20 x 15

At test time: sort all K*20*15 boxes by their score, and take the top ~300 as our region proposals

Faster R-CNN: Learnable Region Proposals

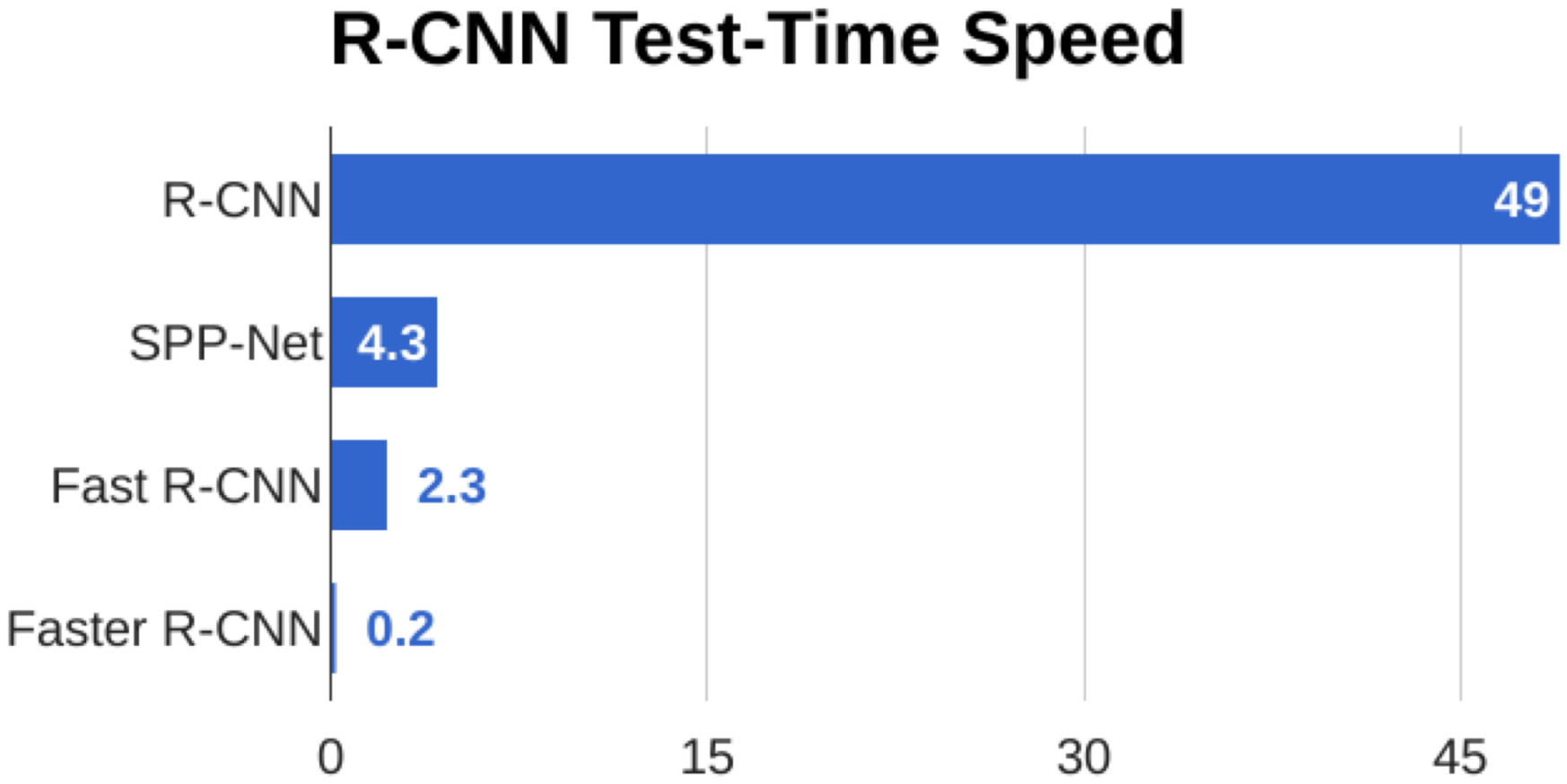
Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN: Learnable Region Proposals

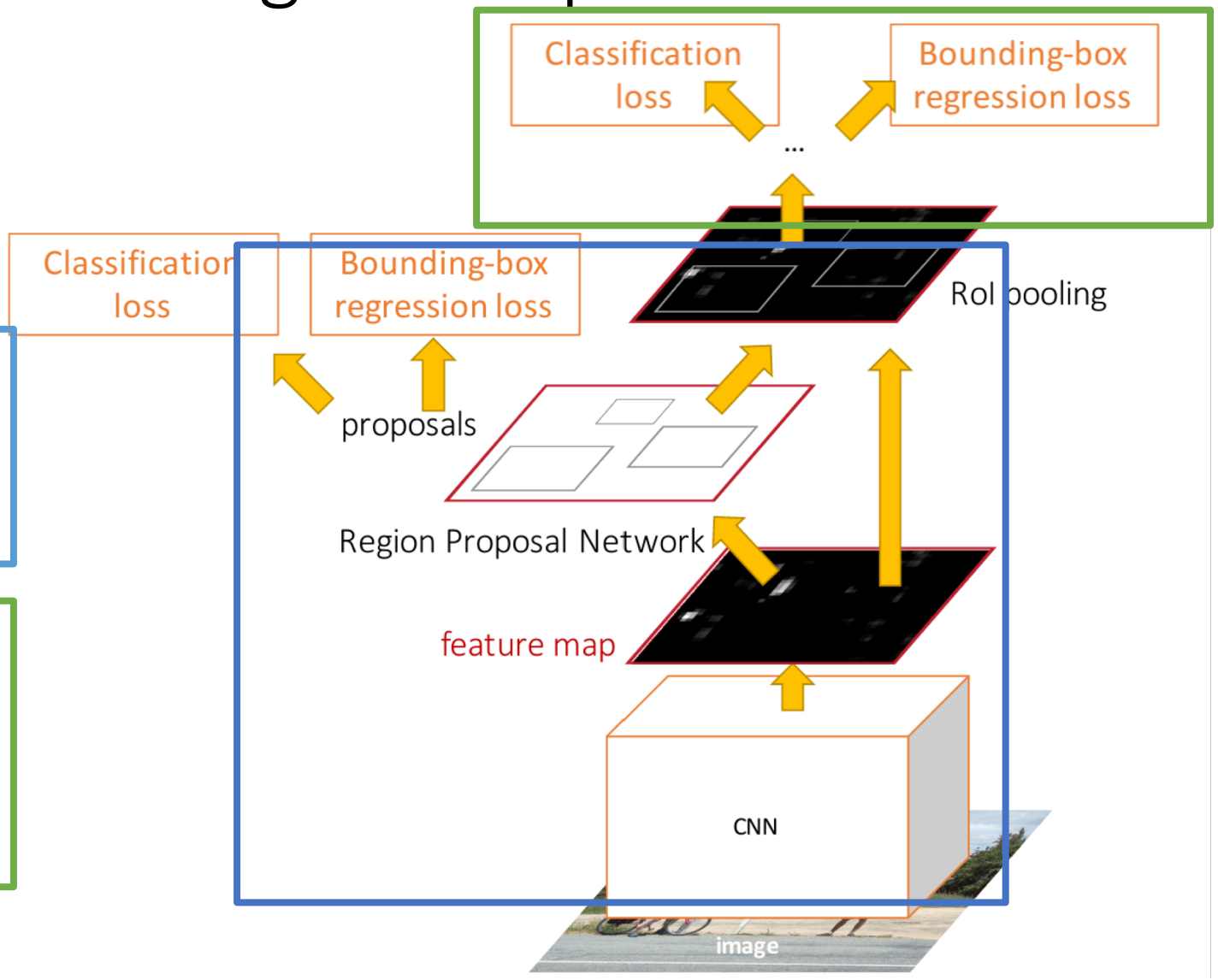


Faster R-CNN: Learnable Region Proposals

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



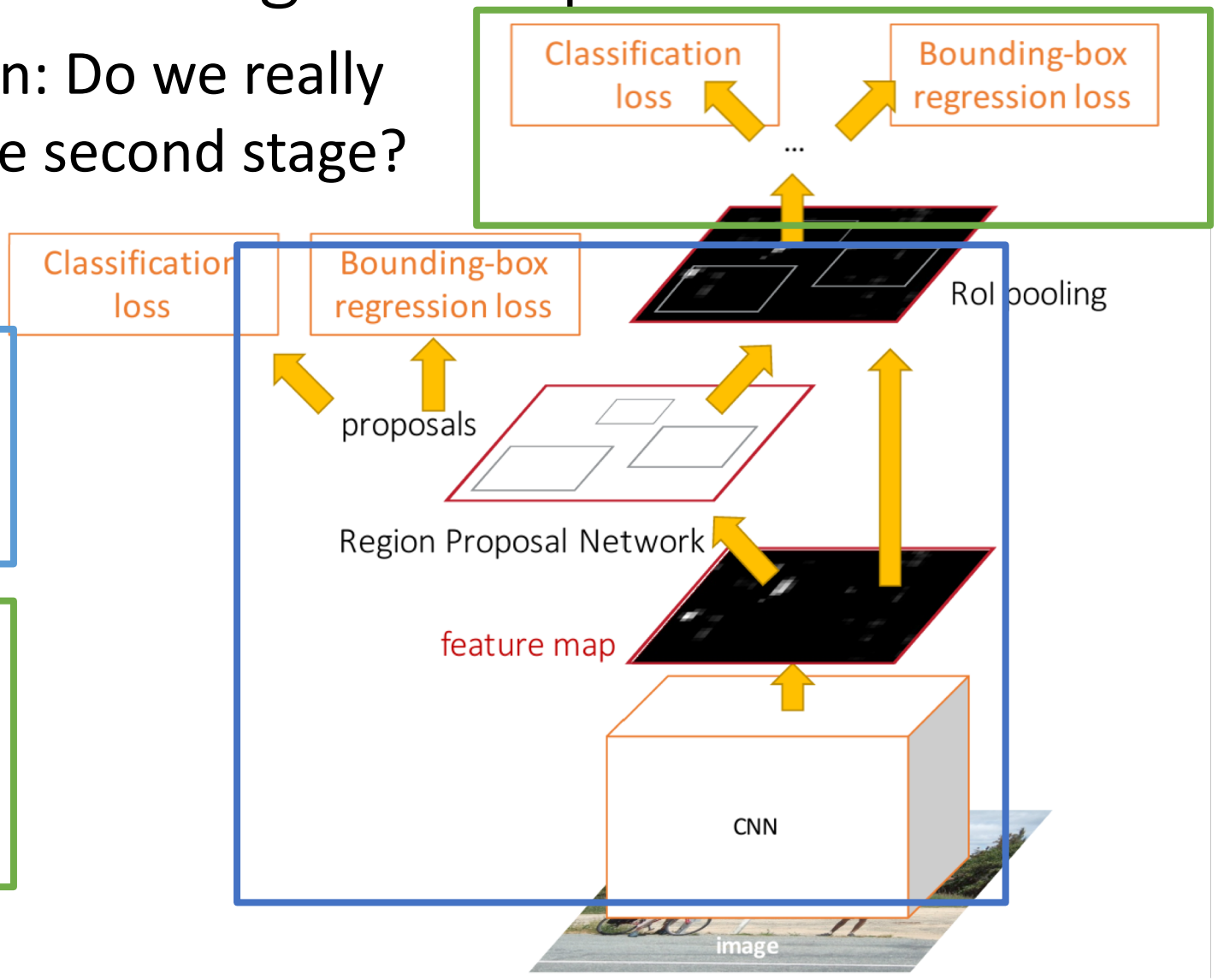
Faster R-CNN: Learnable Region Proposals

Question: Do we really need the second stage?

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Single-Stage Object Detection

RPN: Classify each anchor as object / not object

Single-Stage Detector: Classify each object as one of C categories (or background)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

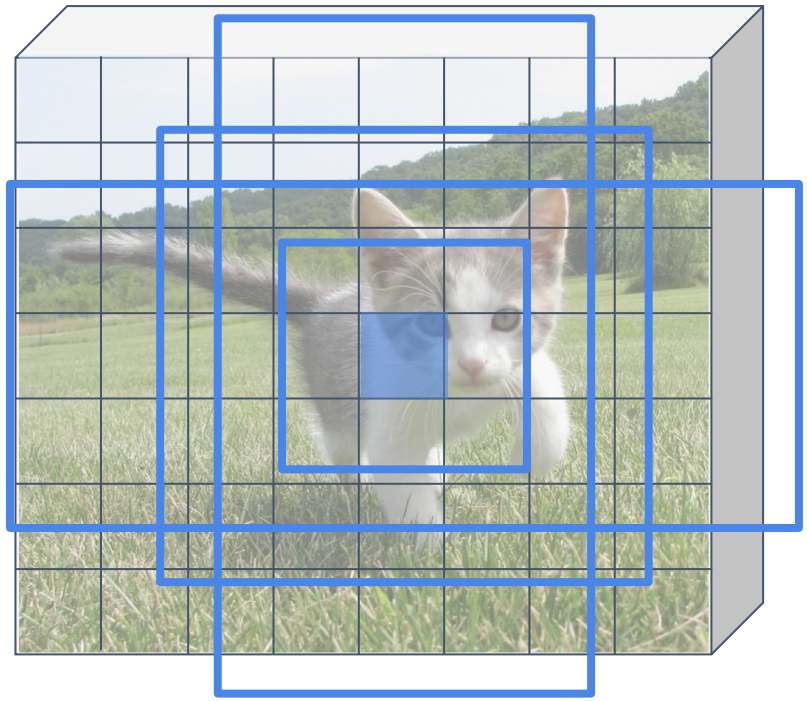


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor category
(C+1) x K x 20 x 15

Box transforms
4K x 20 x 15

Remember: K anchors at each position in image feature map

Single-Stage Object Detection

RPN: Classify each anchor as object / not object

Single-Stage Detector: Classify each object as one of C categories (or background)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

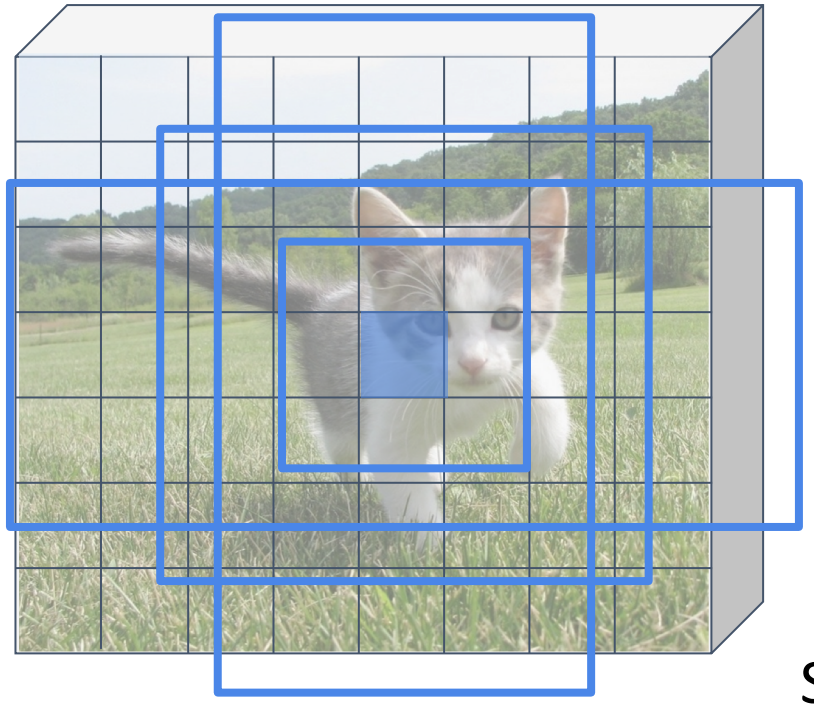


Image features
(e.g. 512 x 20 x 15)

Conv

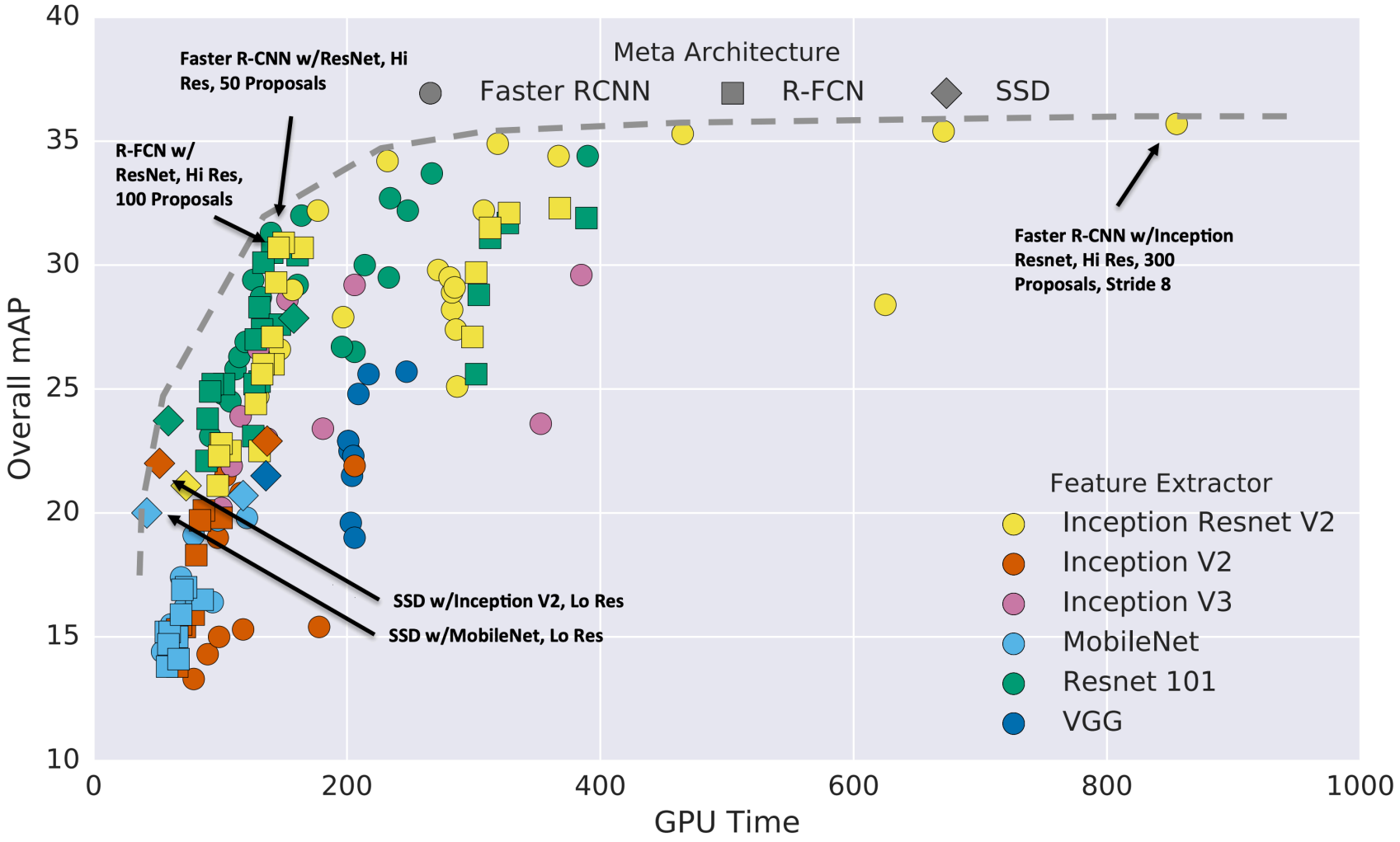
Anchor category
(C+1) x K x 20 x 15

Box transforms
C x 4K x 20 x 15

Sometimes use **category-specific regression**: Predict different box transforms for each category

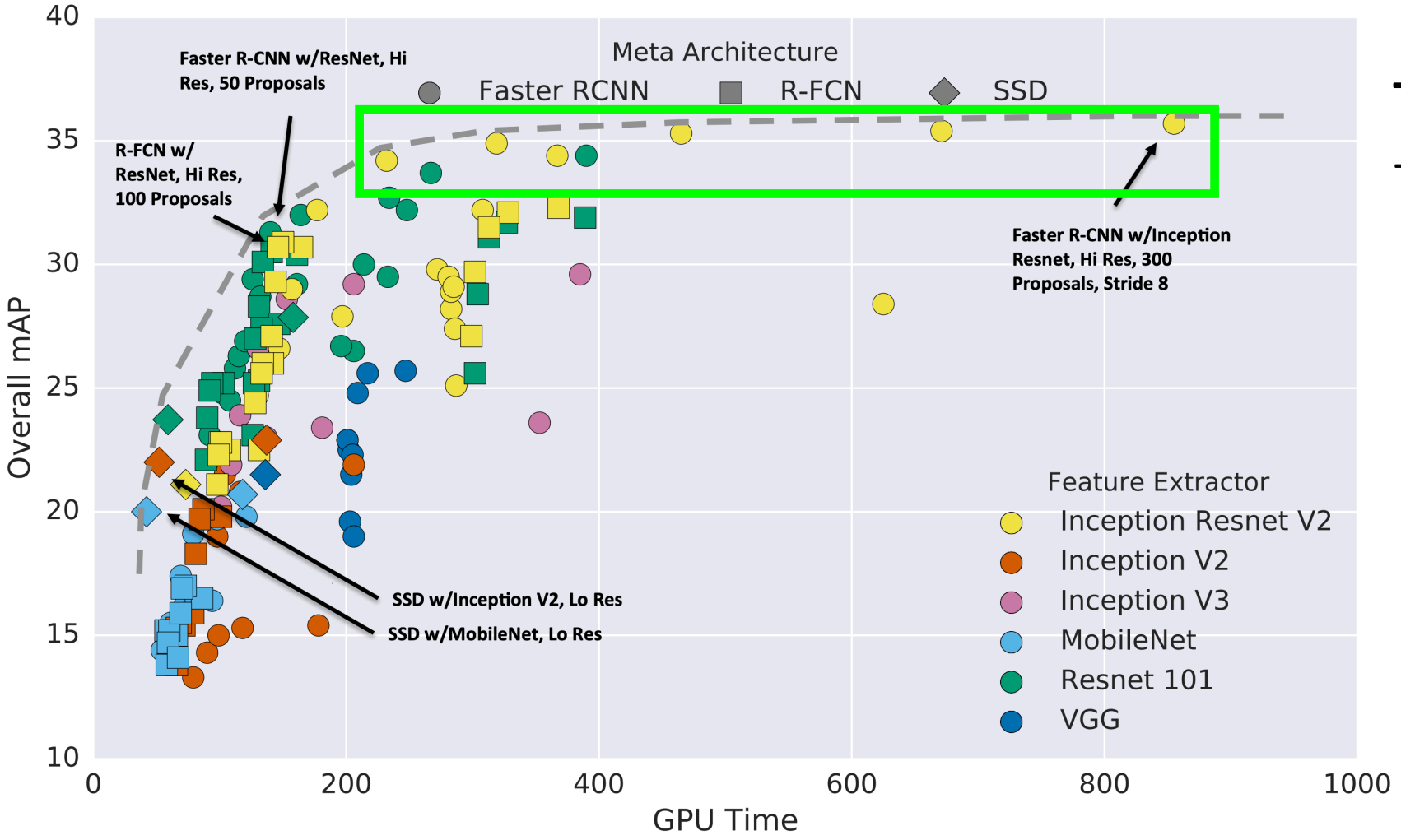
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Object Detection: Lots of variables!



Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object Detection: Lots of variables!

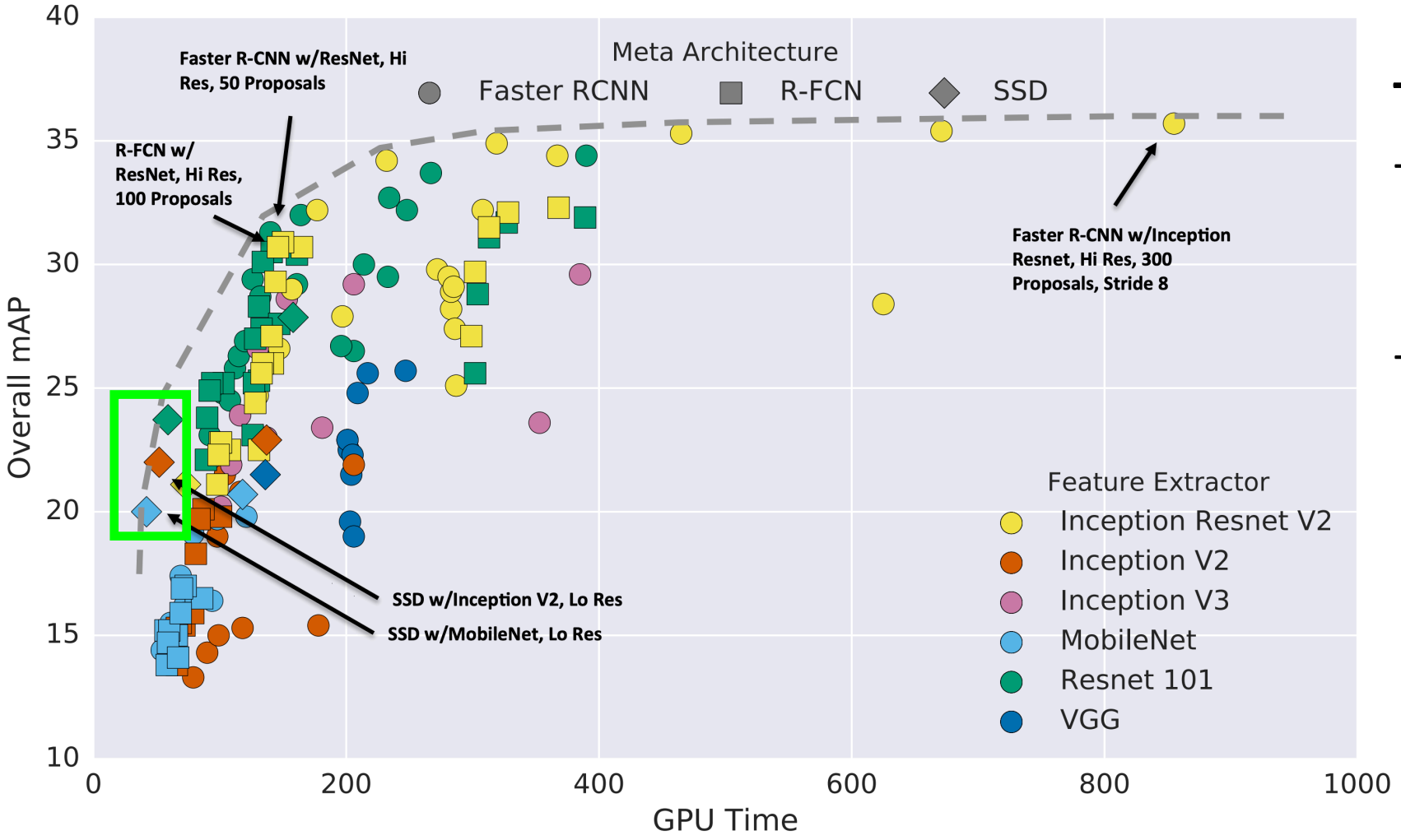


Takeaways:

- Two stage method (Faster R-CNN) get the best accuracy, but are slower

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object Detection: Lots of variables!

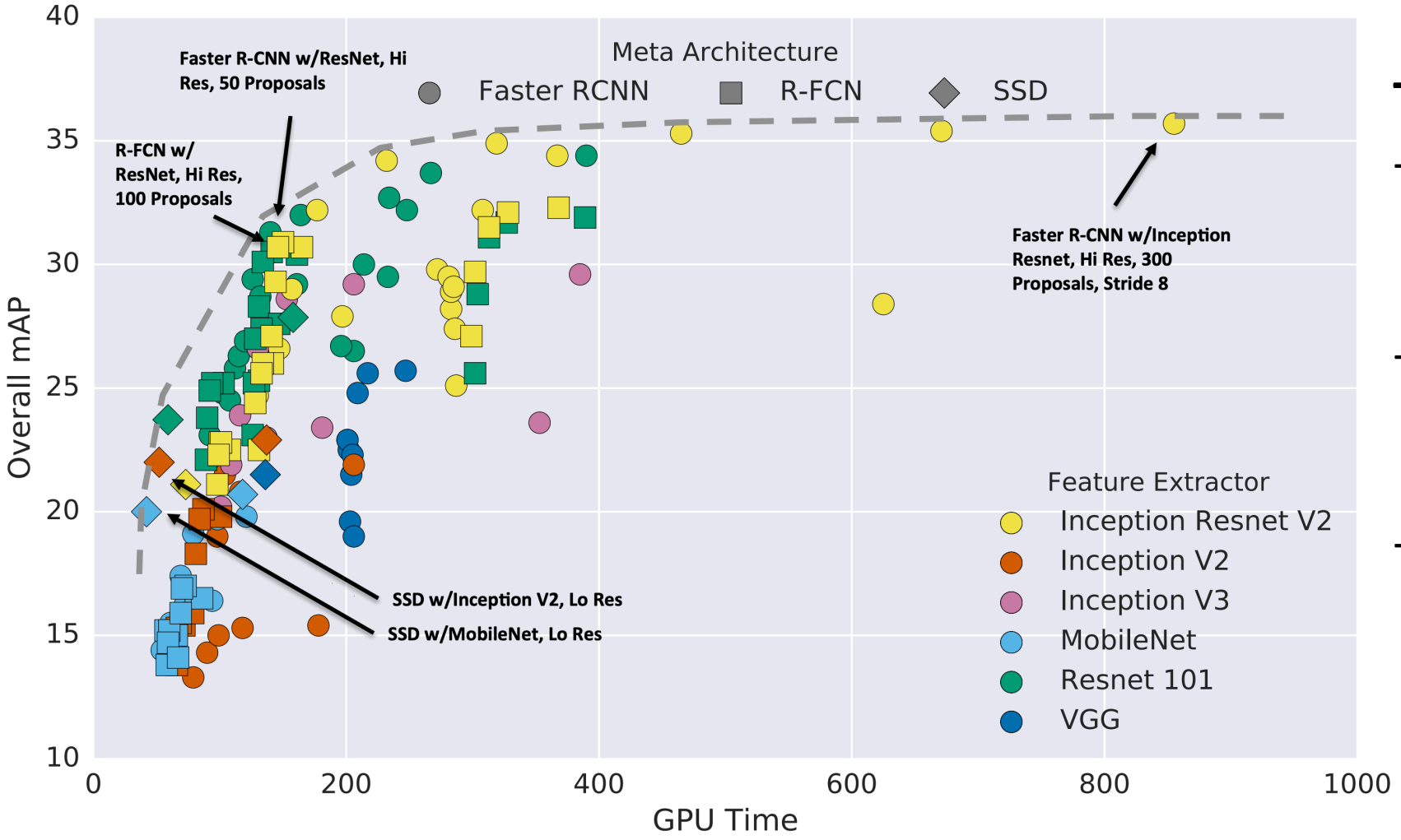


Takeaways:

- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object Detection: Lots of variables!

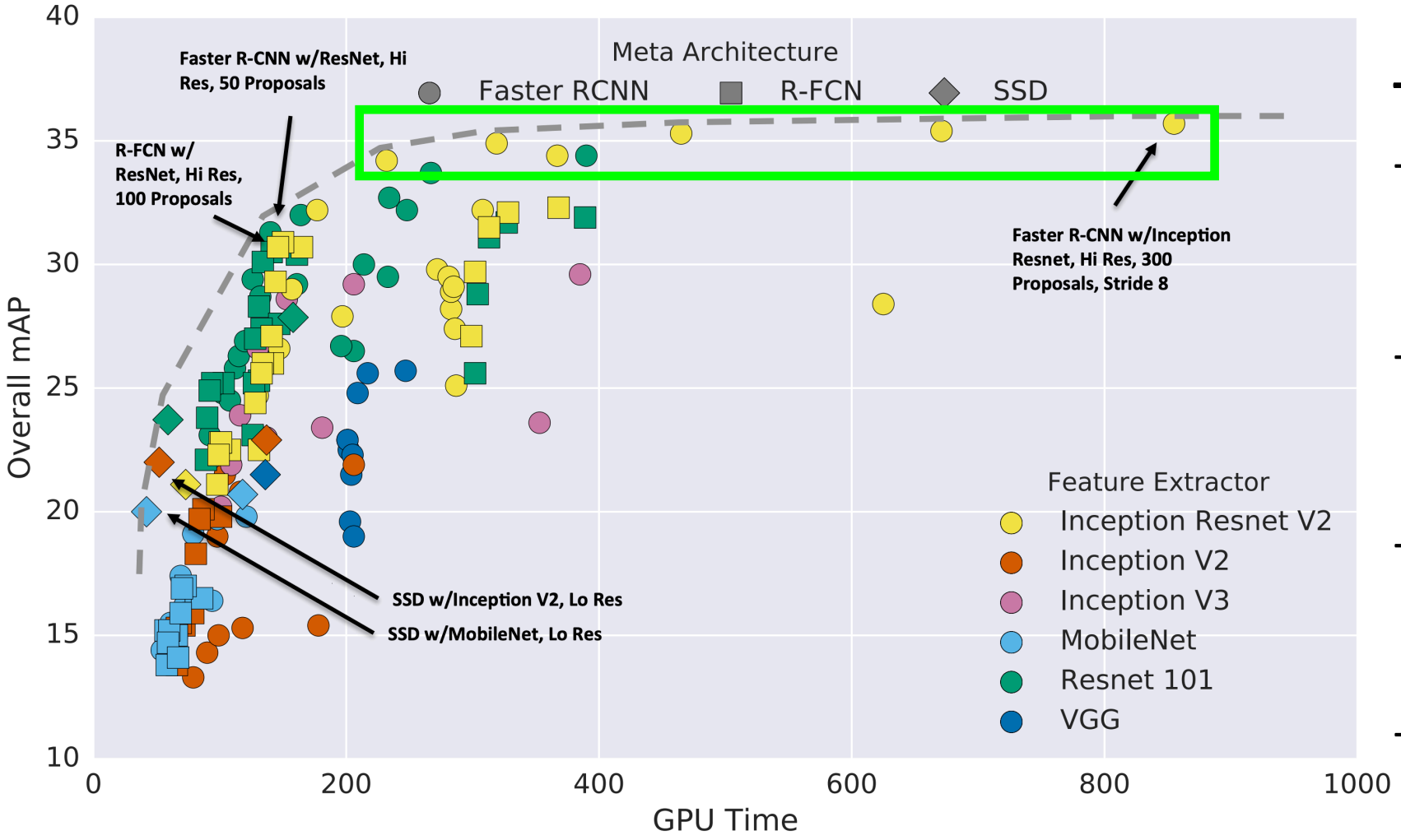


Takeaways:

- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well
- Bigger backbones improve performance, but are slower

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object Detection: Lots of variables!

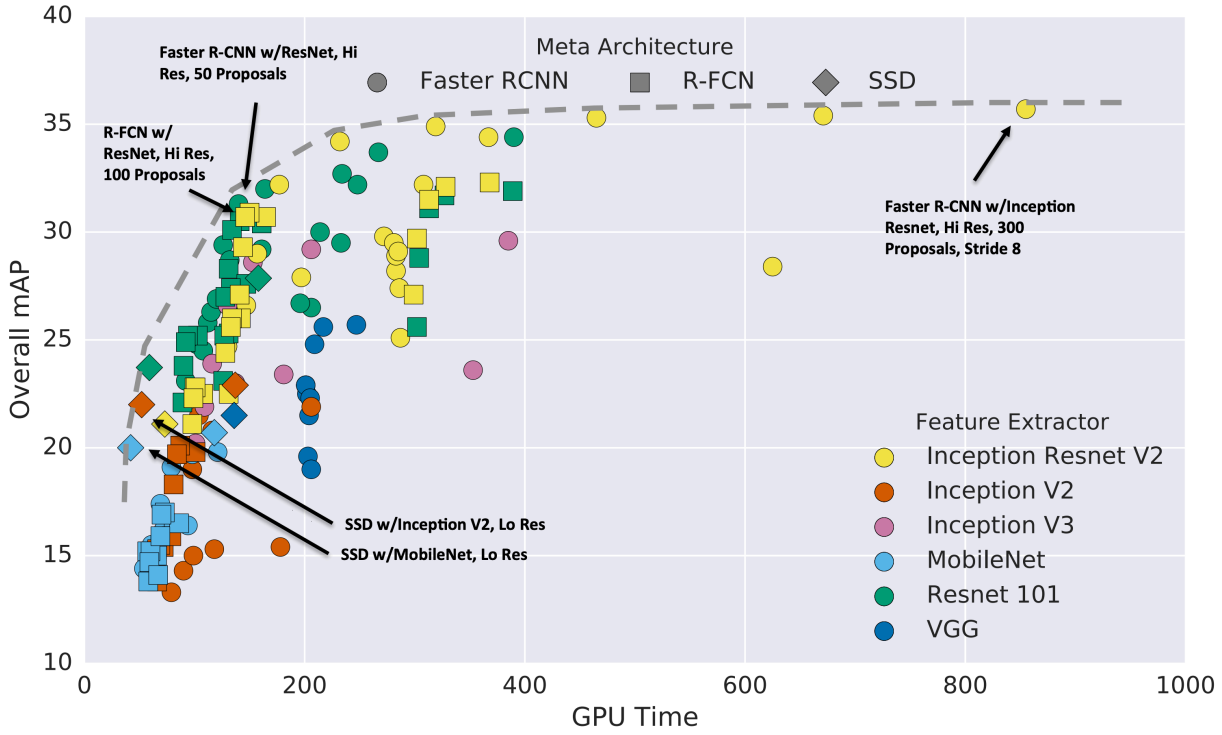


Takeaways:

- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well
- Bigger backbones improve performance, but are slower
- Diminishing returns for slower methods

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object Detection: Lots of variables!



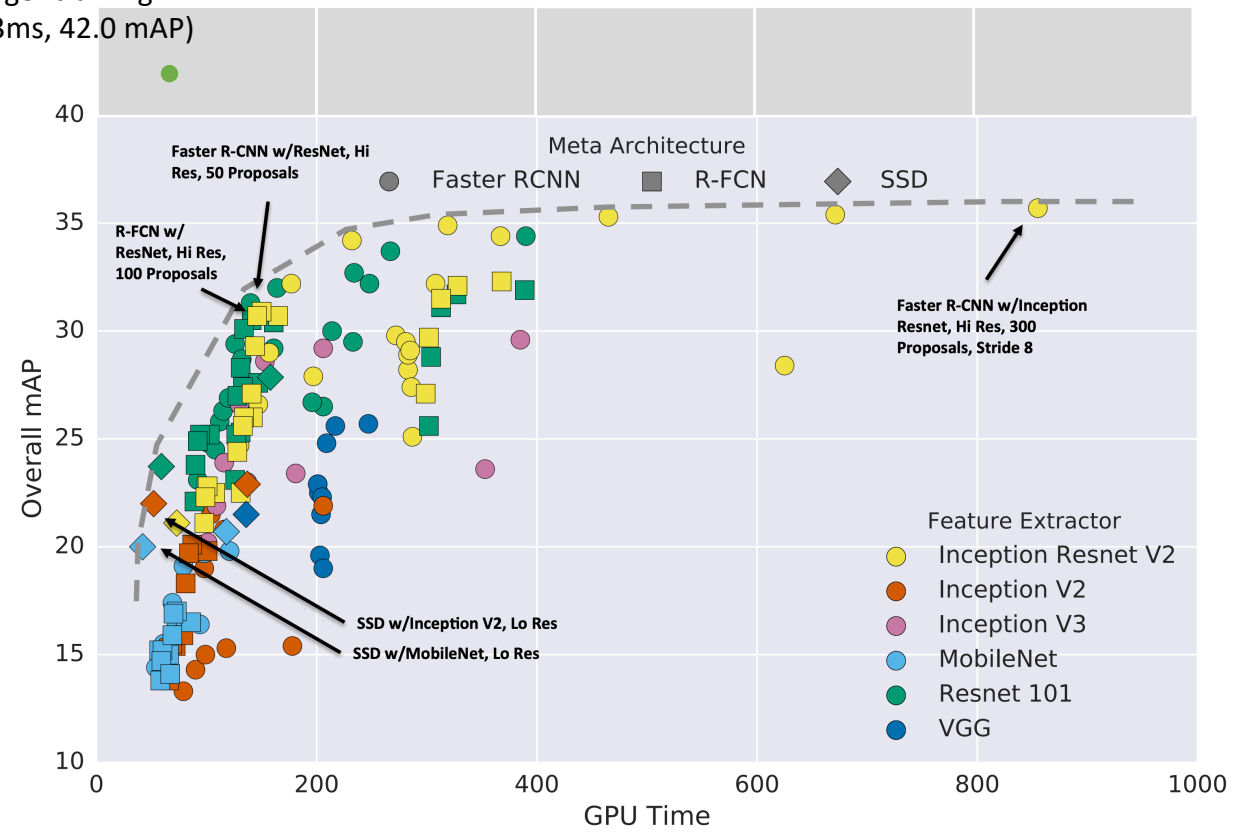
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!

Faster R-CNN w/ResNet-101-FPN, longer training (63ms, 42.0 mAP)



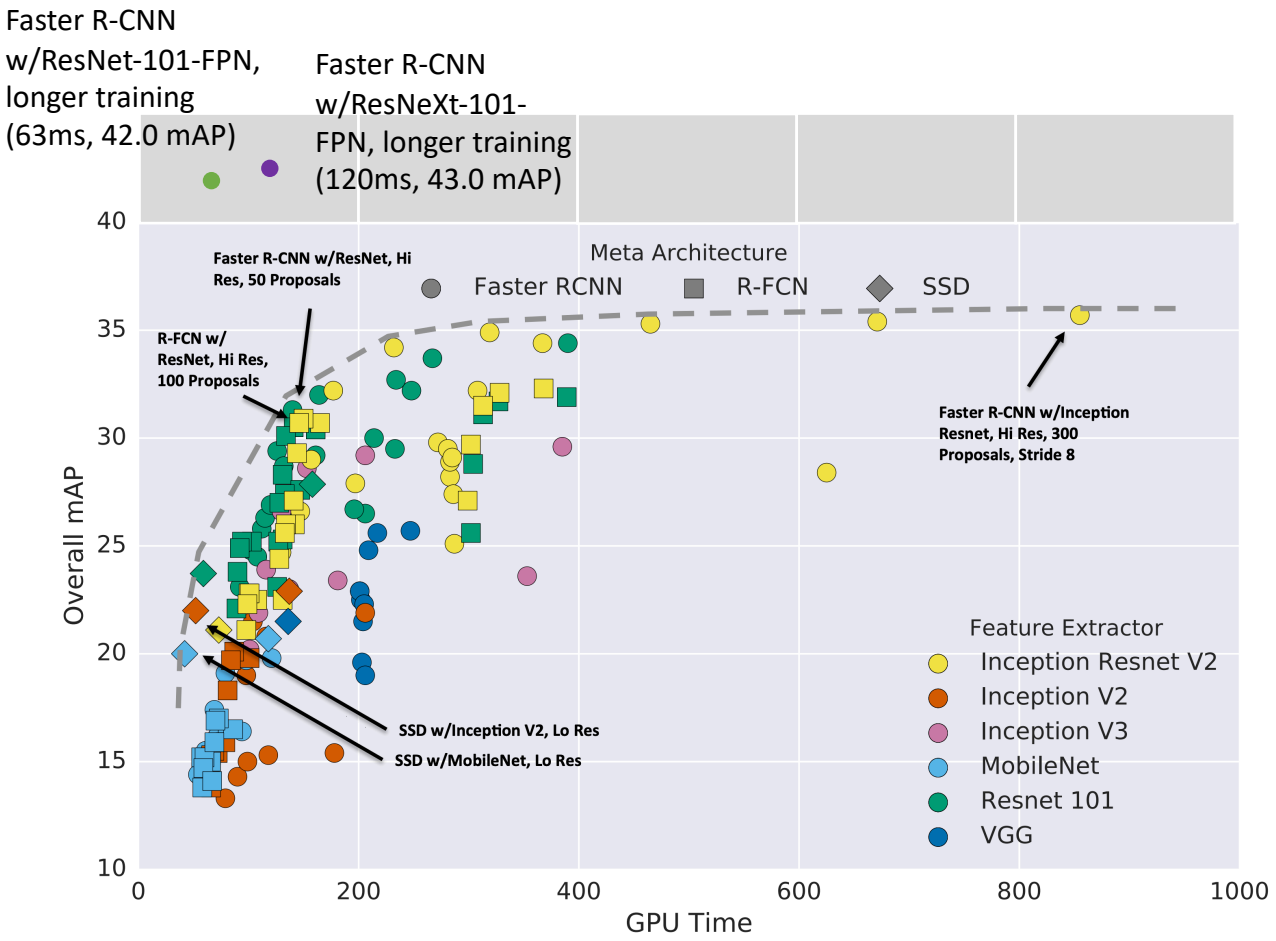
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



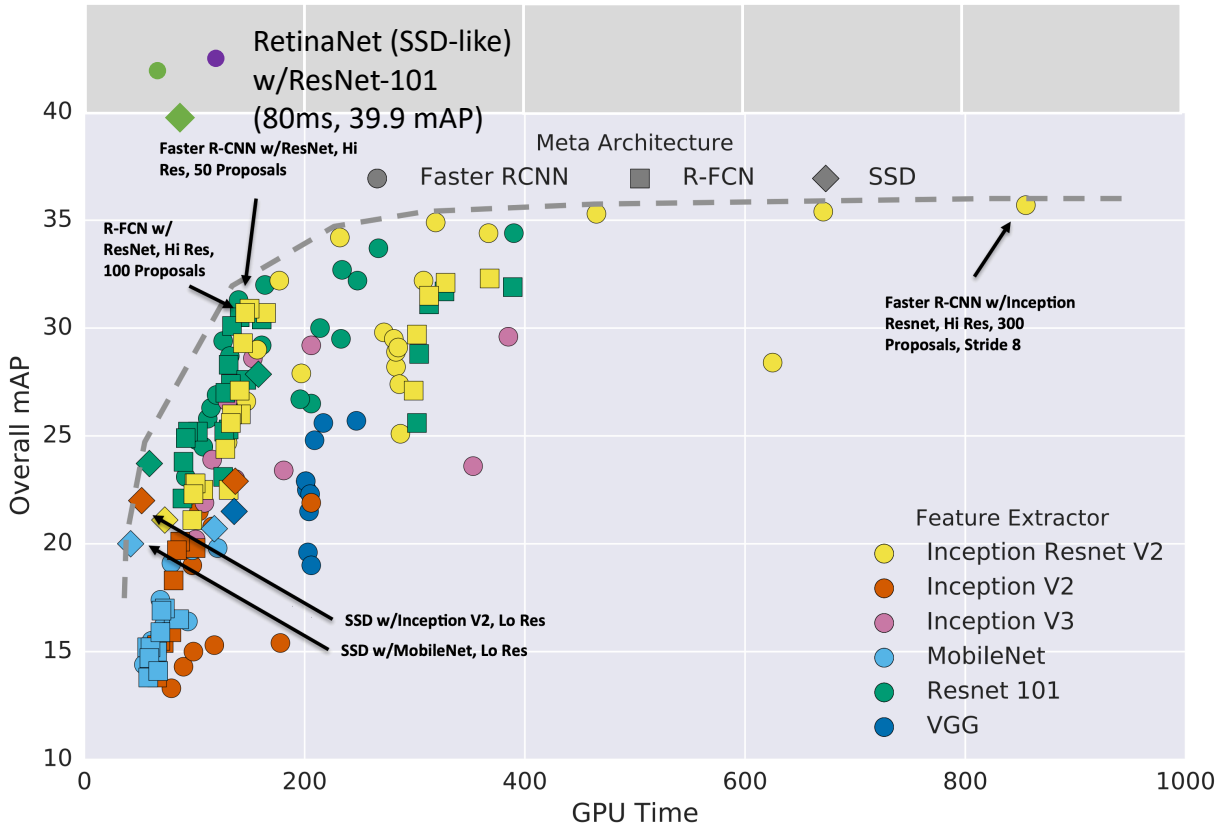
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



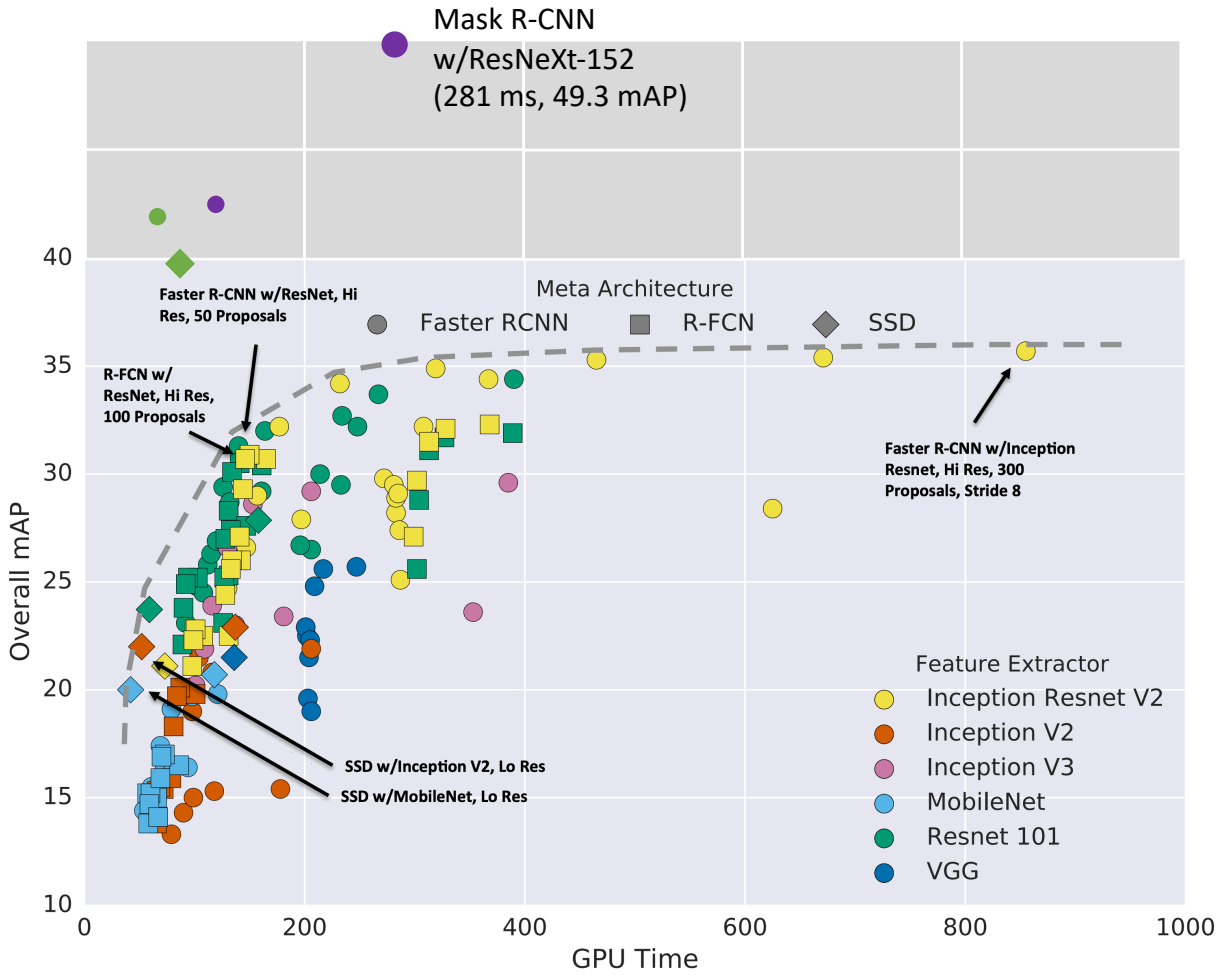
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt
- Single-Stage methods have improved

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt
- Single-Stage methods have improved
- Very big models work better

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



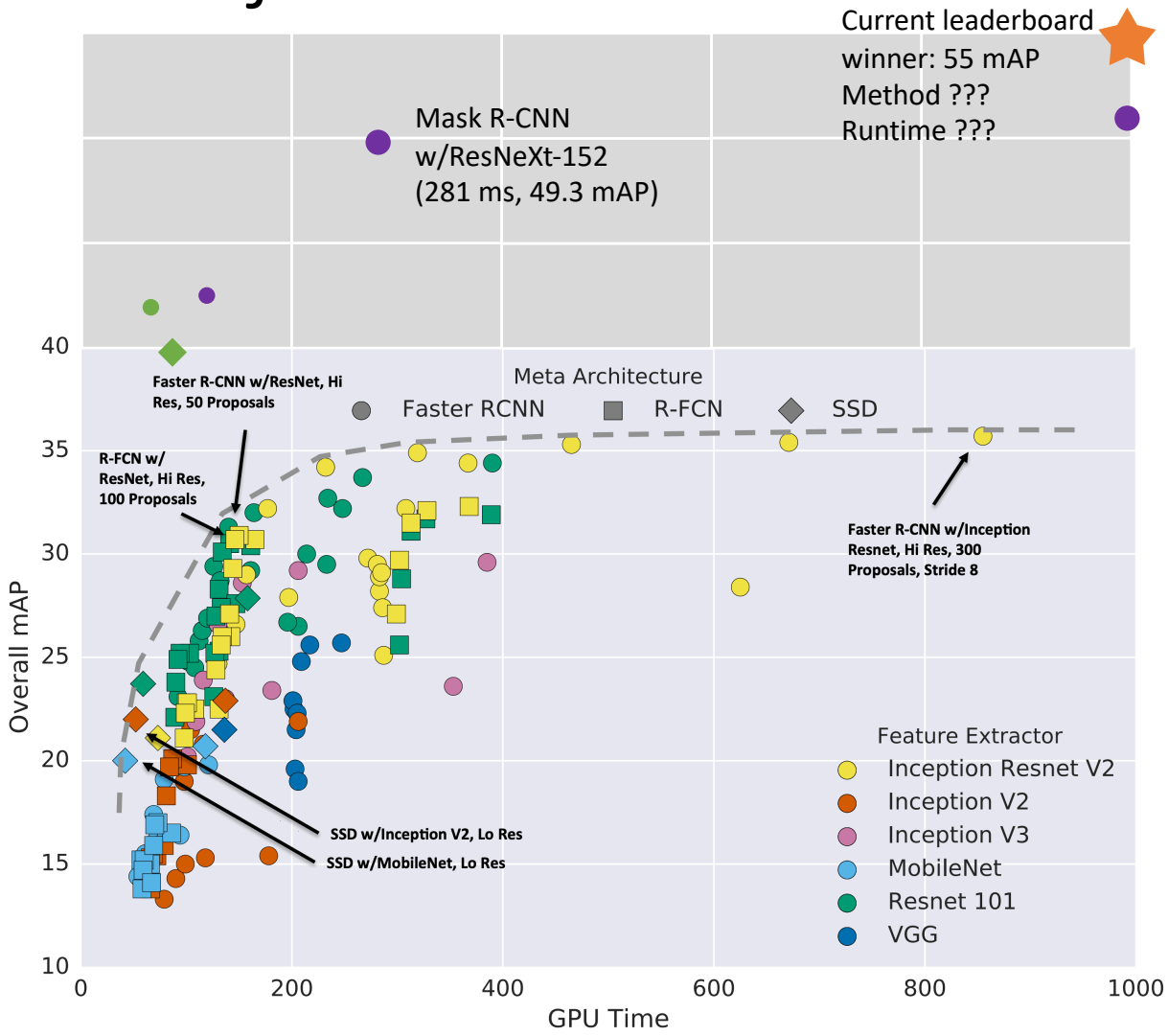
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt
- Single-Stage methods have improved
- Very big models work better
- Test-time augmentation pushes numbers up

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt
- Single-Stage methods have improved
- Very big models work better
- Test-time augmentation pushes numbers up
- Big ensembles, more data, etc

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

https://competitions.codalab.org/competitions/20794#results

Object Detection: Open-Source Code

Object detection is hard! Don't implement it yourself

TensorFlow Detection API:

https://github.com/tensorflow/models/tree/master/research/object_detection

Faster R-CNN, SSD, RFCN, Mask R-CNN

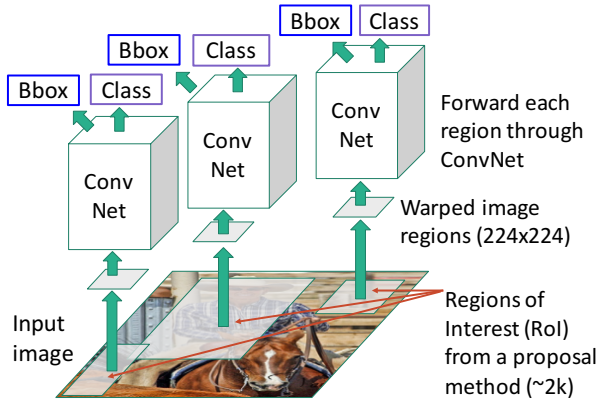
Detectron2 (PyTorch):

https://github.com/facebookresearch/detectron2

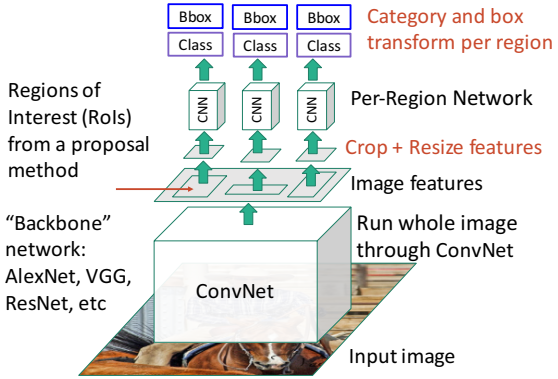
Fast / Faster / Mask R-CNN, RetinaNet

Summary

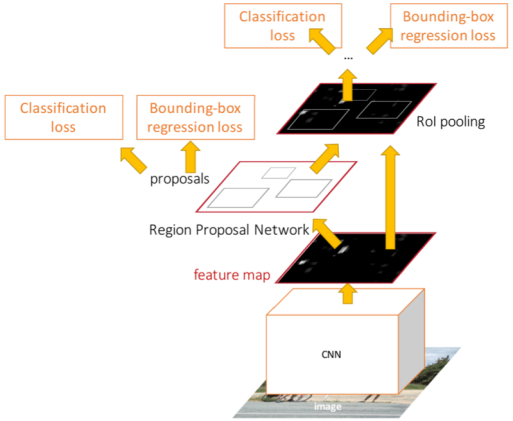
"Slow" R-CNN: Run CNN independently for each region



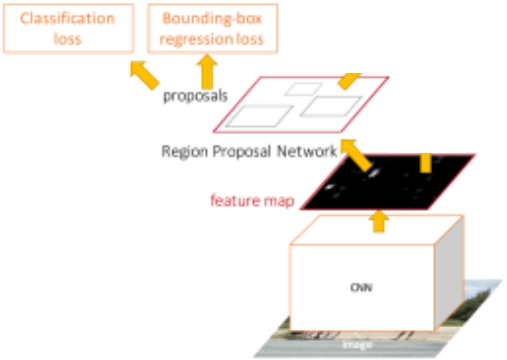
Fast R-CNN: Apply differentiable cropping to shared image features



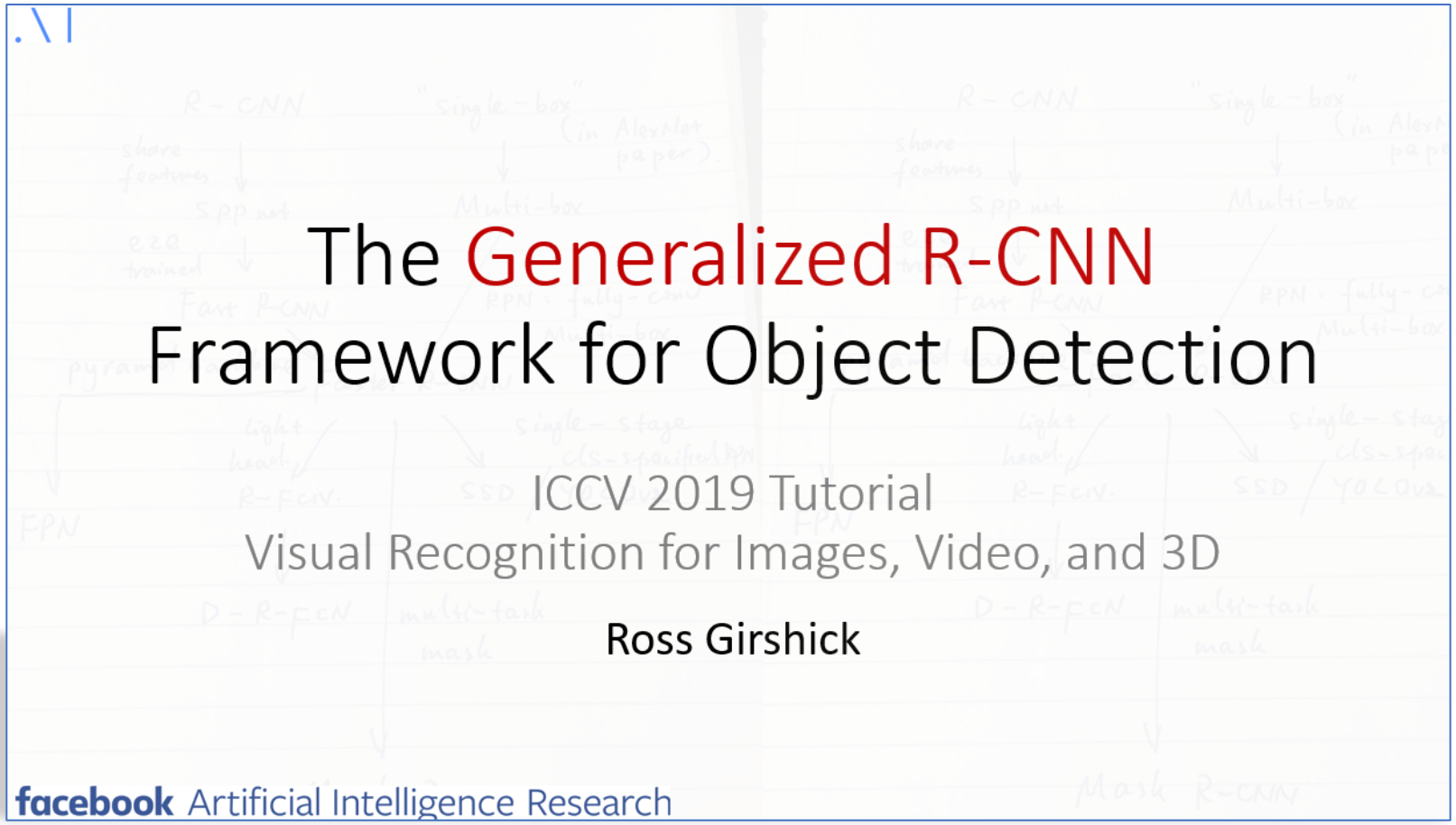
Faster R-CNN: Compute proposals with CNN



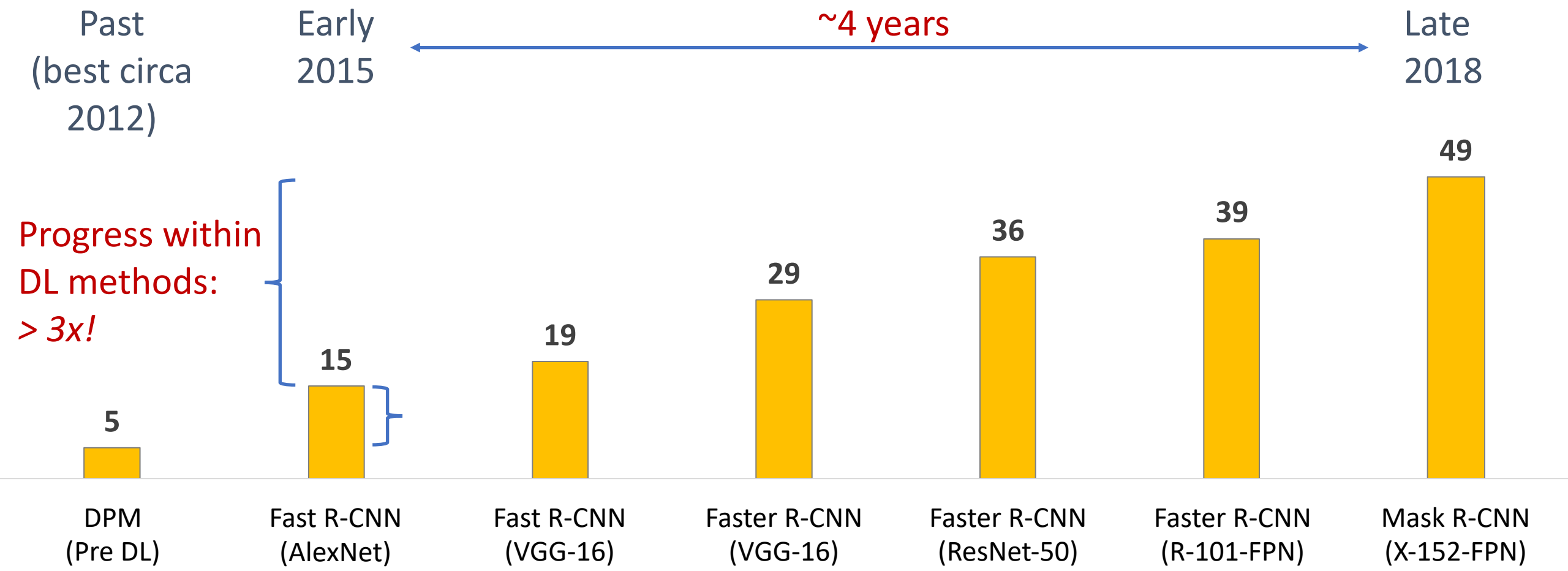
Single-Stage: Fully convolutional detector



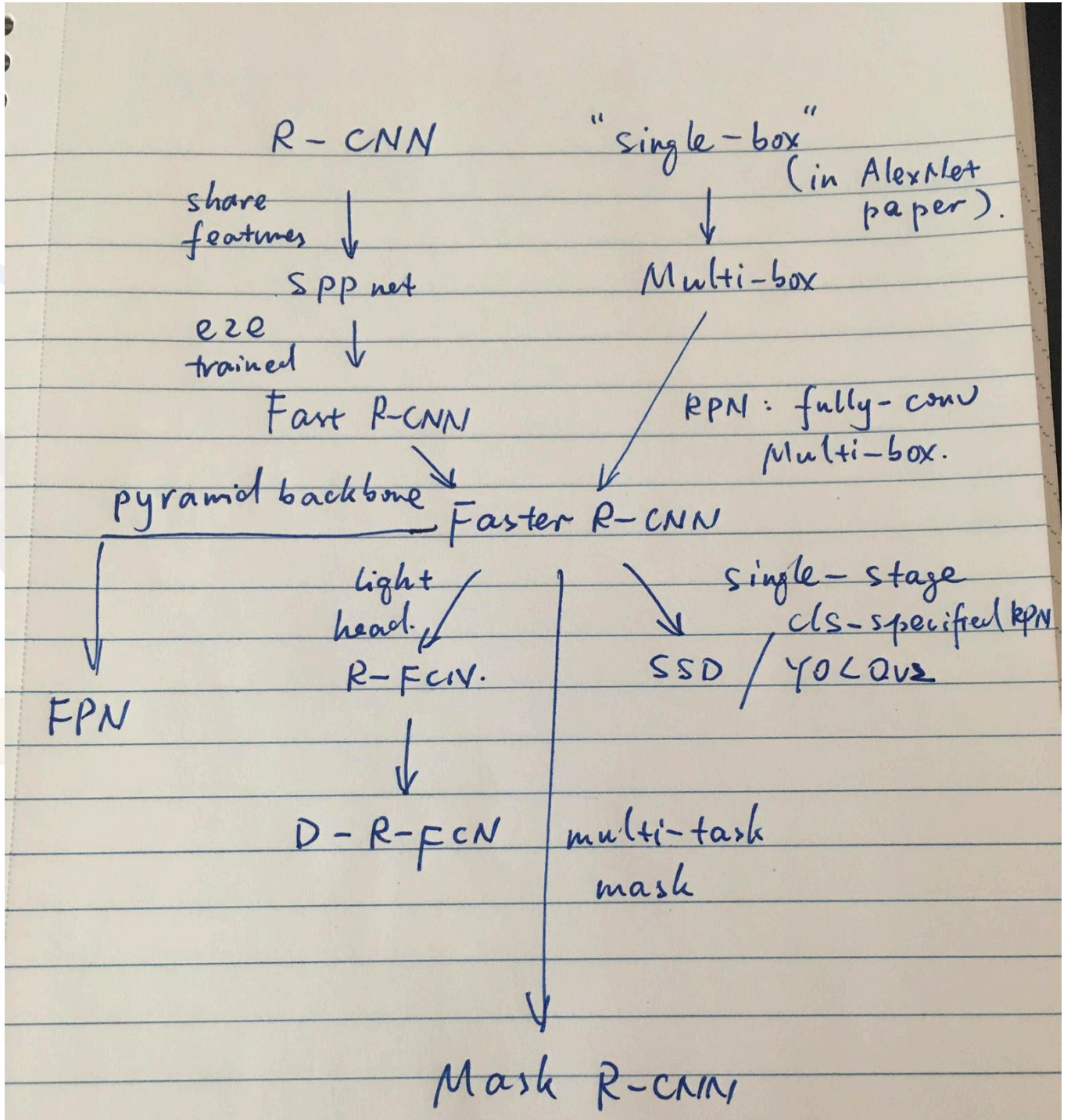
A few more slides from:



COCO Object Detection Average Precision (%)

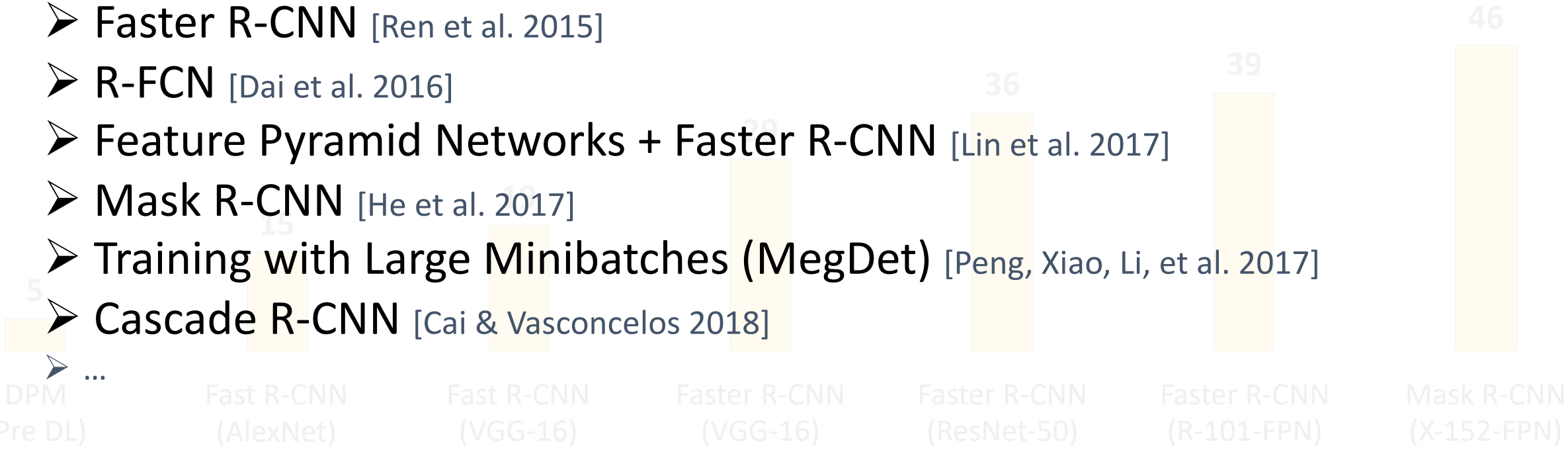


Modern object detection is a complex web of related methods



Steady Progress on Boxes and Masks

- R-CNN [Girshick et al. 2014]
- SPP-net [He et al. 2014]
- Fast R-CNN [Girshick. 2015]
- Faster R-CNN [Ren et al. 2015]
- R-FCN [Dai et al. 2016]
- Feature Pyramid Networks + Faster R-CNN [Lin et al. 2017]
- Mask R-CNN [He et al. 2017]
- Training with Large Minibatches (MegDet) [Peng, Xiao, Li, et al. 2017]
- Cascade R-CNN [Cai & Vasconcelos 2018]
- …



Faster R-CNN with a Feature Pyramid Network

Per-image computation

Per-region computation for each $r_i \in r(I)$

f_I = FPN(I)

I:



RPN(f_I)



RoIPool

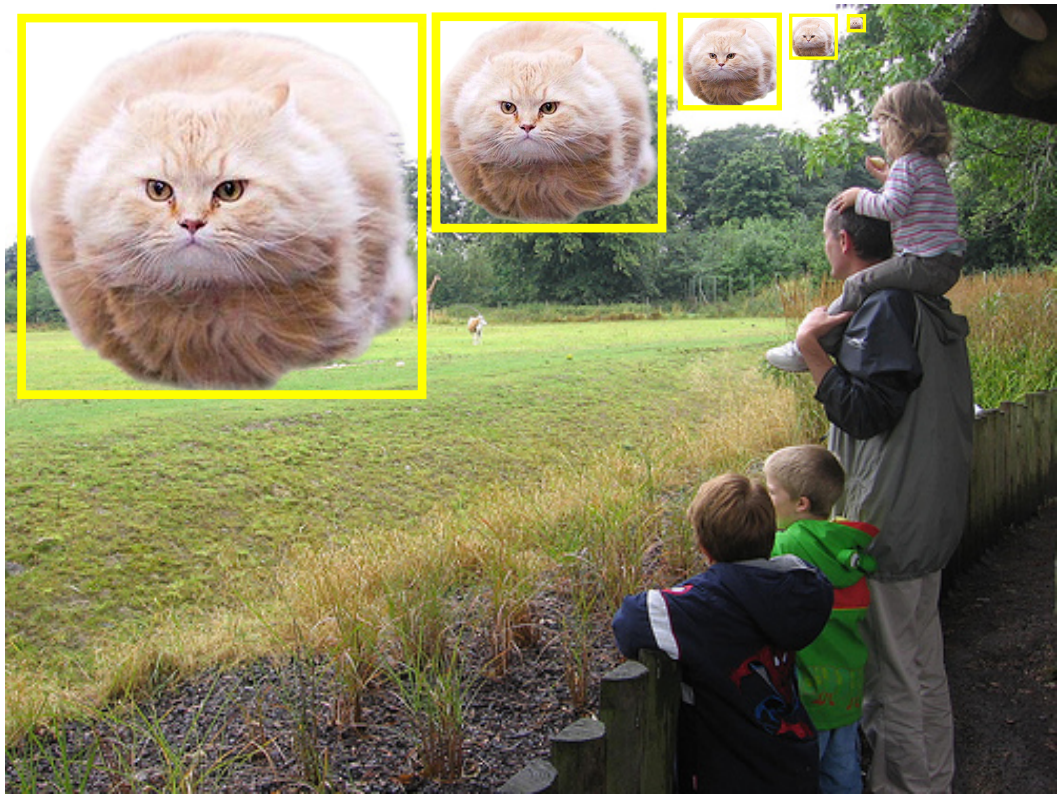
MLP

Softmax clf.

Box regressor

The whole-image feature representation can be improved by making it *multi-scale*

FPN: Improving Scale Invariance and Equivariance



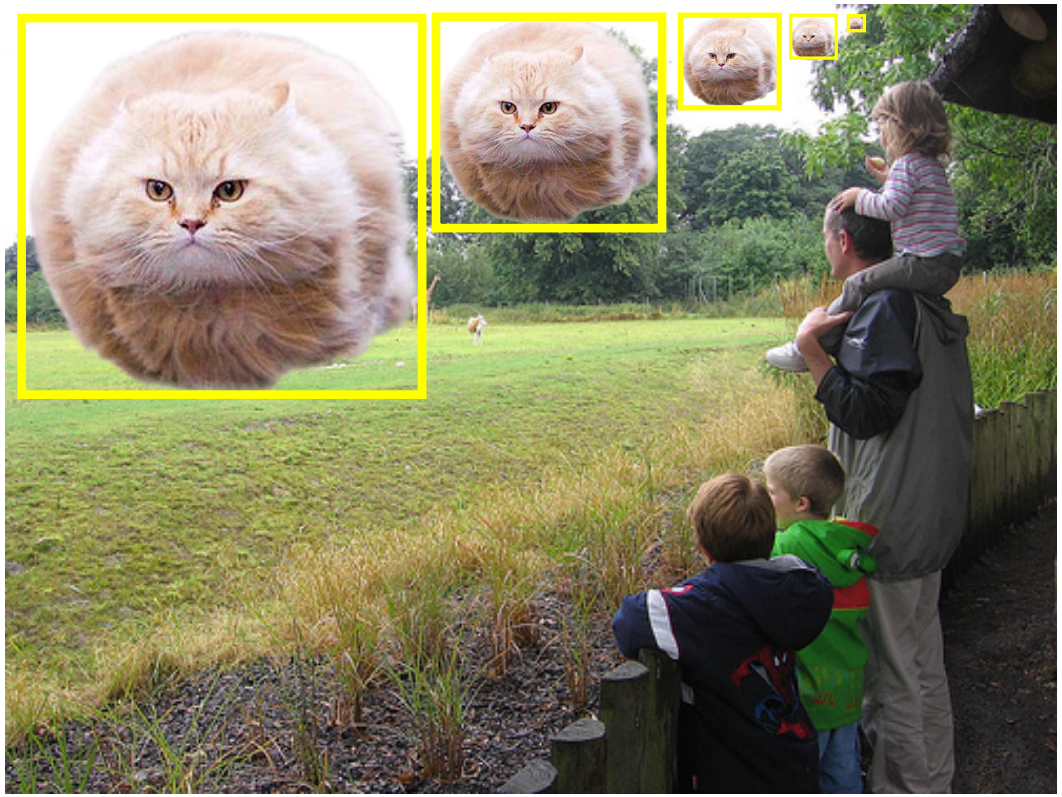
Detectors need to

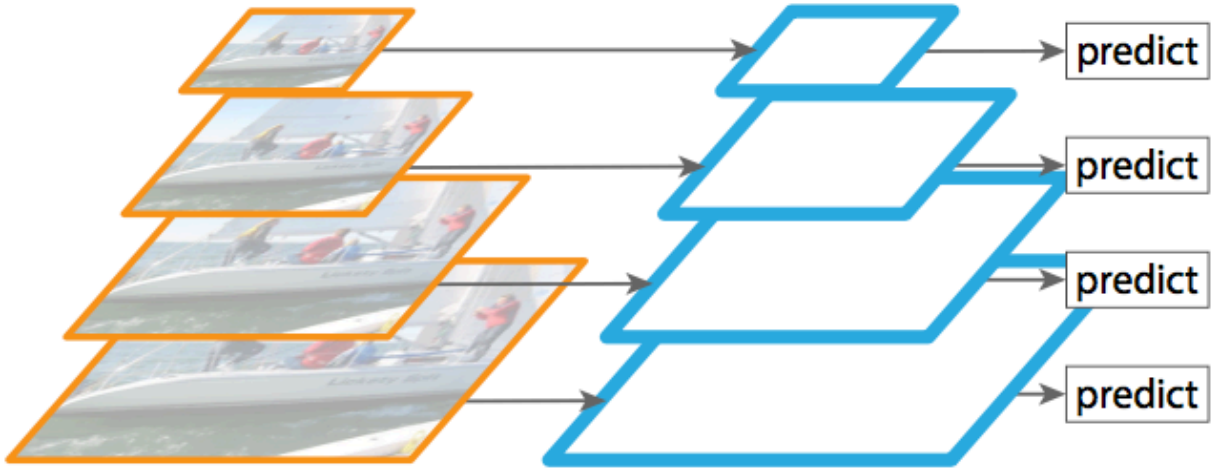
1. Classify (invariance) and
2. Localize (equivariance)

objects over a wide range of scales

FPN improves this ability

Strategy 1: Image Pyramid



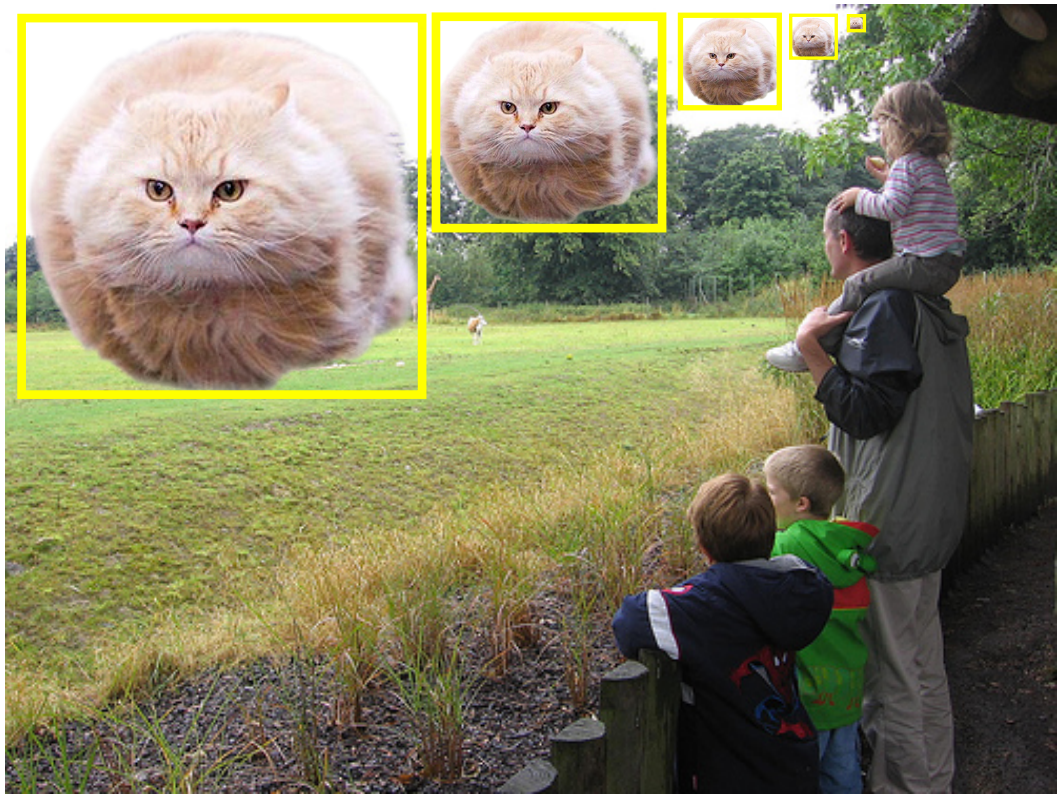


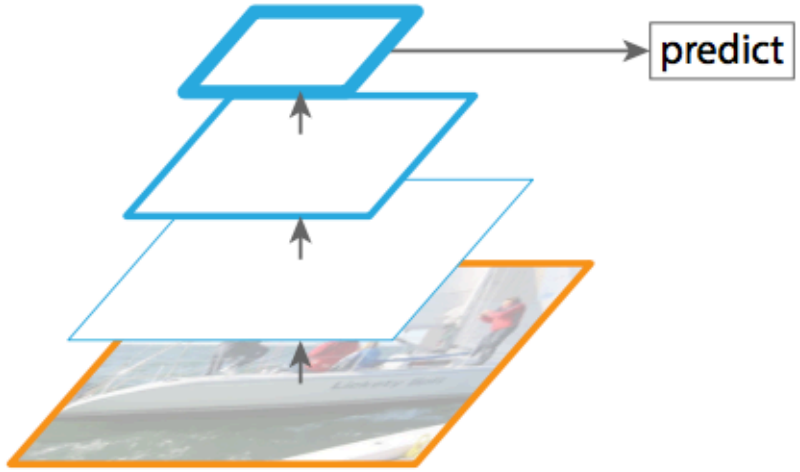
(a) Featurized image pyramid

Standard solution – *slow!*

(E.g., Viola & Jones, HOG, DPM, SPP-net, multi-scale Fast R-CNN, …)

Strategy 2: Multi-scale Features (Single-scale Map)



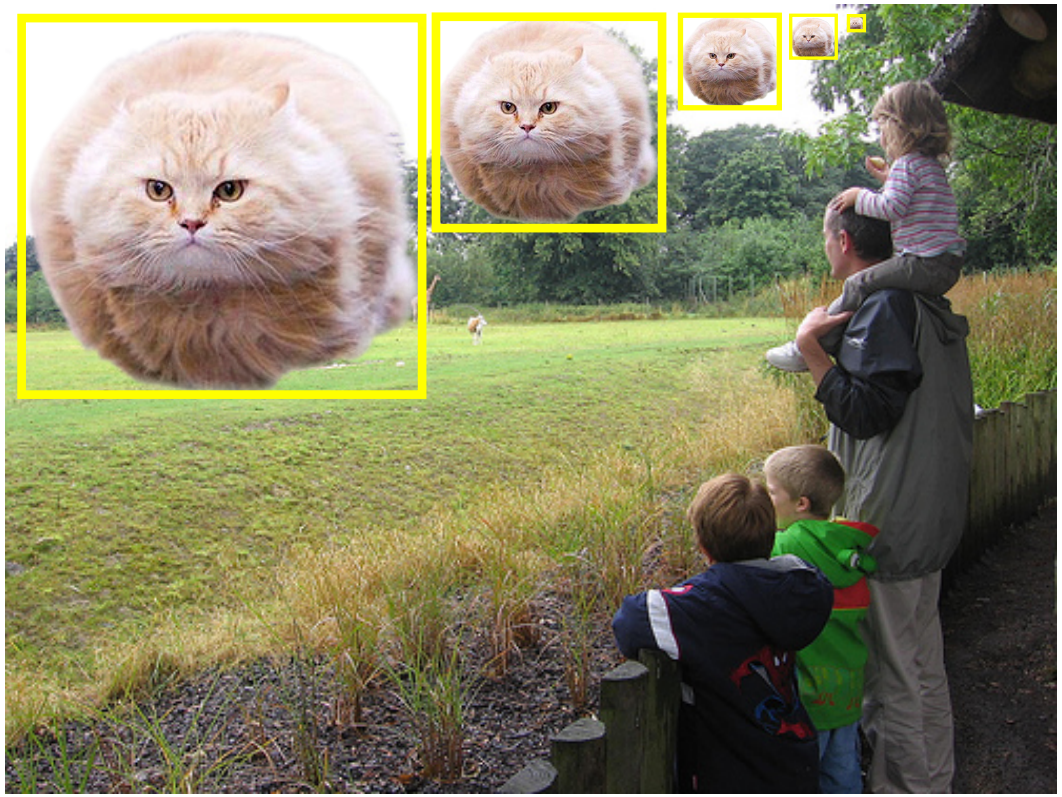


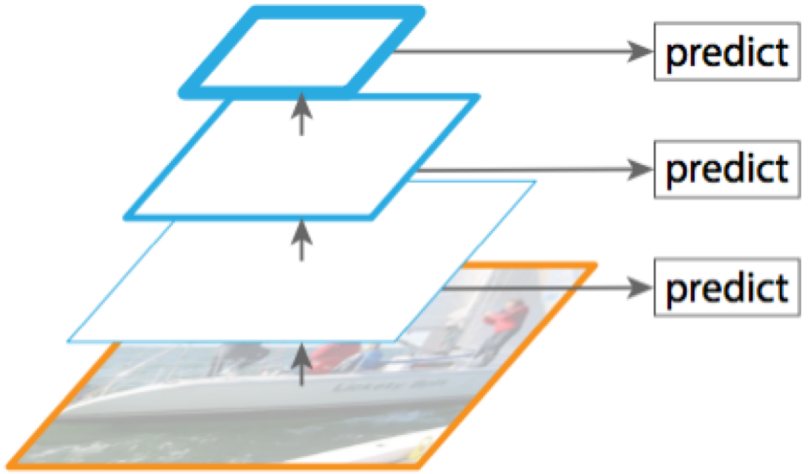
(b) Single feature map

Leave it all to the features – *fast, suboptimal*

(E.g., Fast/er R-CNN, YOLO, …)

Strategy 3: Naïve In-network Pyramid



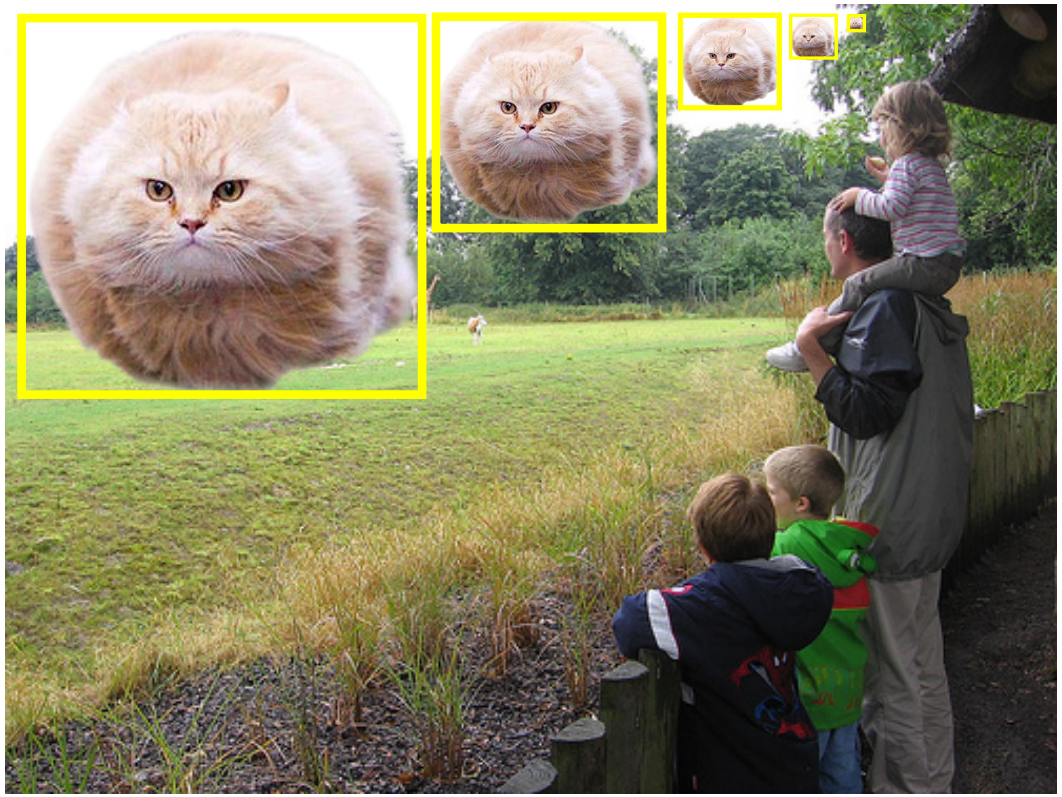


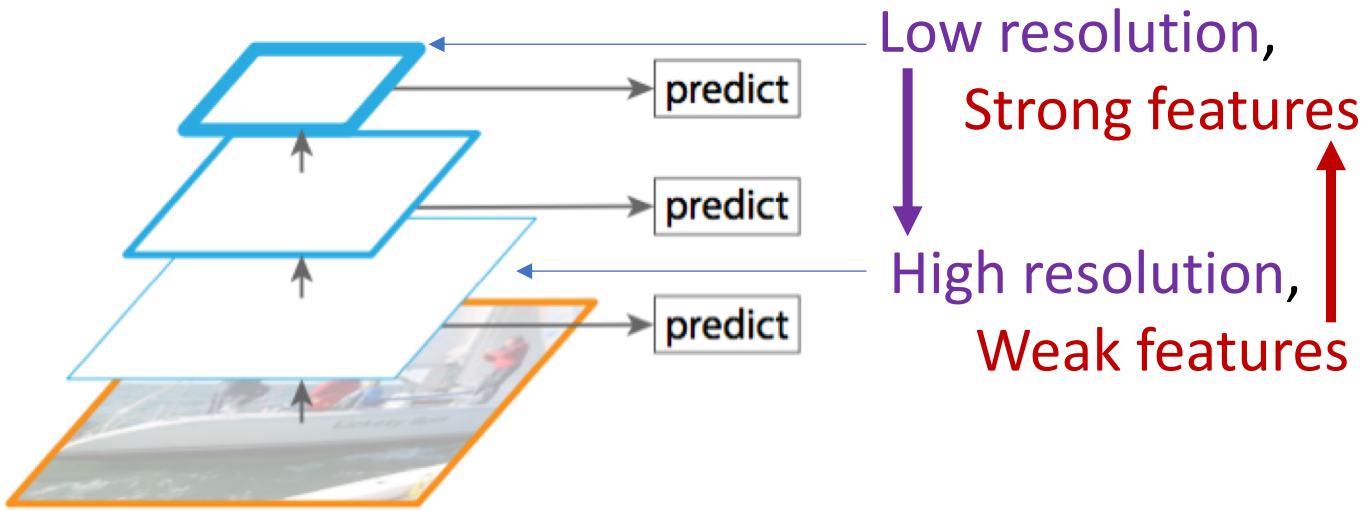
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 3: Naïve In-network Pyramid



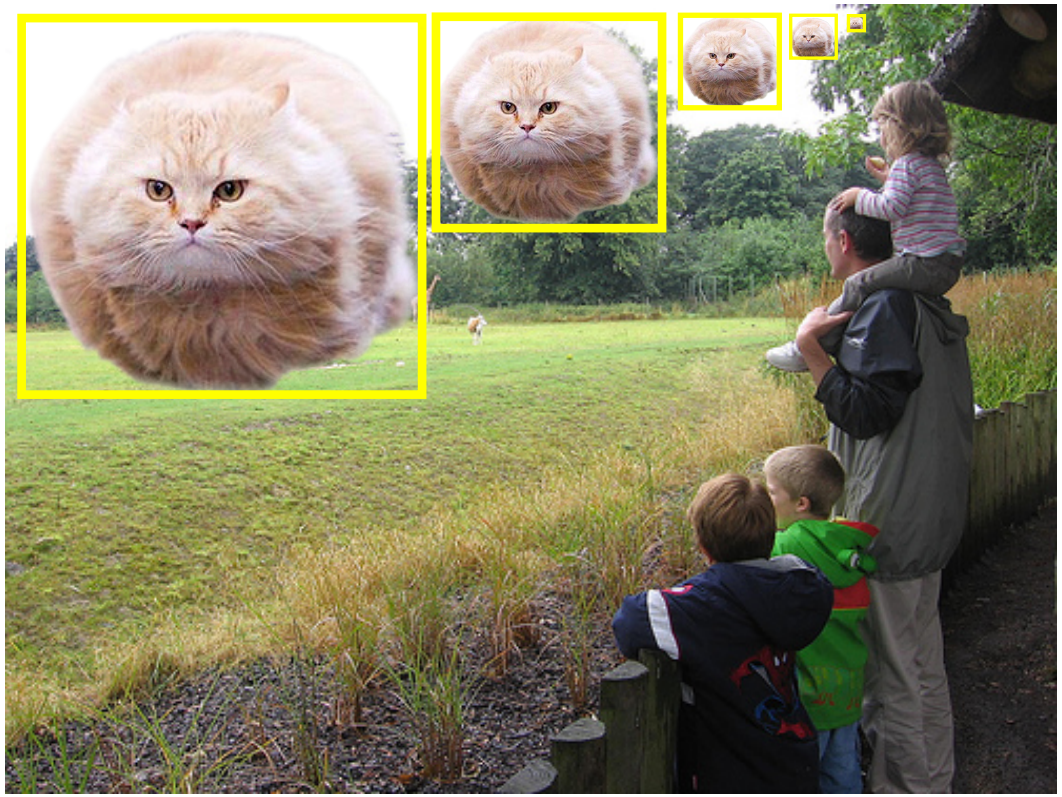


(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 4: Feature Pyramid Network



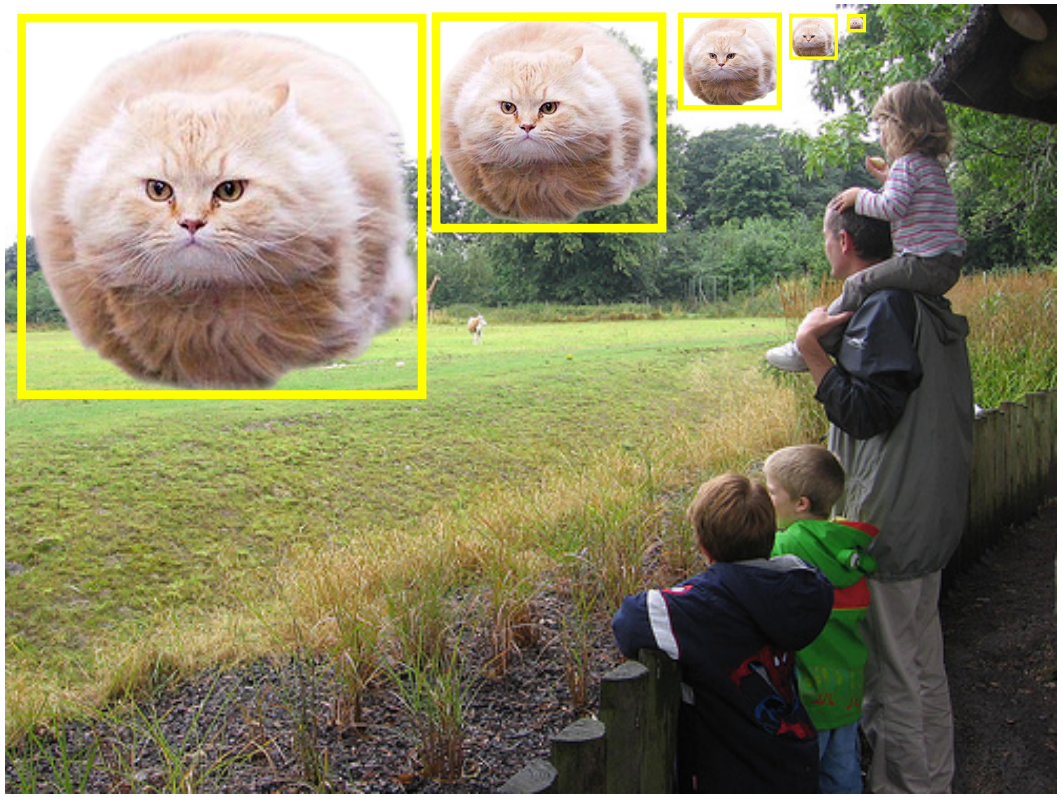


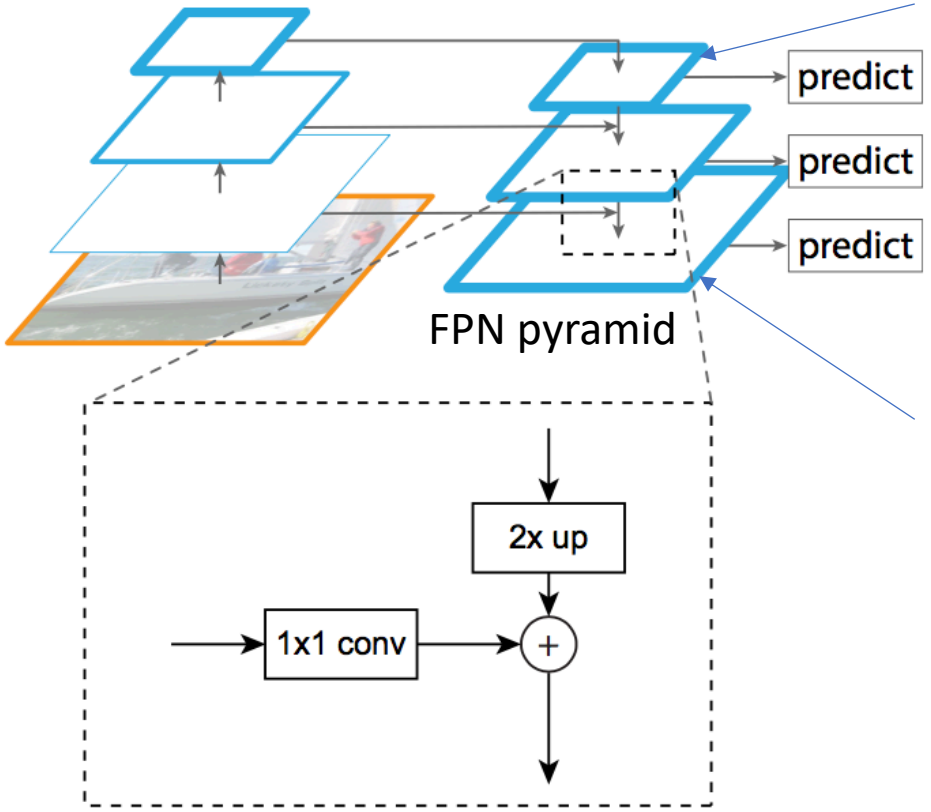
(d) Feature Pyramid Network

Top-down enrichment of high-res features – *fast, less suboptimal*

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

No Compromise on Feature Quality, Still Fast



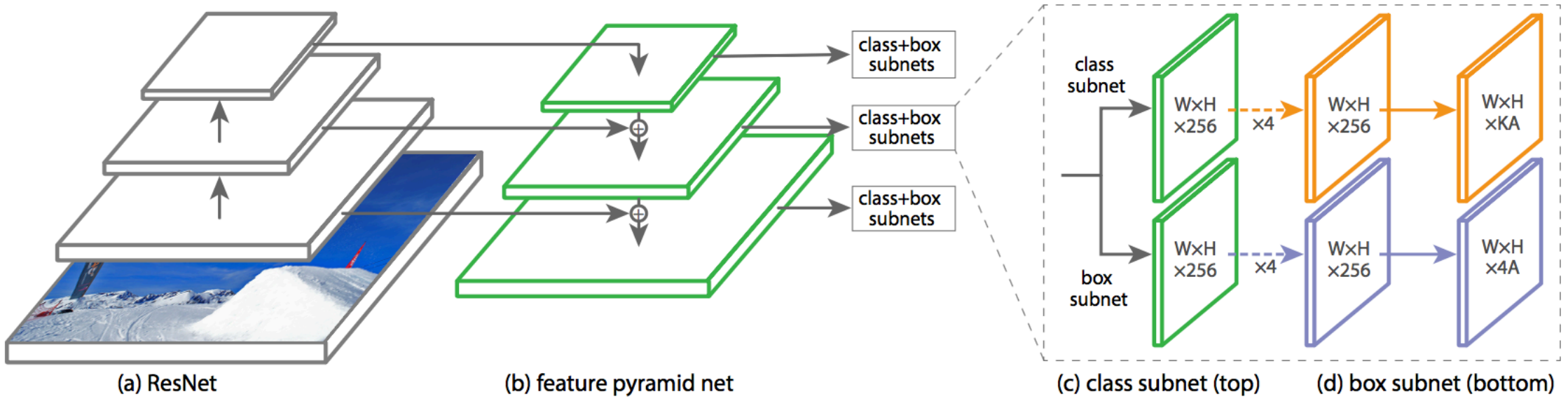


Low resolution,
Strong features

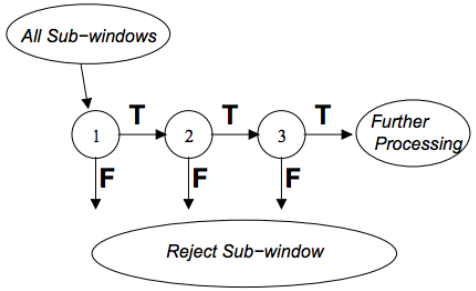
High resolution,
Strong features

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

RetinaNet: Classification and Regression Subnets

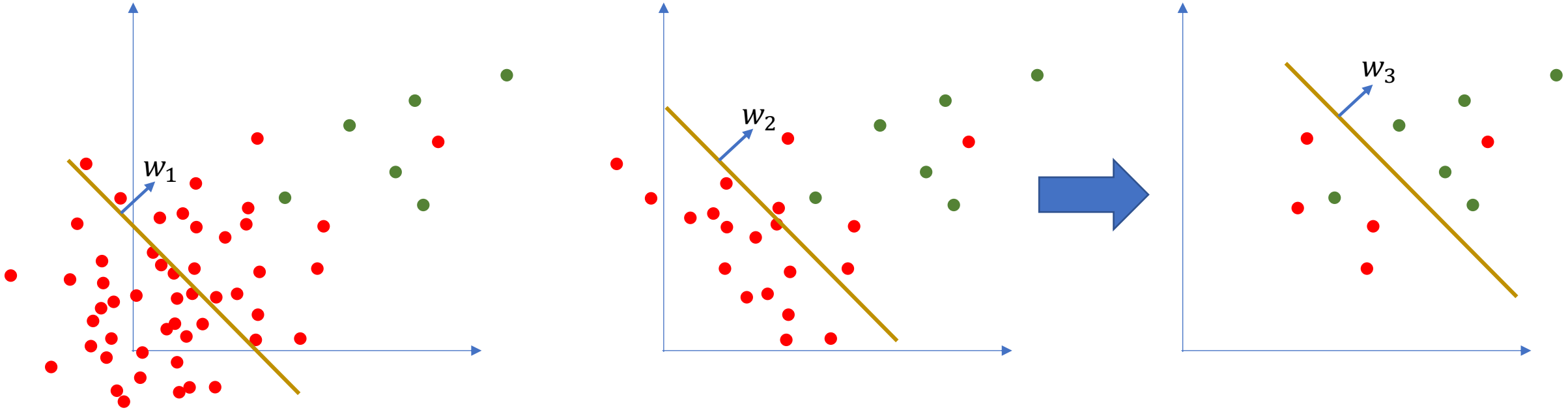


Handling Imbalance with Cascades



eliminate easy background step-by-step, leading to a …

… balanced problem

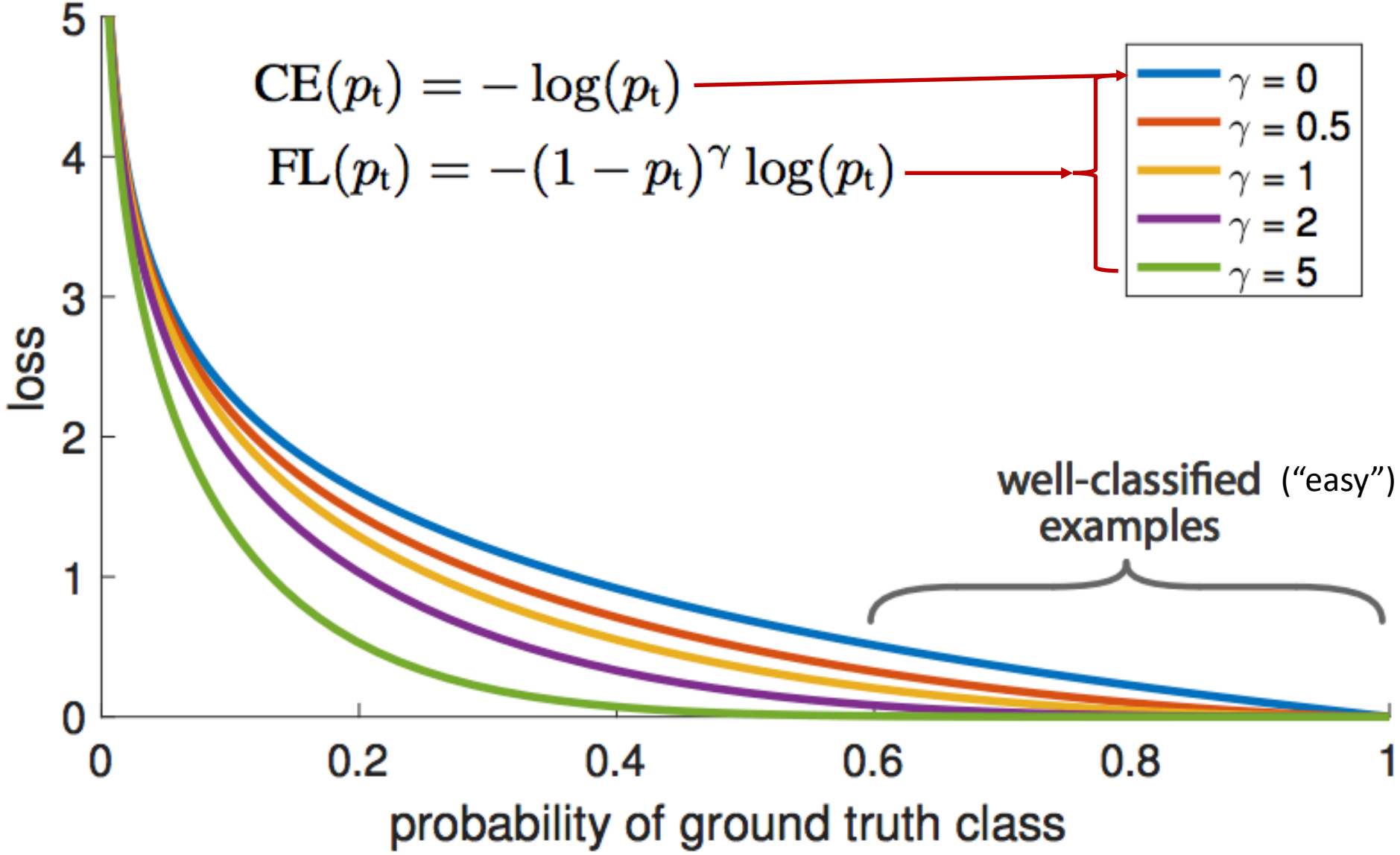


Stage 1

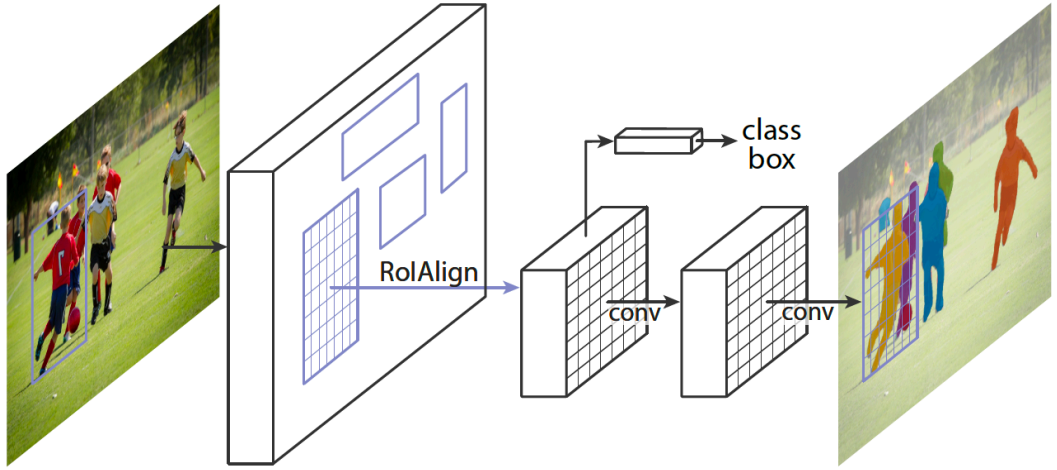
Stage 2

Final Stage

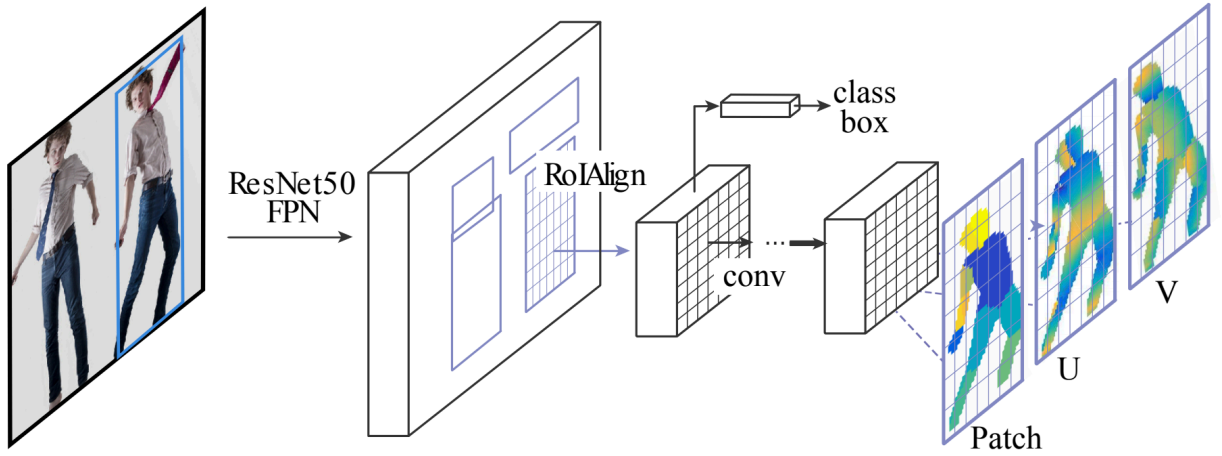
Focal Loss: "Soft" Hard-Example Mining



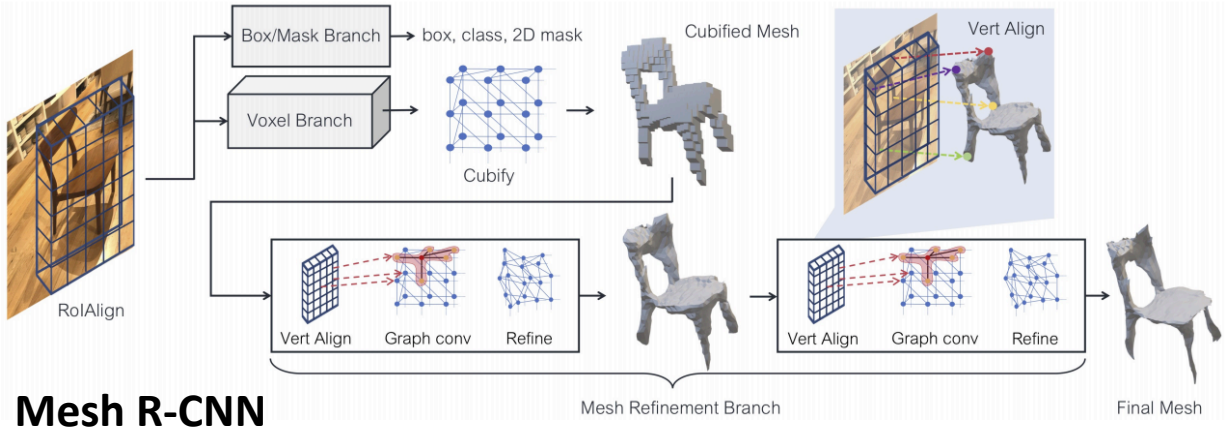
Generalized R-CNN: Adding More Heads



Mask R-CNN
[He, Gkioxari, Dollár, Girshick]



DensePose
[Güler, Neverova, Kokkinos]



Mesh R-CNN
[Gkioxari, Malik, Johnson. ICCV 2019]

Segmentation and detection

- Adversarial examples
- Bottlenecks and U-Nets
- Segmentation
- Face detection
- Object detection

