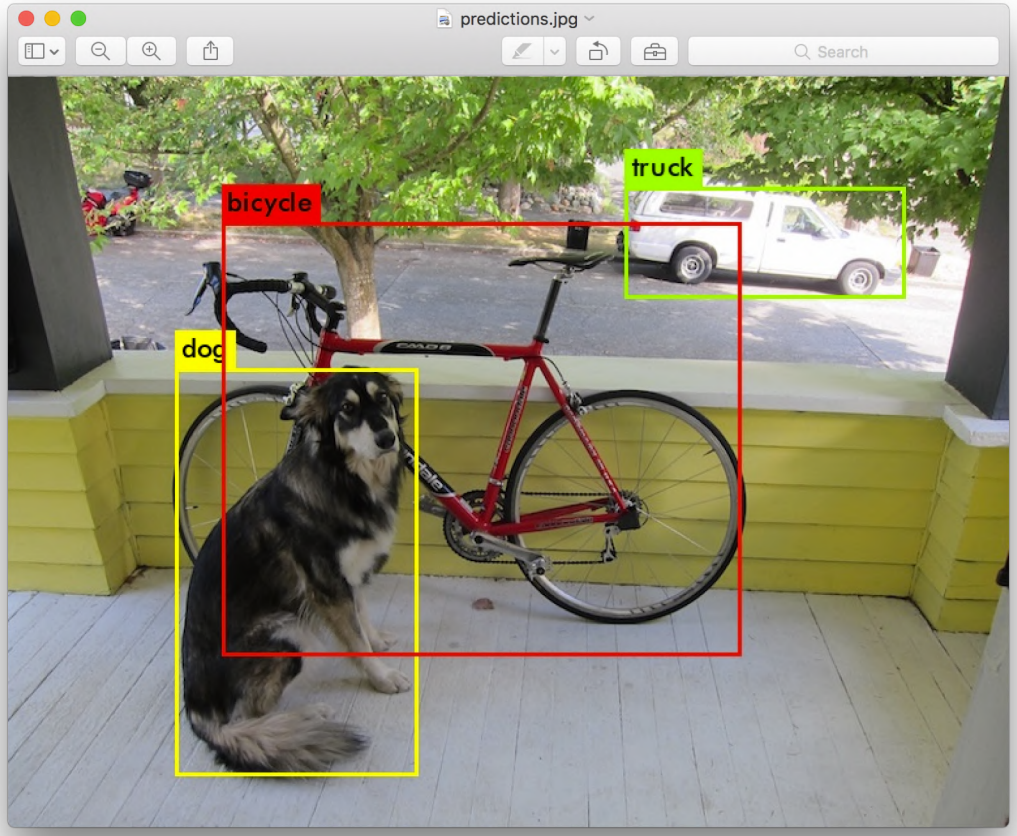
Detection and Localization



Many slides from: Stanford CS231 , Illinois cs498 and Justin Johnson

Computer Vision Tasks

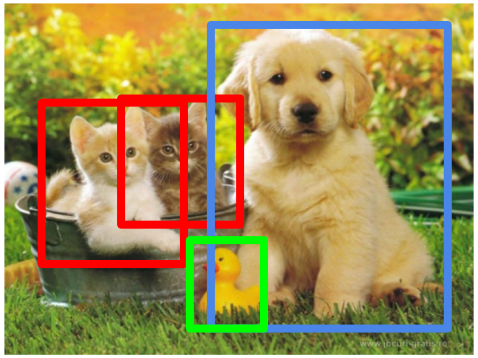
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

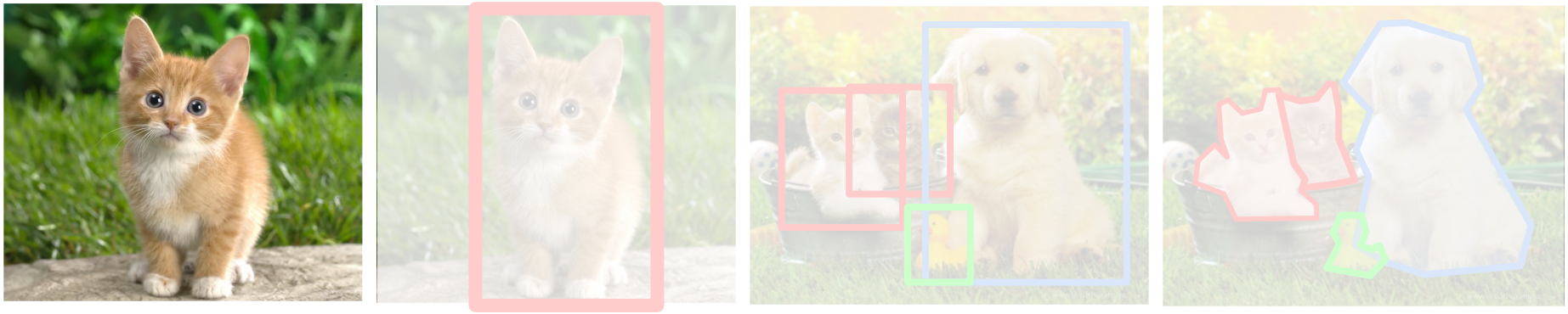
So Far: Image Classification

Classification

Classification + Localization

Object Detection

Instance Segmentation



CAT

CAT

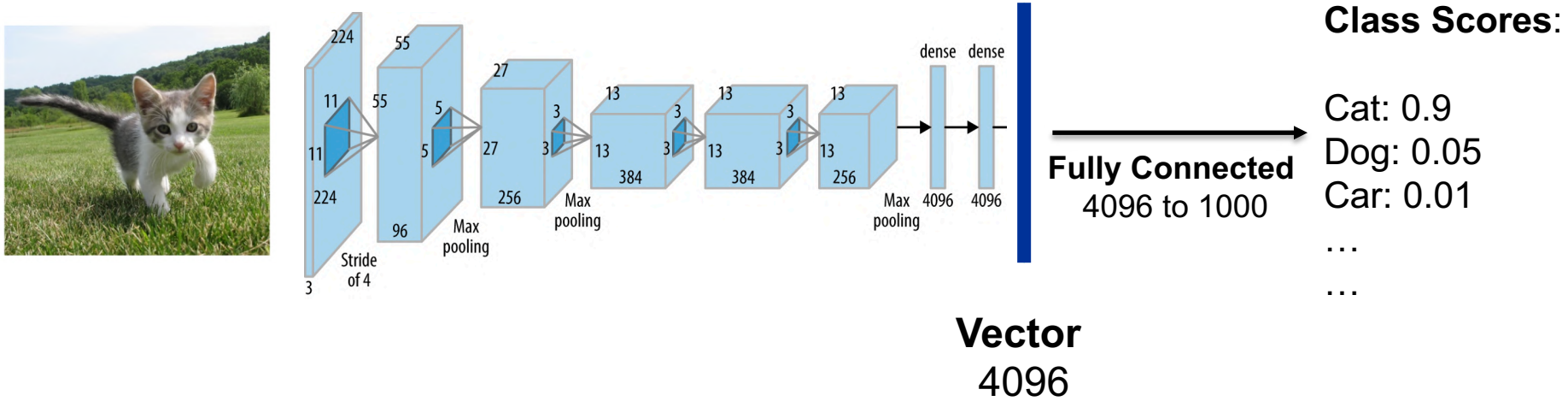
CAT, DOG, DUCK

CAT, DOG, DUCK

Single object

Multiple objects

So Far: Image Classification



Computer Vision Tasks

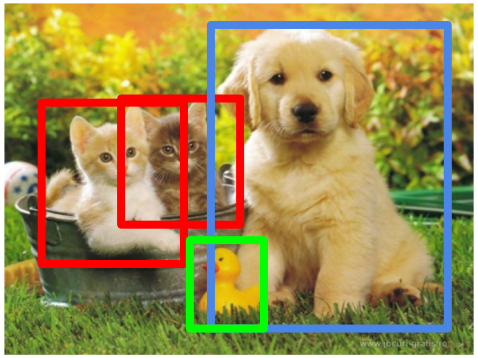
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

Today's Lesson: Object Detection

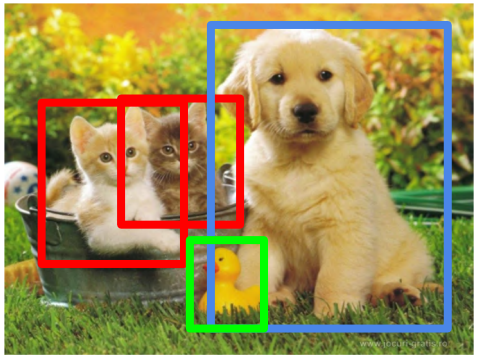
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

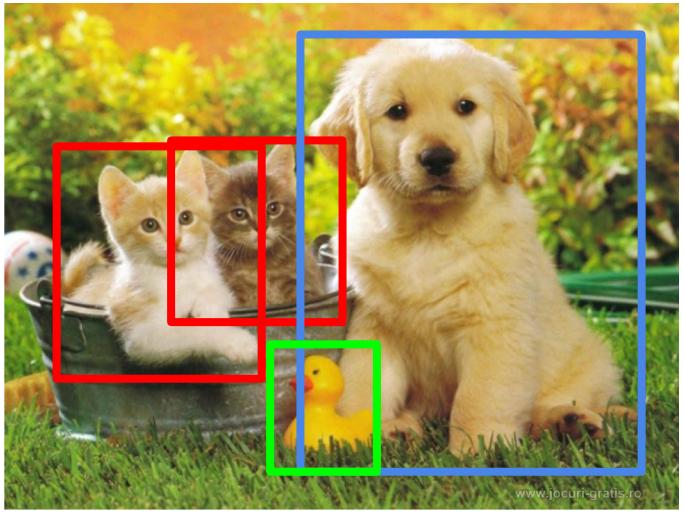
Object Detection: Task Definition

- **Input**: A single RGB image

- **Output**: A set of detected objects:

  For each detected object:

  - Category label (from a fixed set)
  - Bounding box = (x,y,width,height)

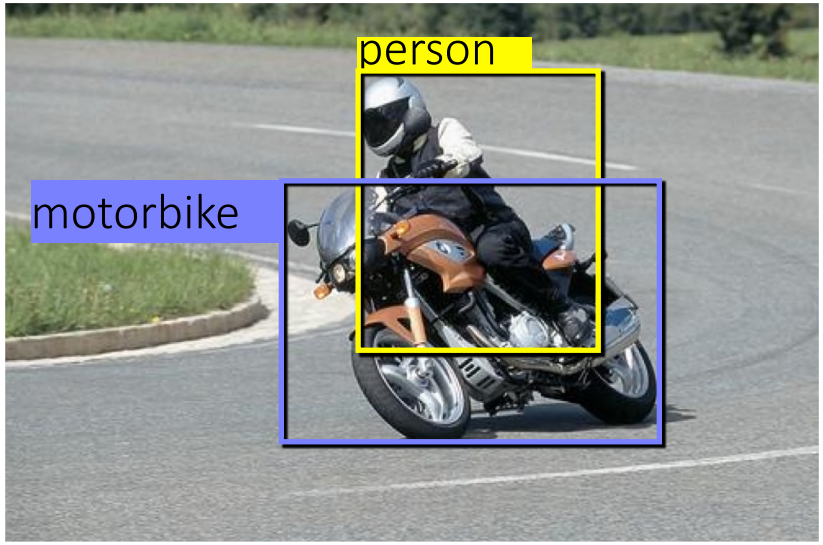


Source: Justin Johnson

Object Detection: Task Definition

classes={airplane, bird, motorbike, person, sofa}

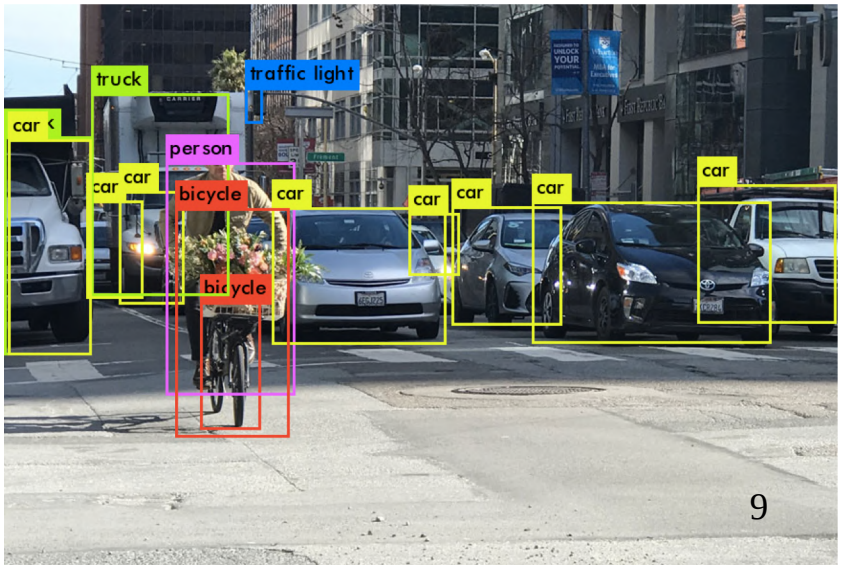
Input



Desired output

What are the challenges of an object detector?

- **Multiple Outputs**: Variable numbers of objects per image
- **Multiple types of Outputs:** Need to predict "what" (class) as well as "where" (bounding box)
- **Multiple Sizes**: multiple objects with variable sizes
- Multiple detections of the same object
- Multiple objects at the same location

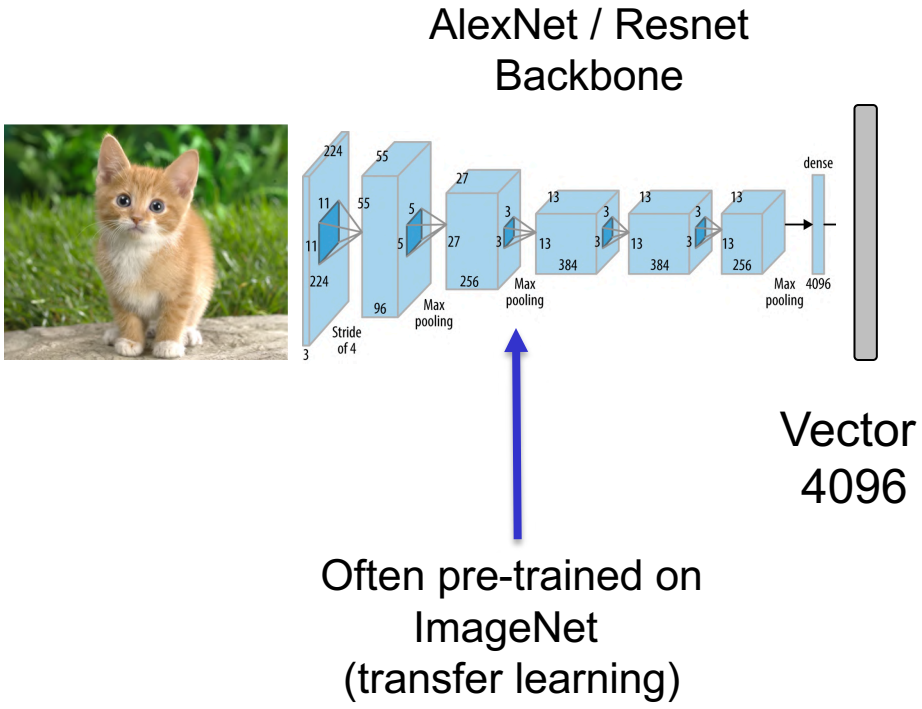


Outline

- Task definition and evaluation
- Conceptual approaches for detection
- Region-Proposal based Convolutional Networks
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single Stage Object Detectors

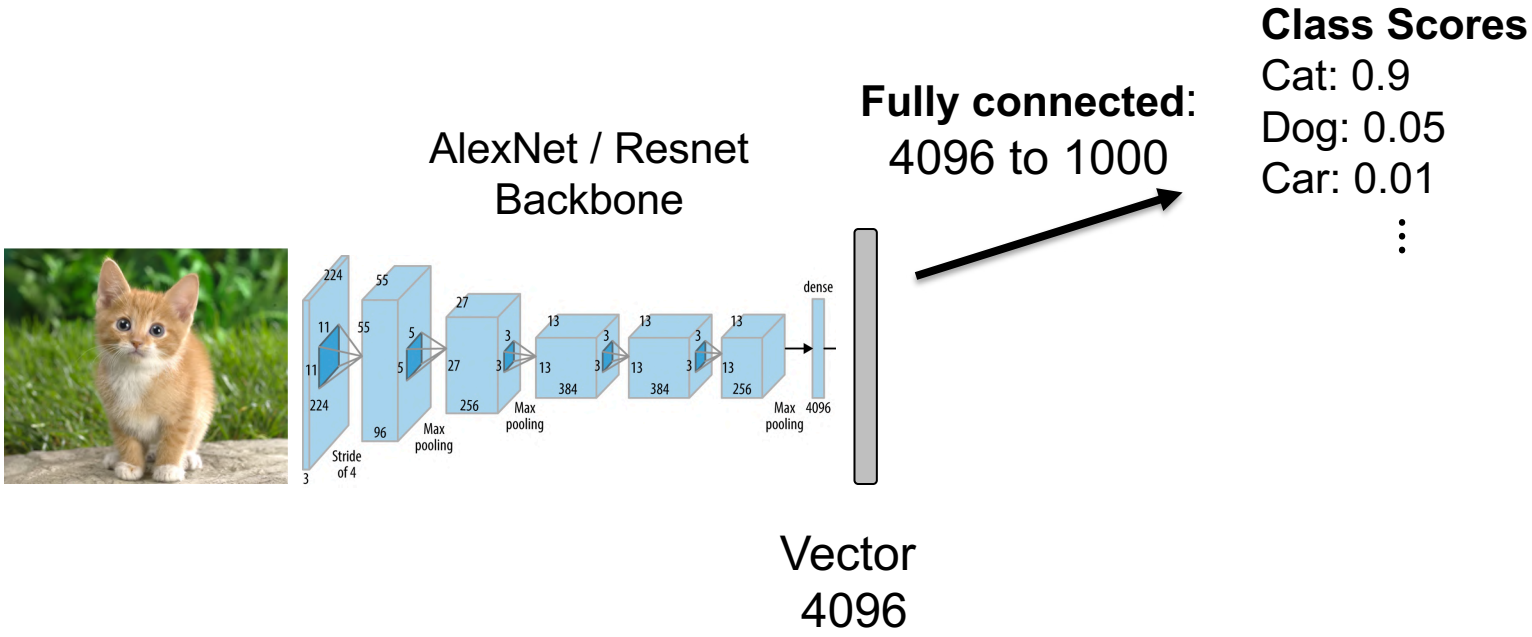
Object Detection: Single Object

- Can be treated as classification + localization



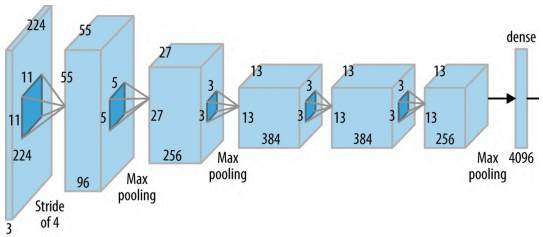
Object Detection: Single Object

- Can be treated as classification + localization



Object Detection: Single Object

- Can be treated as classification + localization



AlexNet / Resnet Backbone

Vector 4096

Fully connected: 4096 to 1000

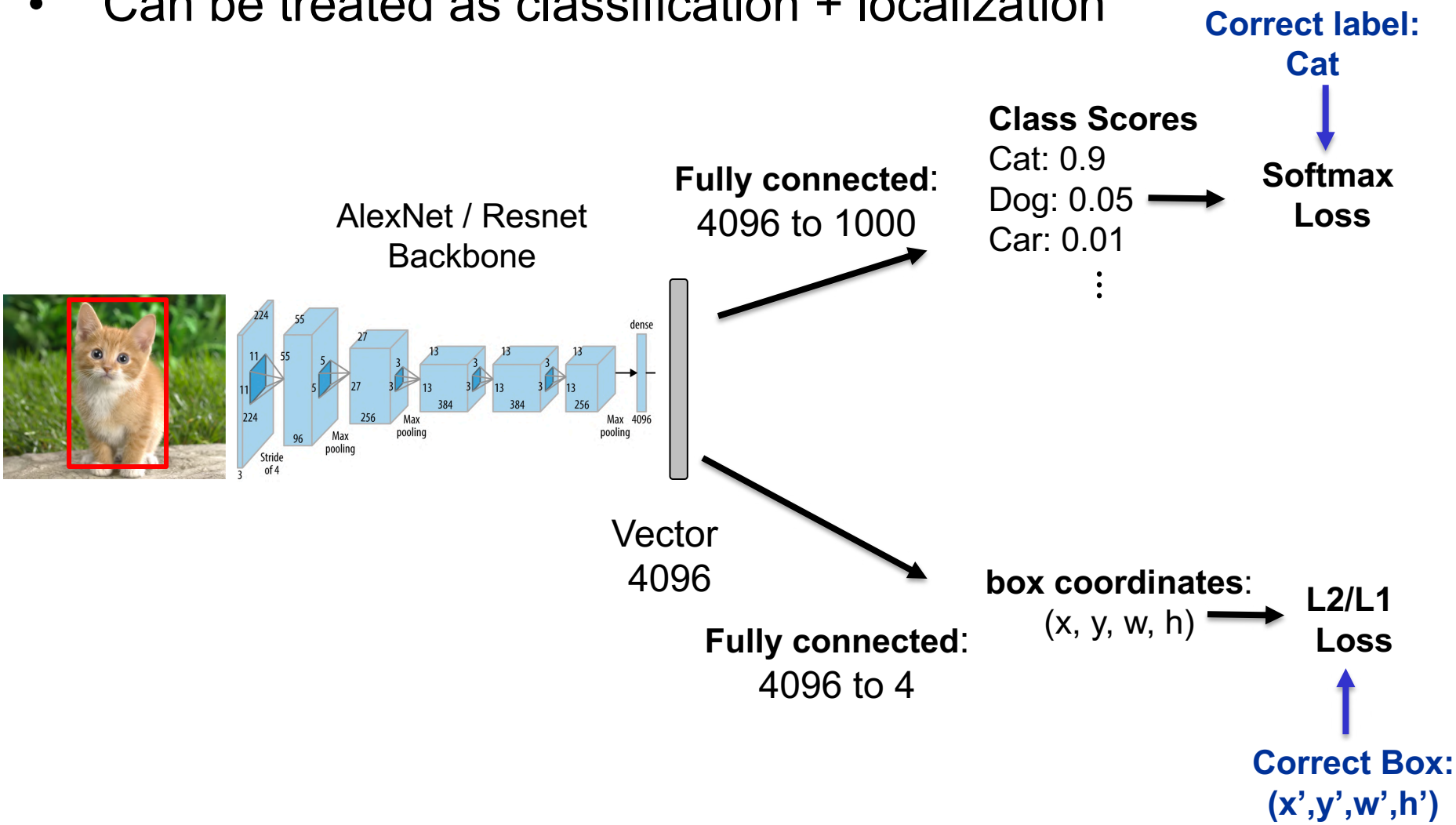
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
⋮

Fully connected: 4096 to 4

box coordinates: (x, y, w, h)

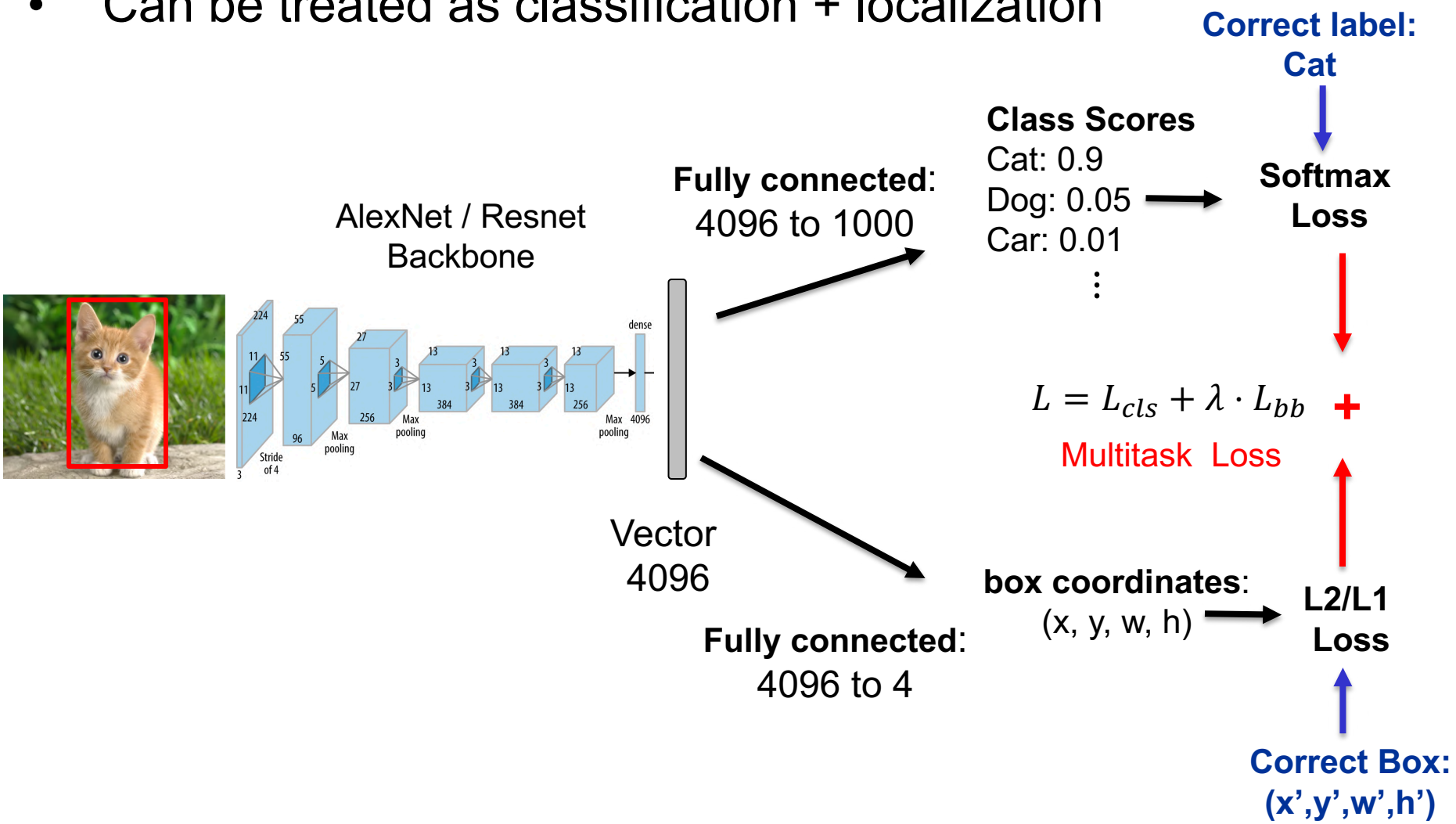
Object Detection: Single Object

- Can be treated as classification + localization



Object Detection: Single Object

- Can be treated as classification + localization



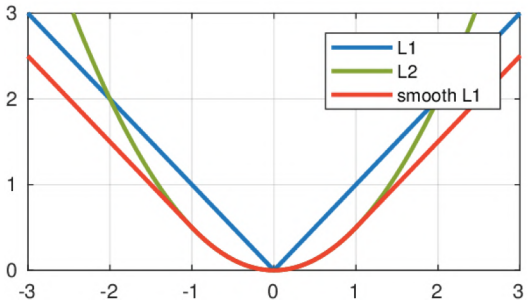
Multi-task loss

- Loss for ground truth class y, predicted class probabilities $P(y)$, ground truth box b, and predicted box b':

$$L(y', y, P, b, b') = \underbrace{-\log P(y')}_{\text{CE loss}} + \underbrace{\lambda L_{\text{reg}}(b, b')}_{\text{regression loss}}$$

- **Regression loss**: *smooth L1 loss* offsets relative to GT

$$L_{\text{reg}}(b, b') = \sum_{i=\{x,y,w,h\}} \text{smooth}_{L_1}(b_i - b_i')$$



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

Single object Evaluation

classification + localization: Regress both

Classification: C classes

Input: Image

Output: Class label

Evaluation metric: Accuracy

CAT

Localization:

Input: Image

Output: Box in the image (x, y, w, h)

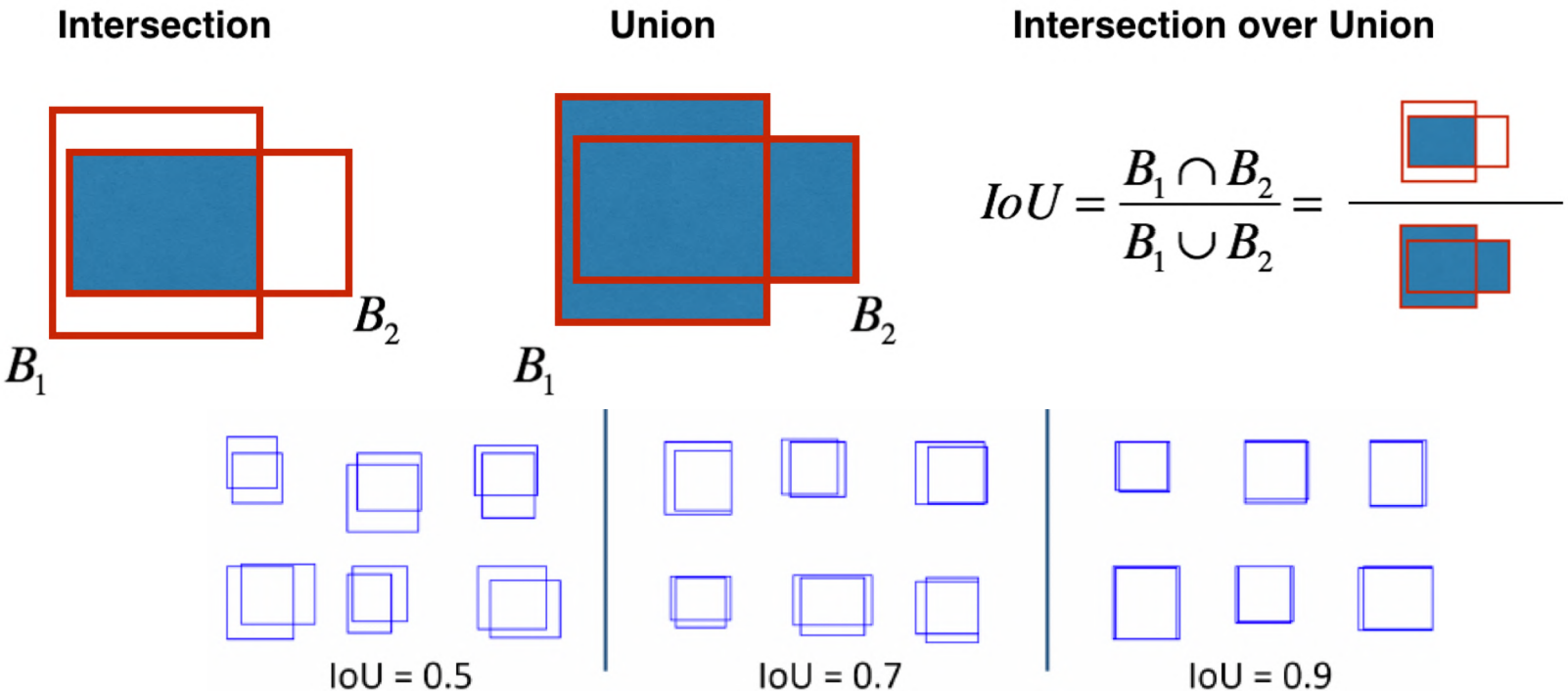
Evaluation metric: Intersection over Union

(x, y, w, h)

Comparing Boxes

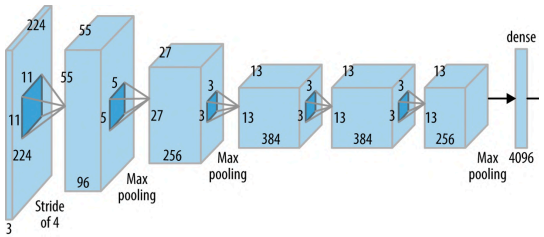
Intersection over Union (IoU).

- The IoU score is between [0..1] and is scale invariant.
- IoU is calculated between predicted box & GT box
- IoU > 0.5 is “decent”, IoU > 0.7 is “very good”, and IoU > 0.9 is “almost perfect”.



Object Detection: Single object

- A single object detection can be treated as classification + localization



AlexNet / Resnet Backbone

Fully connected: 4096 to 1000

Vector 4096

Fully connected: 4096 to 4

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

Correct label: Cat

Softmax Loss

$L = L_{cls} + \lambda \cdot L_{bb}$ +

Multitask Loss

box coordinates: (x, y, w, h)

L2/IoU Loss

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

Problem: Images can have more than a single object

Object Detection: Multiple objects

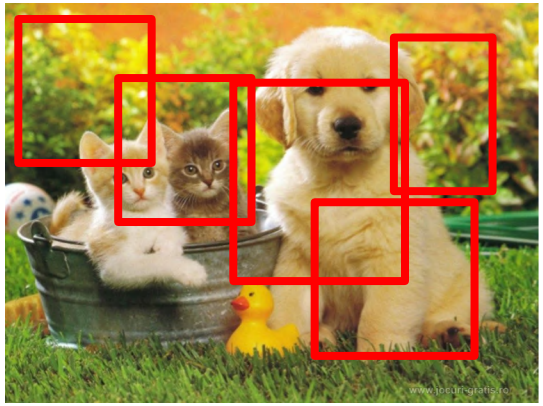
→ CAT, (x, y, w, h) = 4 numbers

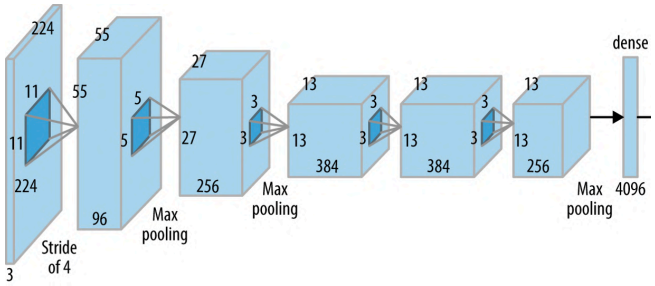
→ DOG, (x, y, w, h)
CAT, (x, y, w, h) = 8 numbers

→ DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h) = 16 numbers

Object Detection: Sliding Window

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

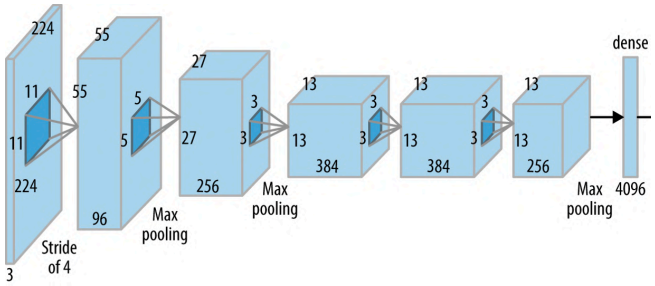




Source: Justin Johnson

Object Detection: Sliding Window

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

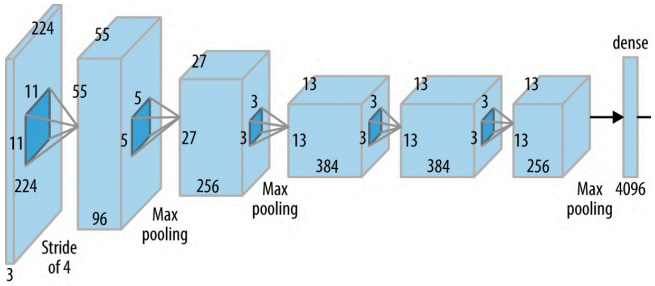


Dog: No
Cat: No
Duck: No
Background: Yes

Source: Justin Johnson

Object Detection: Sliding Window

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

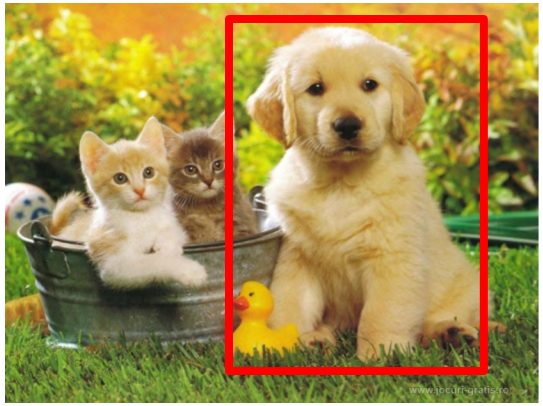


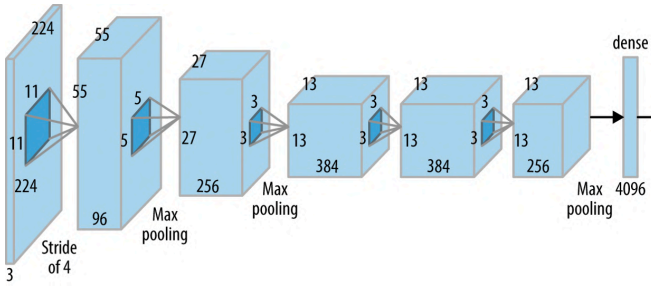
Dog: No
Cat: Yes
Duck: No
Background: No

Source: Justin Johnson

Object Detection: Sliding Window

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



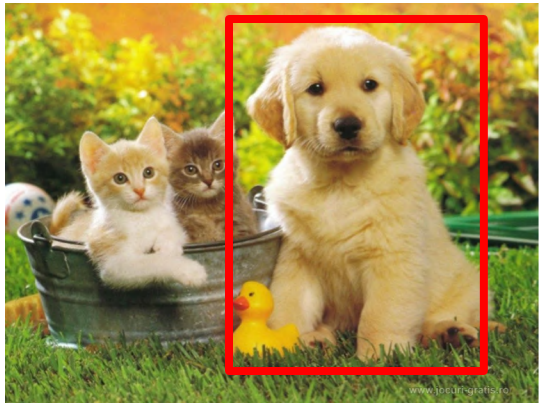


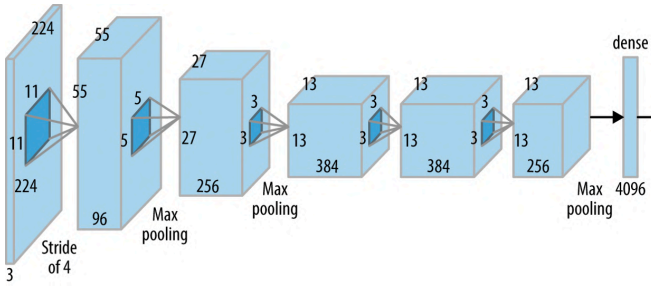
Dog: Yes
Cat: No
Duck: No
Background: No

Source: Justin Johnson

Object Detection: Sliding Window

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





Dog: Yes
Cat: No
Duck: No
Background: No

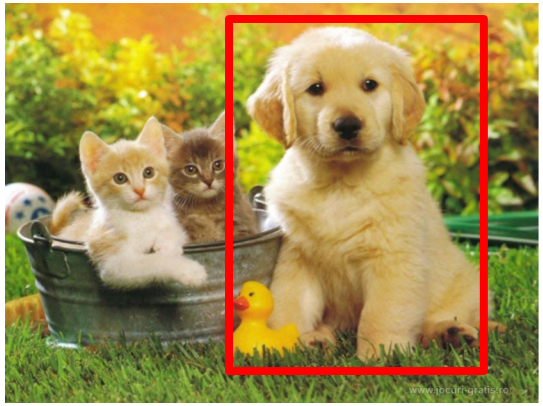
Question:
How many possible crops are there in an image of size HxW?

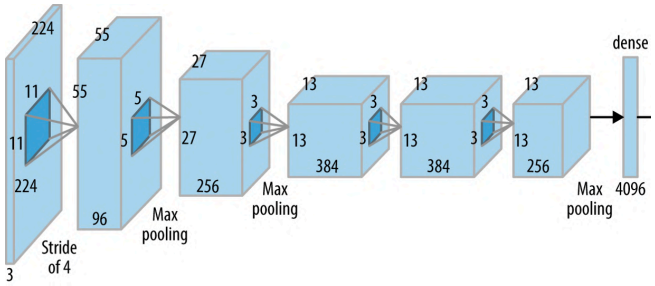
Answer:
Consider an image of size WxH
Total Possible boxes: $= O(W^2H^2)$

Source: Justin Johnson

Object Detection: Sliding Window

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





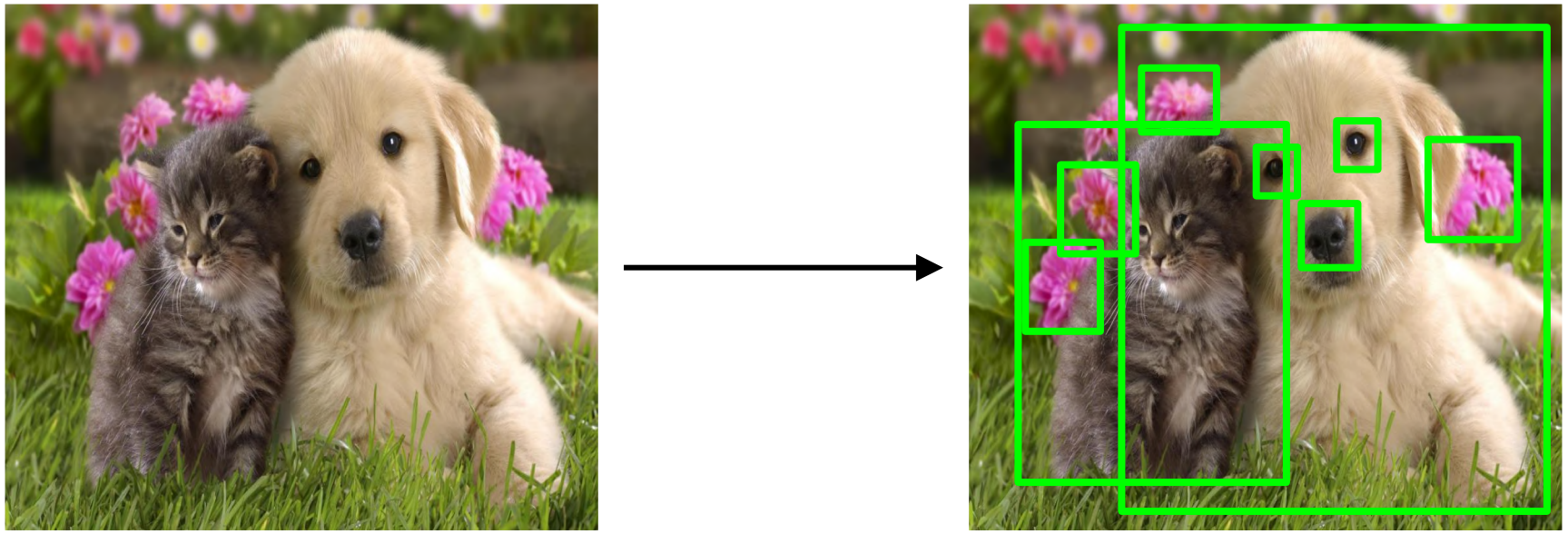
Dog: Yes
Cat: No
Duck: No
Background: No

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

Source: Justin Johnson

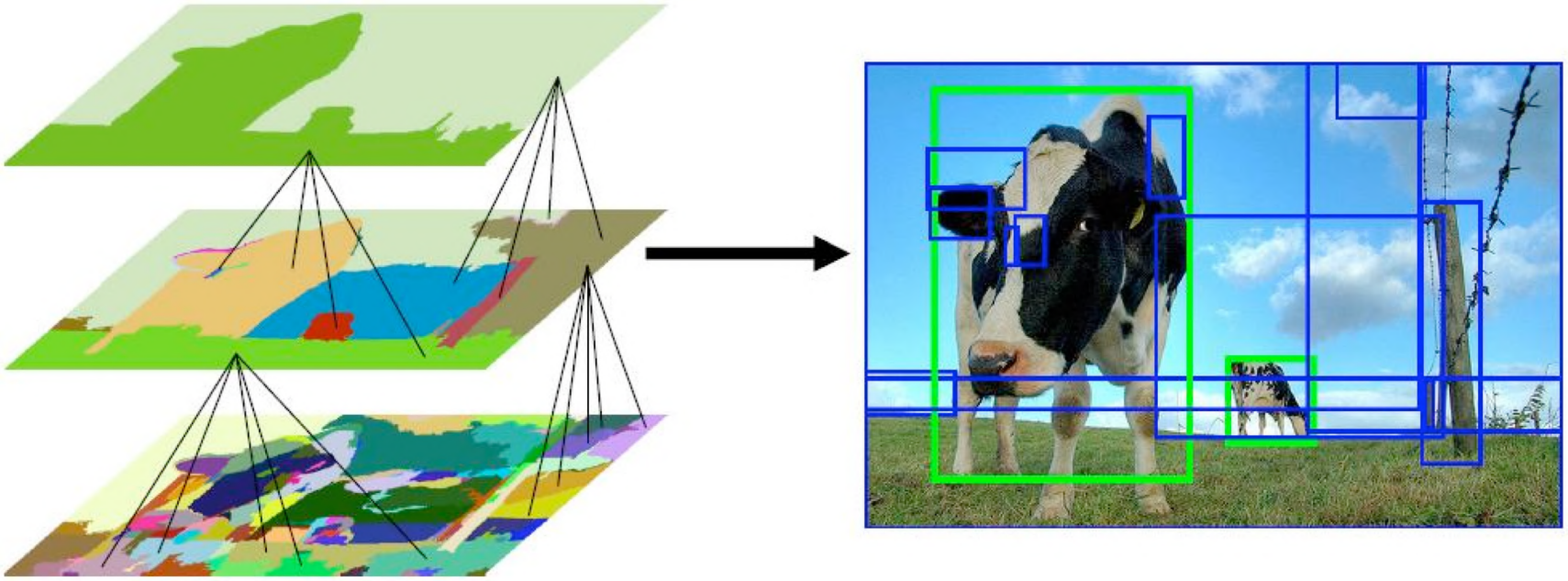
Solution: Region proposal

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. **Selective Search** gives 2000 regions
- Region proposals in a few seconds on CPU



Region proposal: Selective Search

- **Selective Search**: Use hierarchical segmentation; Start with small *superpixels* and group segments based on homogeneity cues

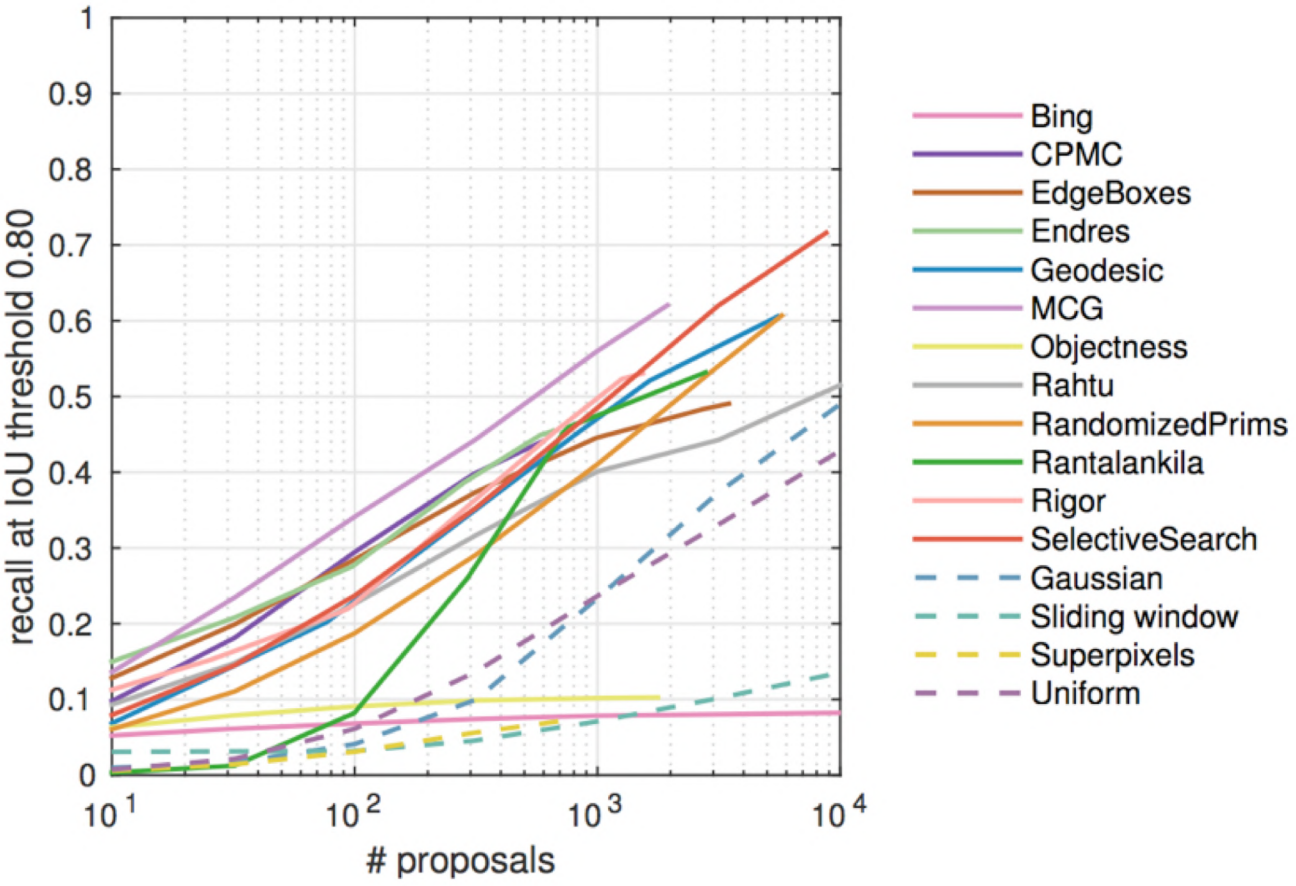


J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, Selective Search for Object Recognition, IJCV 2013

Region proposal: Many other choices

- Tens of ways of generating candidates ("proposals")

What fraction of GT objects have proposals near them?



Hosang et al, "What makes for effective detection proposals?", PAMI 2015

Object Detection: Region proposal

We will cover three Region Proposal approaches:

1. R-CNN (Region CNN):
   - Region proposal using classical image processing approaches
   - Independent CNN for each ROI
2. Fast R-CNN
   - Region proposal using classical image processing approaches
   - Shared CNN and feature maps
3. Faster R-CNN
   - Region proposal using CNN

R-CNN: Region-Based CNN

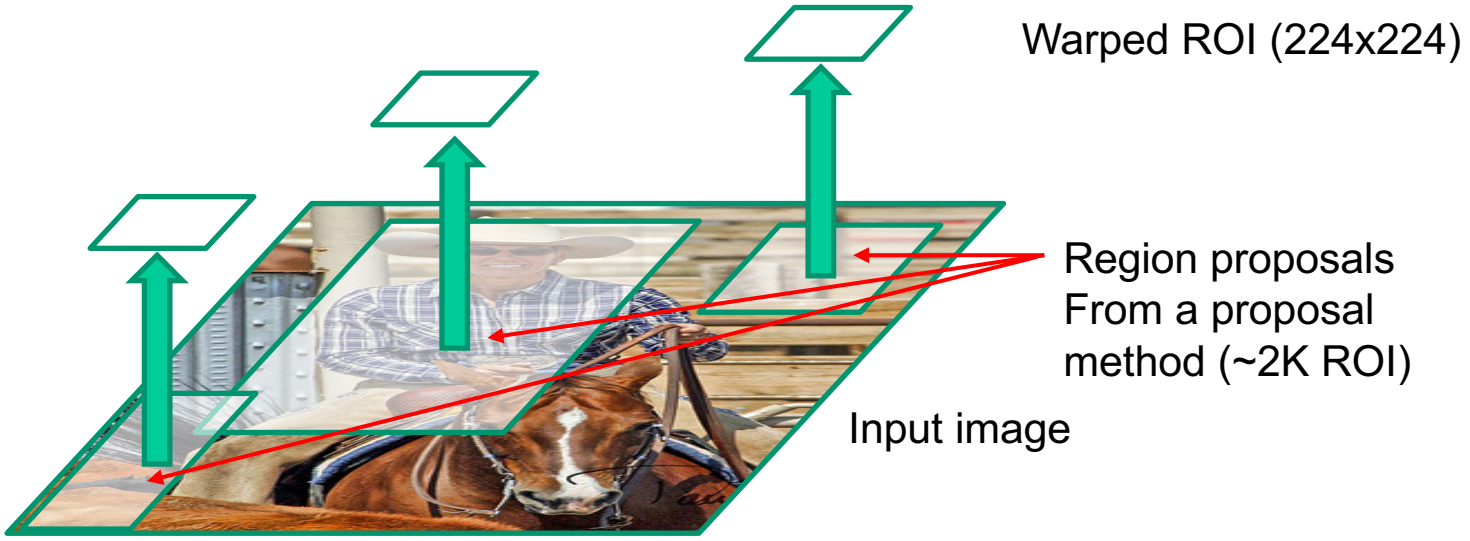
Input image

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN

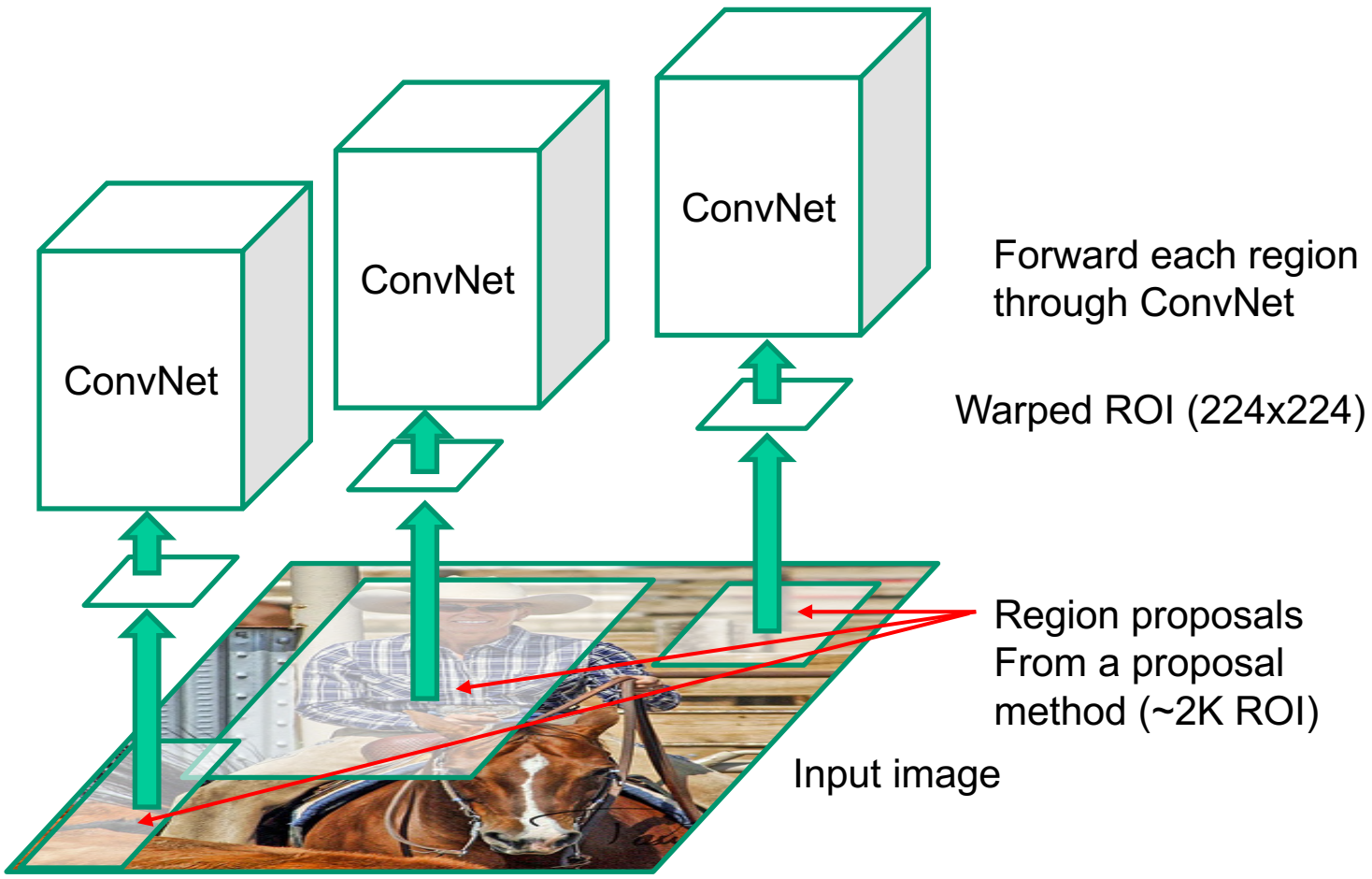
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN



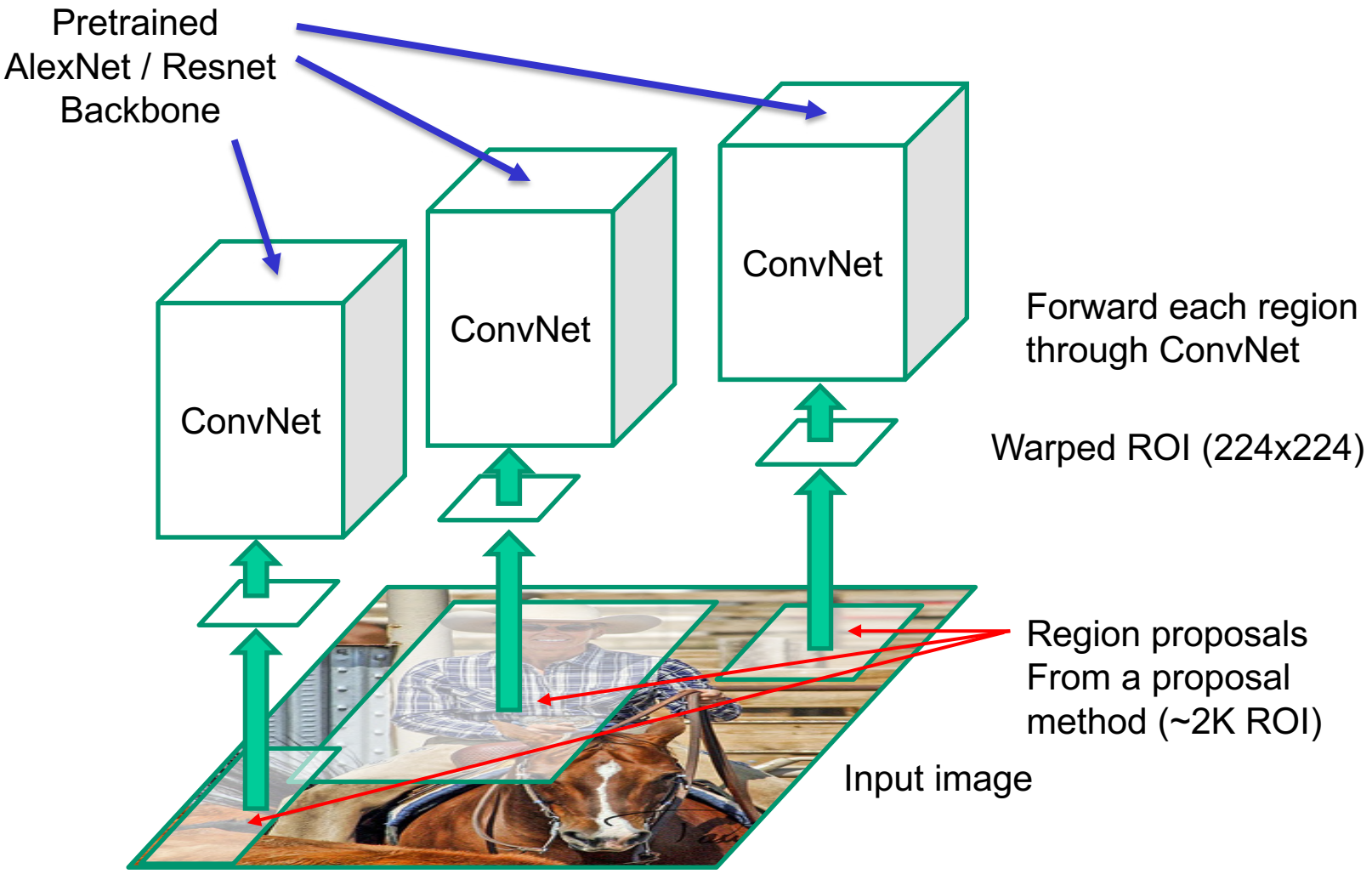
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN



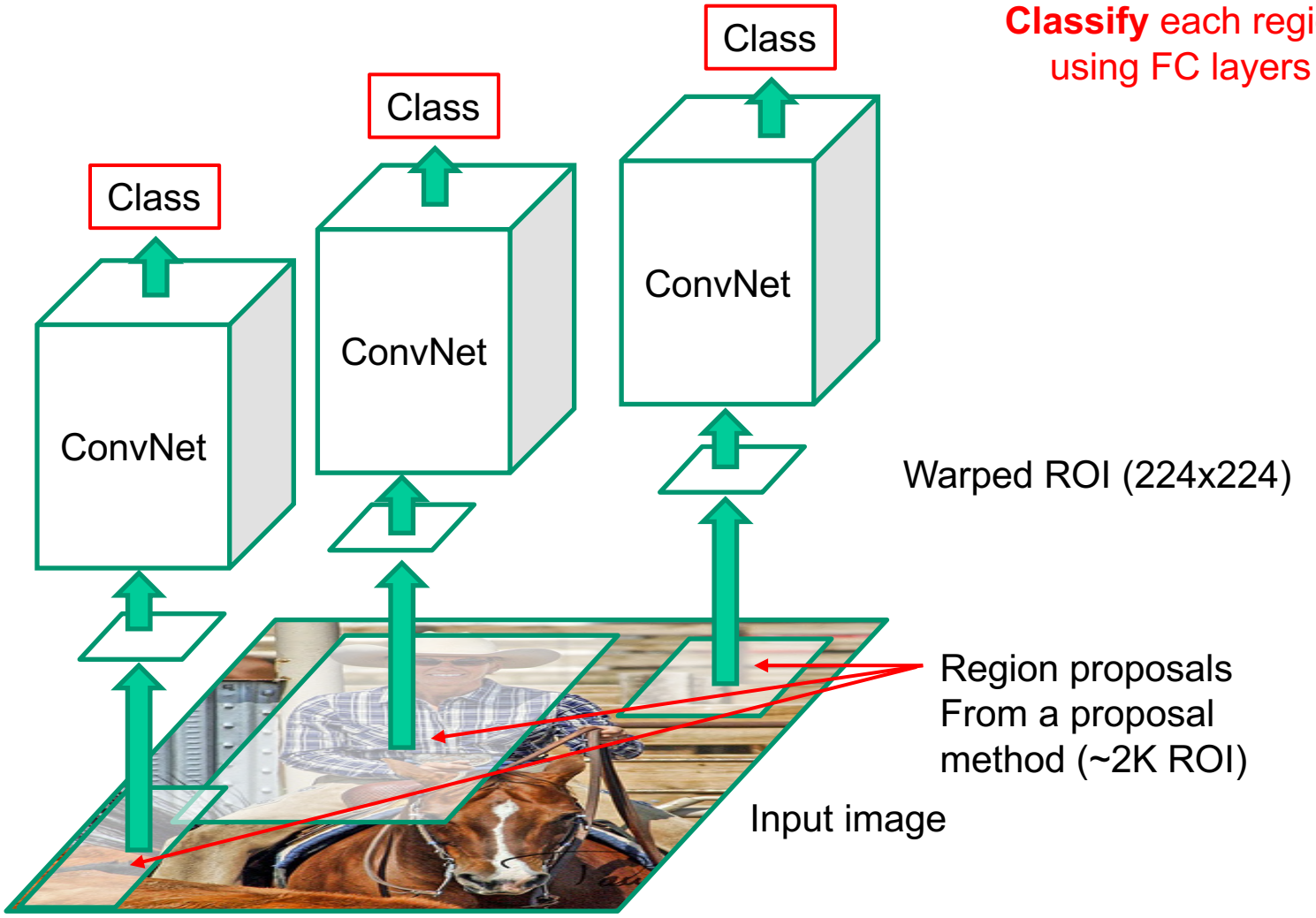
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN



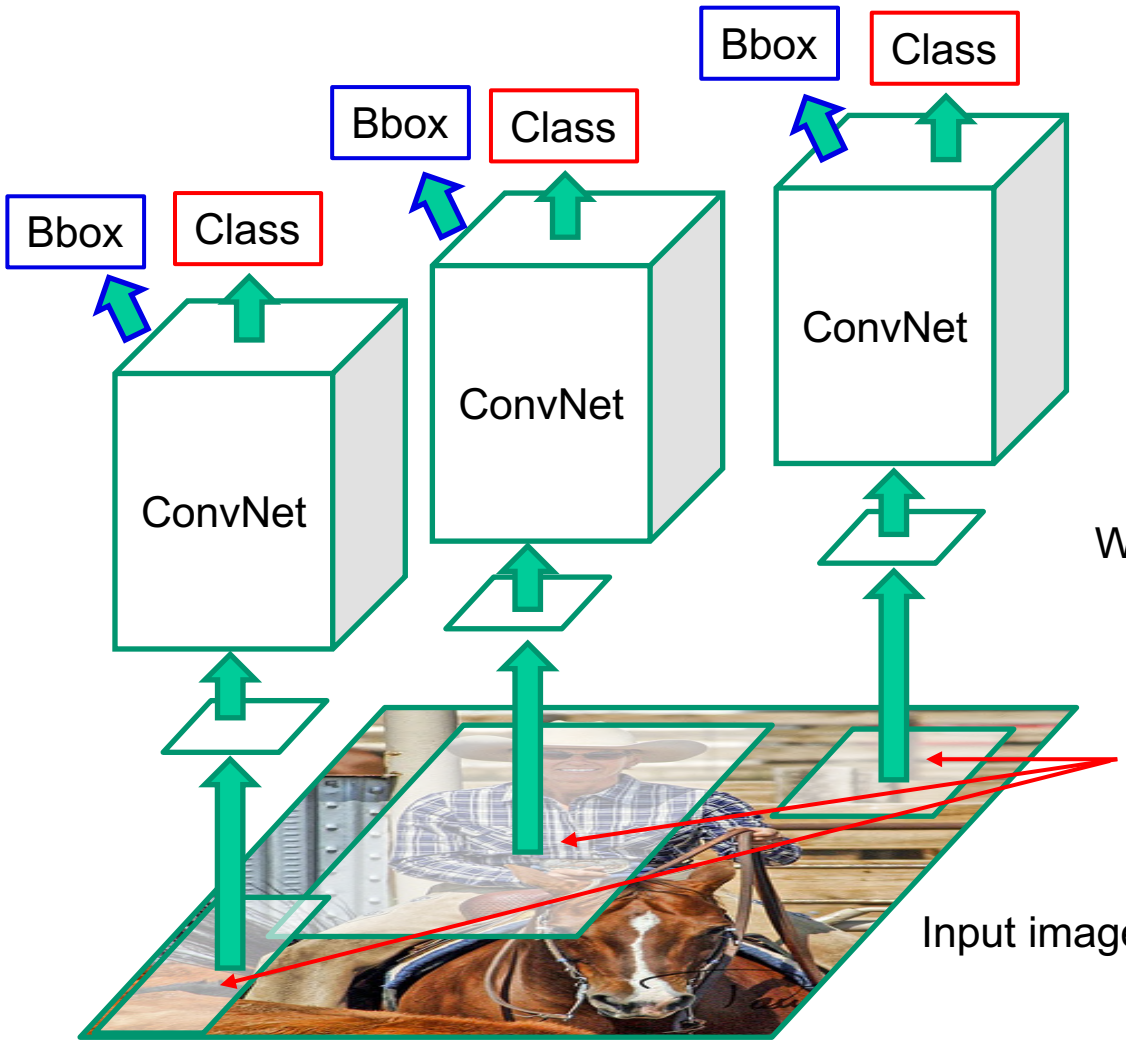
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN



R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

R-CNN: Region-Based CNN



Classify each region using FC layers

Regress Bounding Box: predict "transform" to correct the ROI - (t_x, t_y, t_h, t_w)

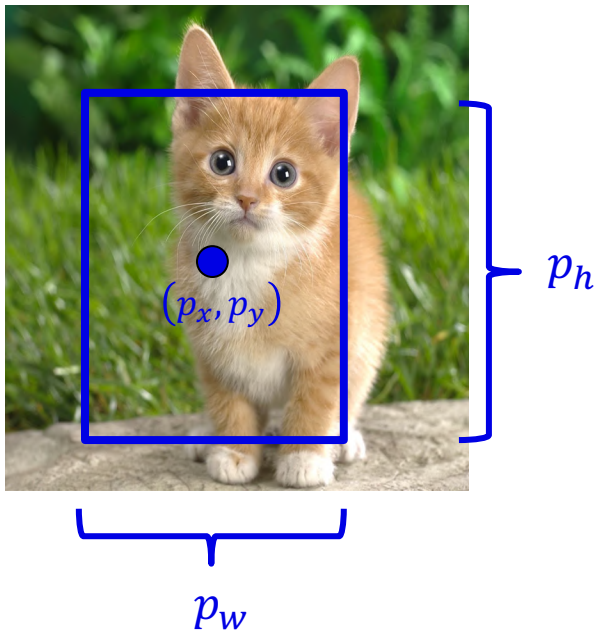
Warped ROI (224x224)

Region proposals From a proposal method (~2K ROI)

Input image

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR14.

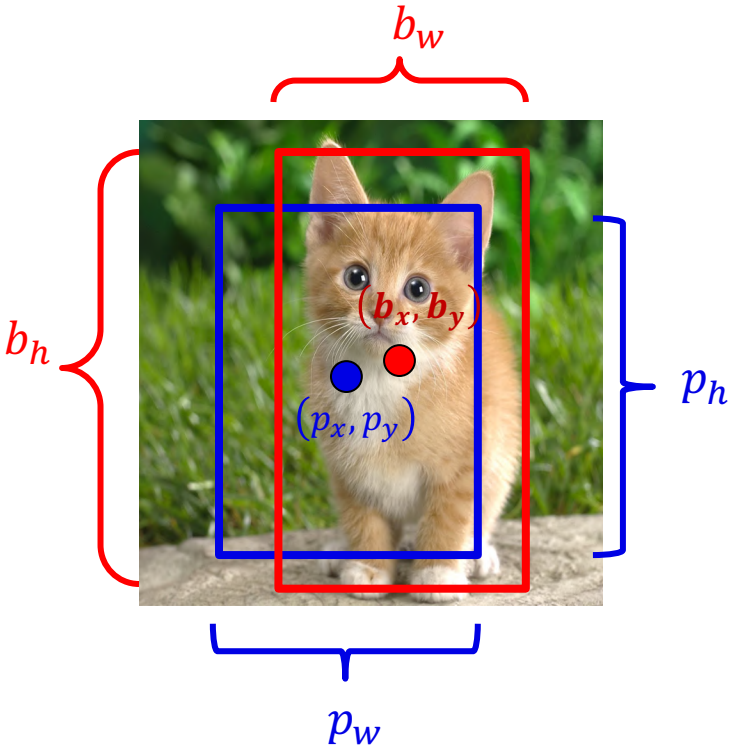
R-CNN: Box Regression



- Consider a Region Proposal with center (p_x, p_y) and size (p_h, p_w)
- The model predicts a transform to correct the region proposal: (t_x, t_y, t_w, t_h)

Source: Justin Johnson

R-CNN: Box Regression



- Consider a Region Proposal with center (p_x, p_y) and size (p_h, p_w)
- The model predicts a transform to correct the region proposal: (t_x, t_y, t_w, t_h)
- The output box is defined by:

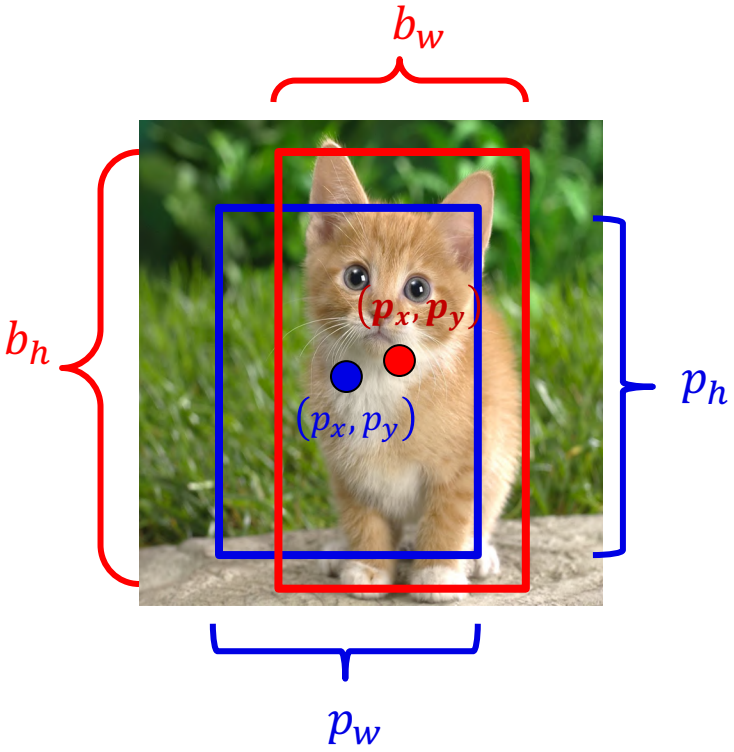
$$b_x = p_x + p_w * t_x$$
$$b_y = p_y + p_h * t_y$$
$$b_w = p_w * \exp(t_w)$$
$$b_h = p_h * \exp(t_h)$$

Shift center by amount relative to the RP size

Scale RP ; exp ensures that scaling factor > 0

Source: Justin Johnson

R-CNN: Box Regression



- Consider a Region Proposal with center (p_x, p_y) and size (p_h, p_w)
- The model predicts a transform to correct the region proposal: (t_x, t_y, t_w, t_h)
- The output box is defined by:

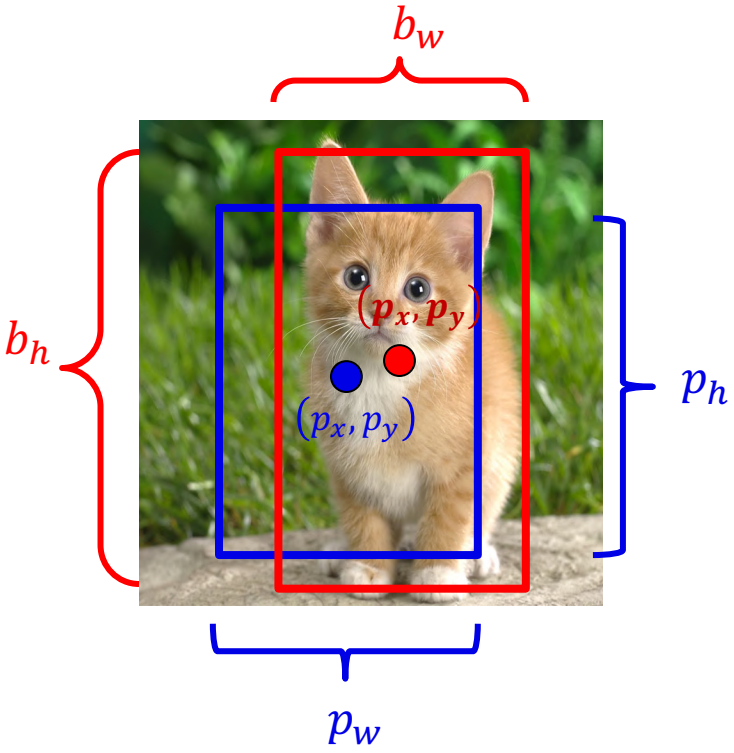
$$b_x = p_x + p_w * t_x$$
$$b_y = p_y + p_h * t_y$$
$$b_w = p_w * \exp(t_w)$$
$$b_h = p_h * \exp(t_h)$$

When transform=0 output=proposal

L2 regularization encourages leaving the proposal unchanged

Source: Justin Johnson

R-CNN: Box Regression



- Consider a Region Proposal with center (p_x, p_y) and size (p_h, p_w)
- The model predicts a transform to correct the region proposal: (t_x, t_y, t_w, t_h)
- The output box is defined by:

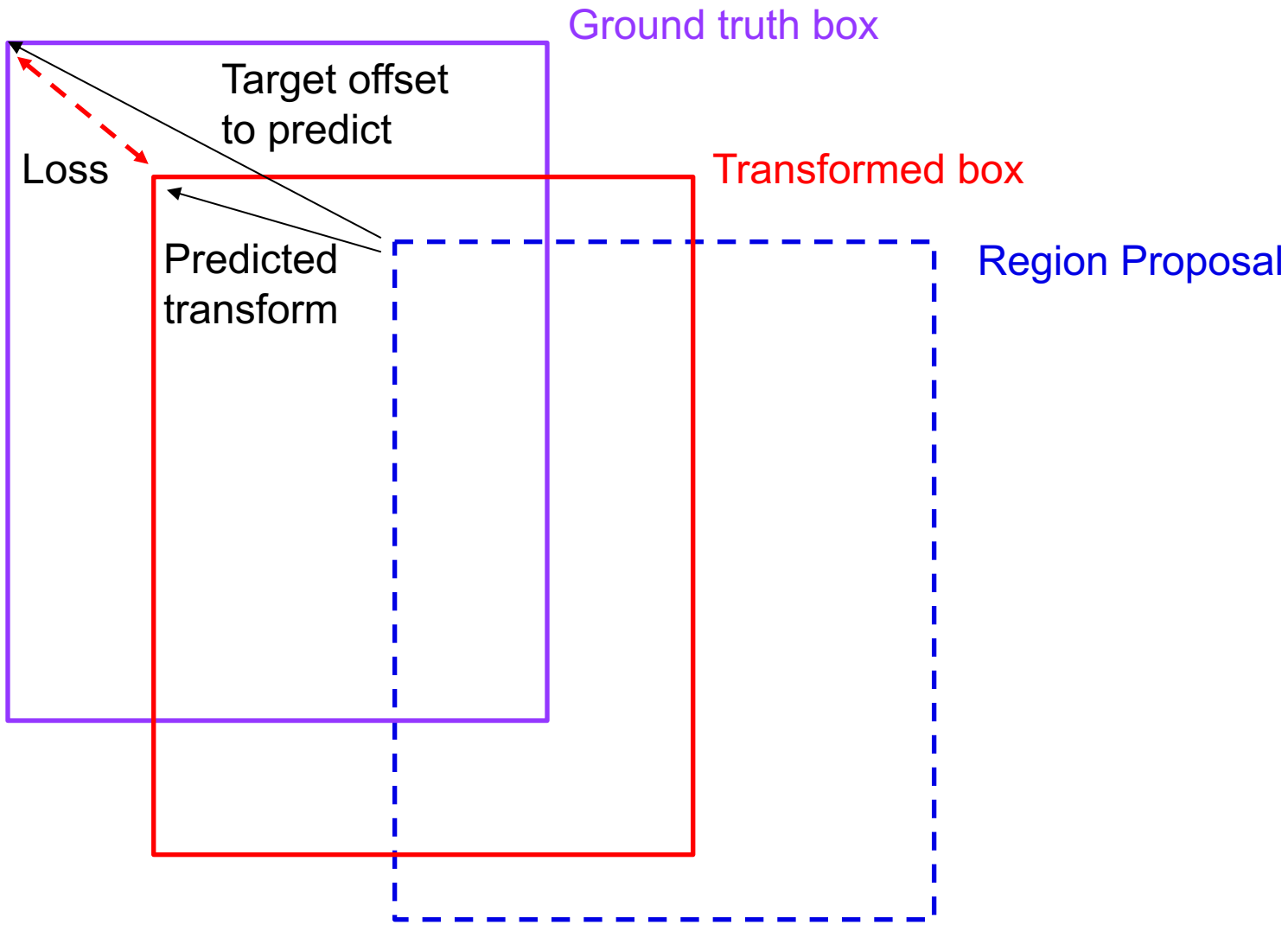
$$b_x = p_x + p_w * t_x$$
$$b_y = p_y + p_h * t_y$$
$$b_w = p_w * \exp(t_w)$$
$$b_h = p_h * \exp(t_h)$$

The transform encodes *relative* difference between proposal and the output;

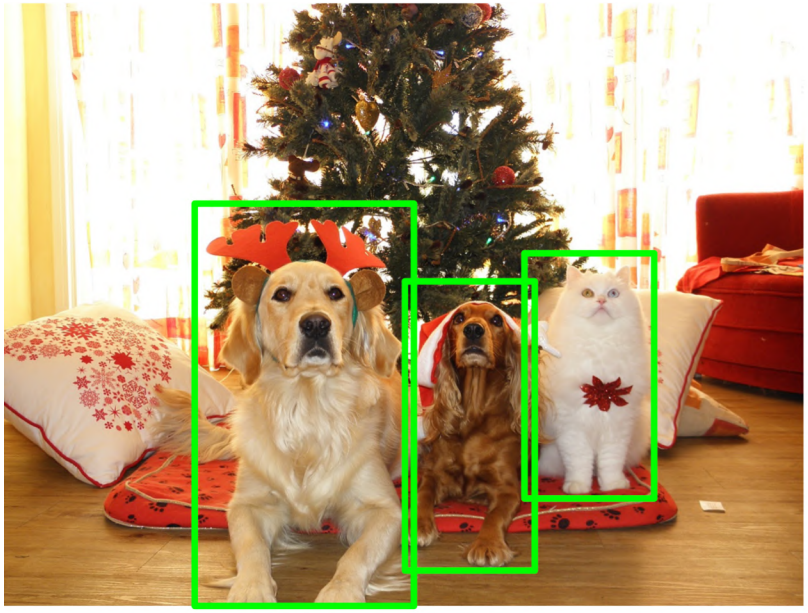
Note, CNN doesn't see absolute size or position after cropping and warping

Source: Justin Johnson

Bounding box regression Loss



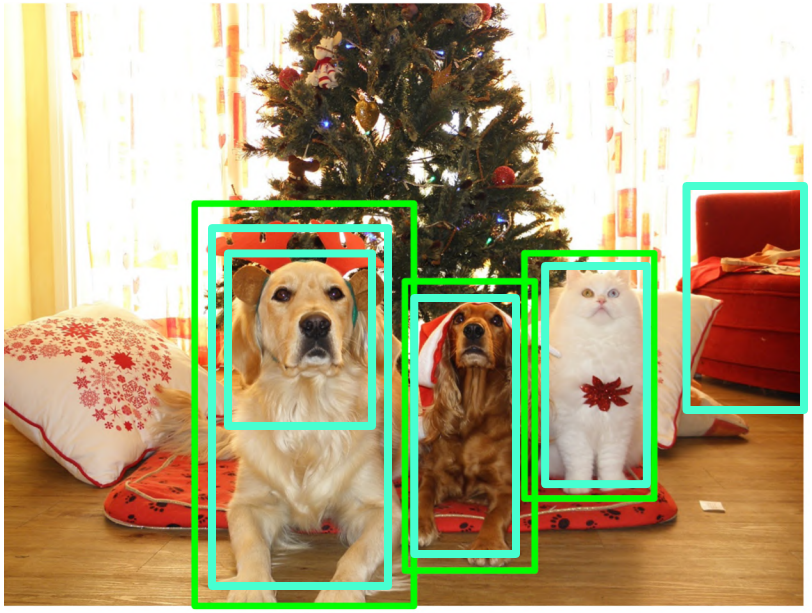
R-CNN Training



GT Boxes

Source: Justin Johnson

R-CNN Training

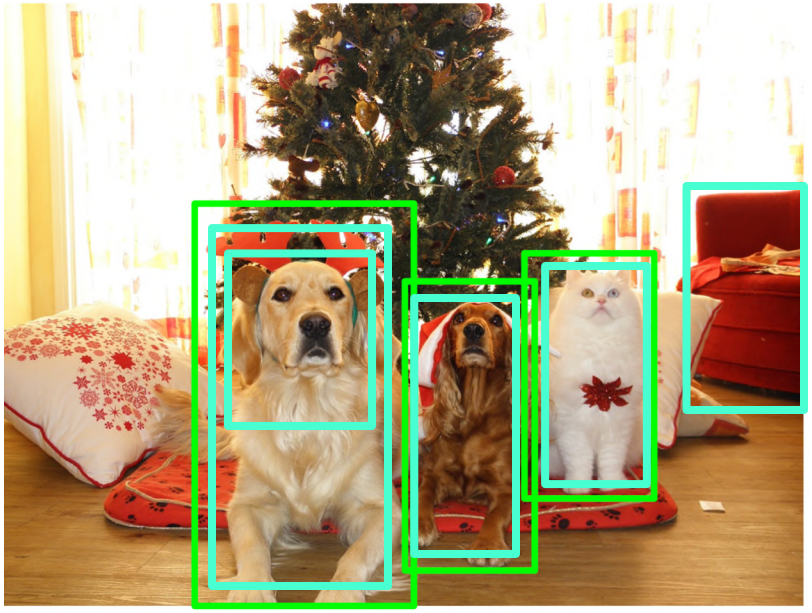


GT Boxes

Region Proposals

Source: Justin Johnson

R-CNN Training



GT Boxes

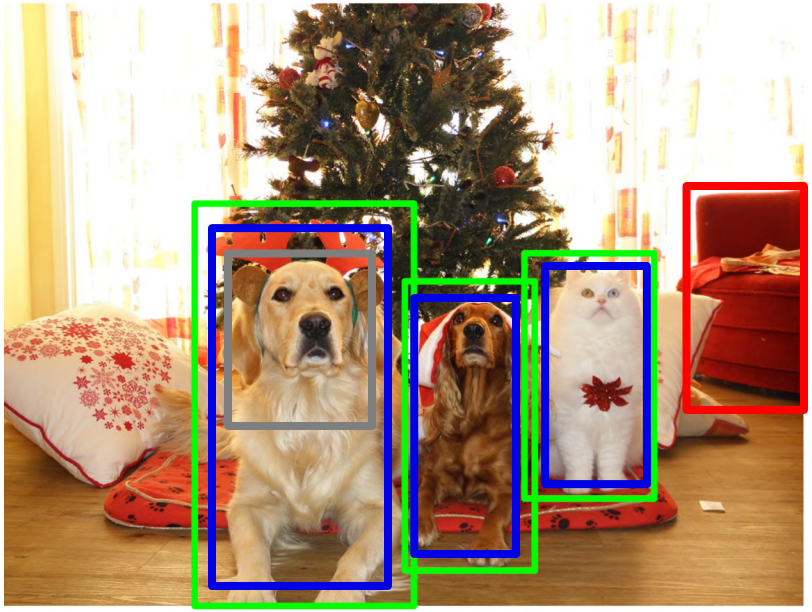
Region Proposals

Categorize each RP as:

- Positive: IOU > 0.5 with GT
- Negative: IOU < 0.3 with GT
- Neutral: Otherwise

Source: Justin Johnson

R-CNN Training



GT Boxes

Positive

Negative

Neutral

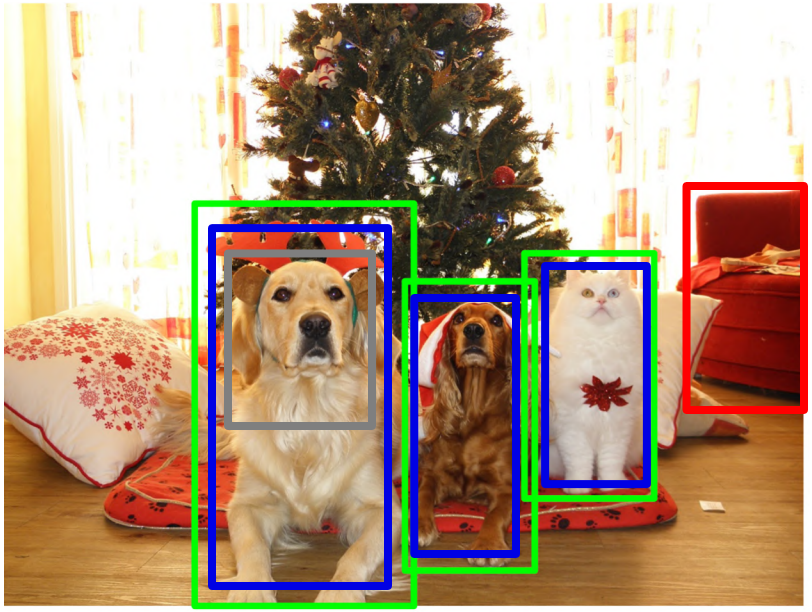
Categorize each RP as:

- Positive: IOU > 0.5 with GT
- Negative: IOU < 0.3 with GT
- Neutral: Otherwise

Source: Justin Johnson

R-CNN Training

Crop pixels from each positive and negative proposals and resize to 224x224:



GT Boxes

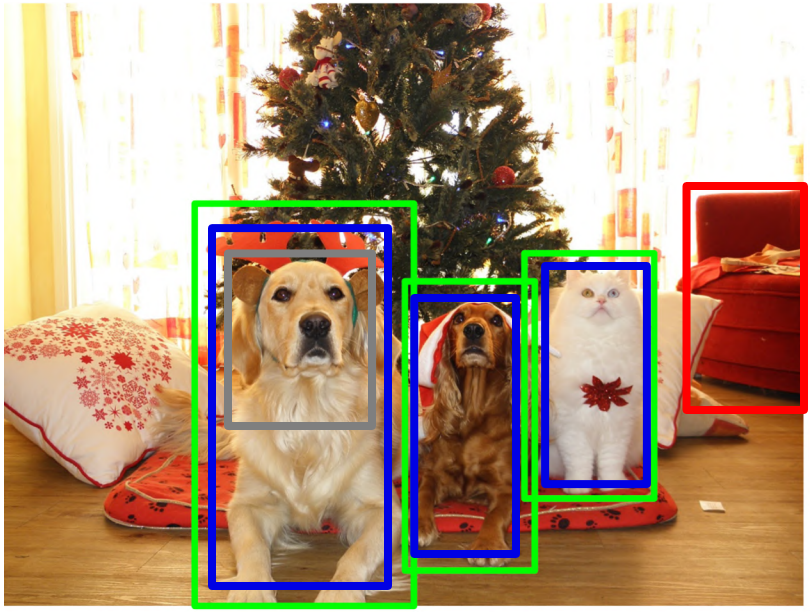
Positive

Negative

Neutral

Source: Justin Johnson

R-CNN Training



GT Boxes

Positive

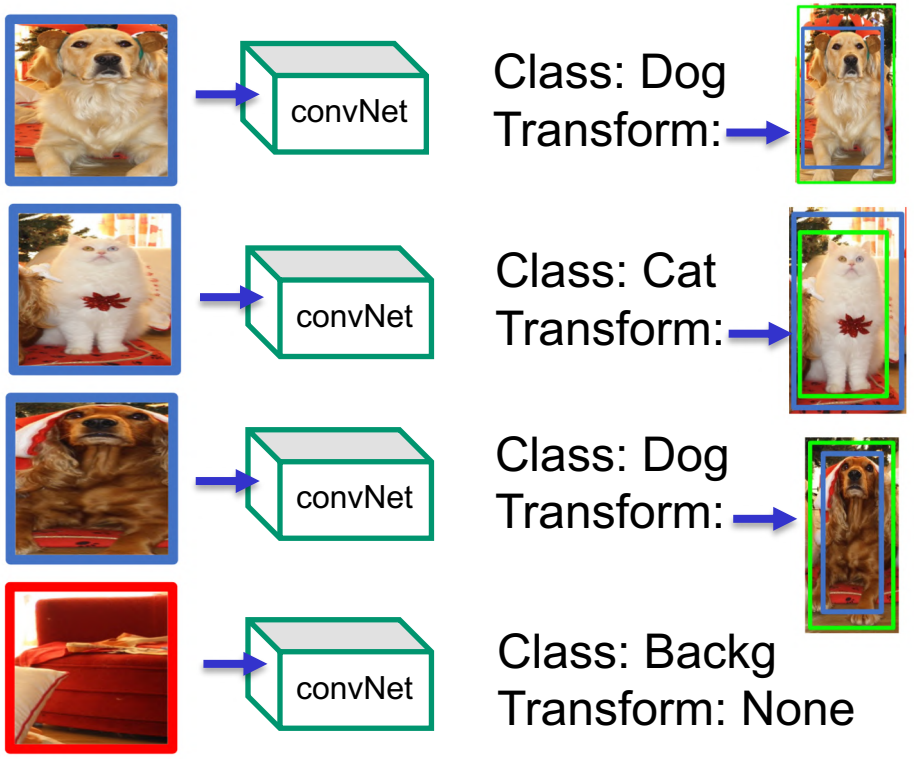
Negative

Neutral

Run each crop through CNN

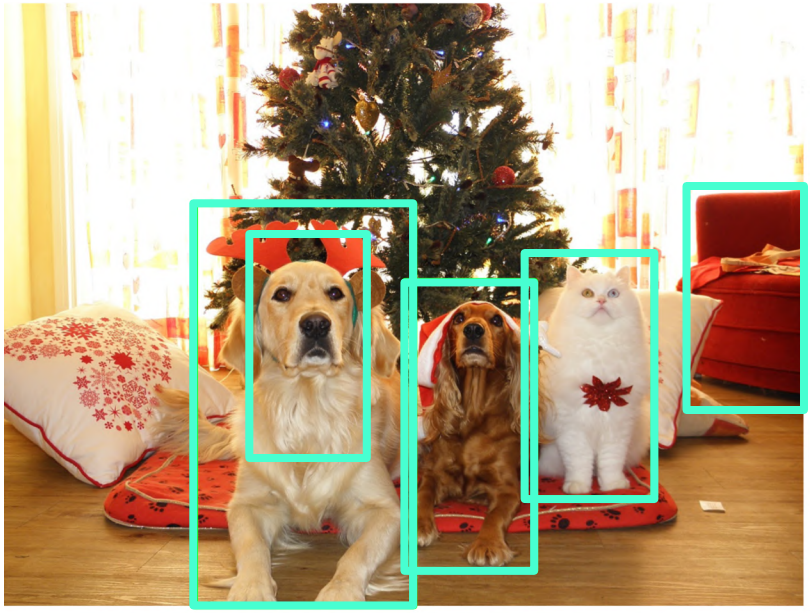
Positive regions: predicts class + transform

Negative regions: predict only class (backg)



Source: Justin Johnson

R-CNN Test-Time



Input: an image

1. Run proposal method ⟹ 2000 RPs.
2. Run CNN on each RP to get Class scores and Transform.
3. Threshold class scores to get a set of detections.

Problems:

- CNN often outputs overlapping boxes
- How to set the Thresholds?

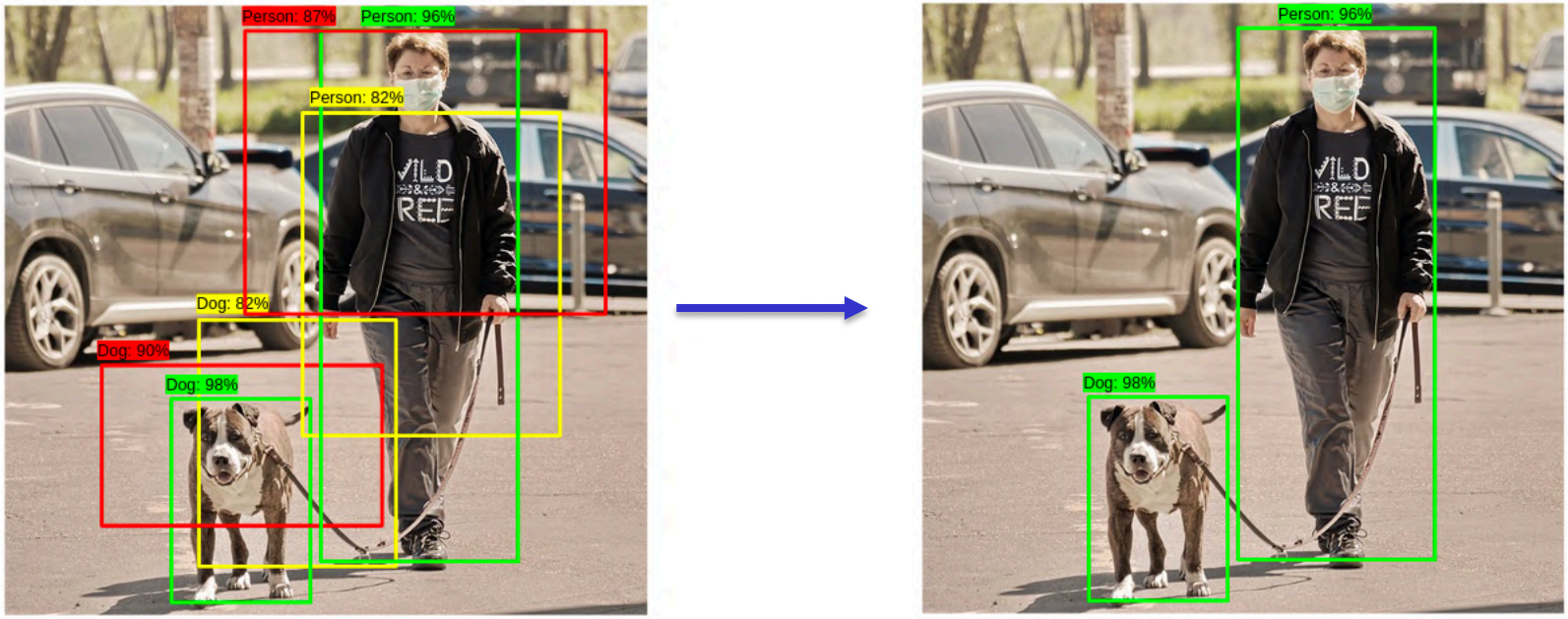
Source: Justin Johnson

Post Processing: Non-Max Supression

Non-max suppression:

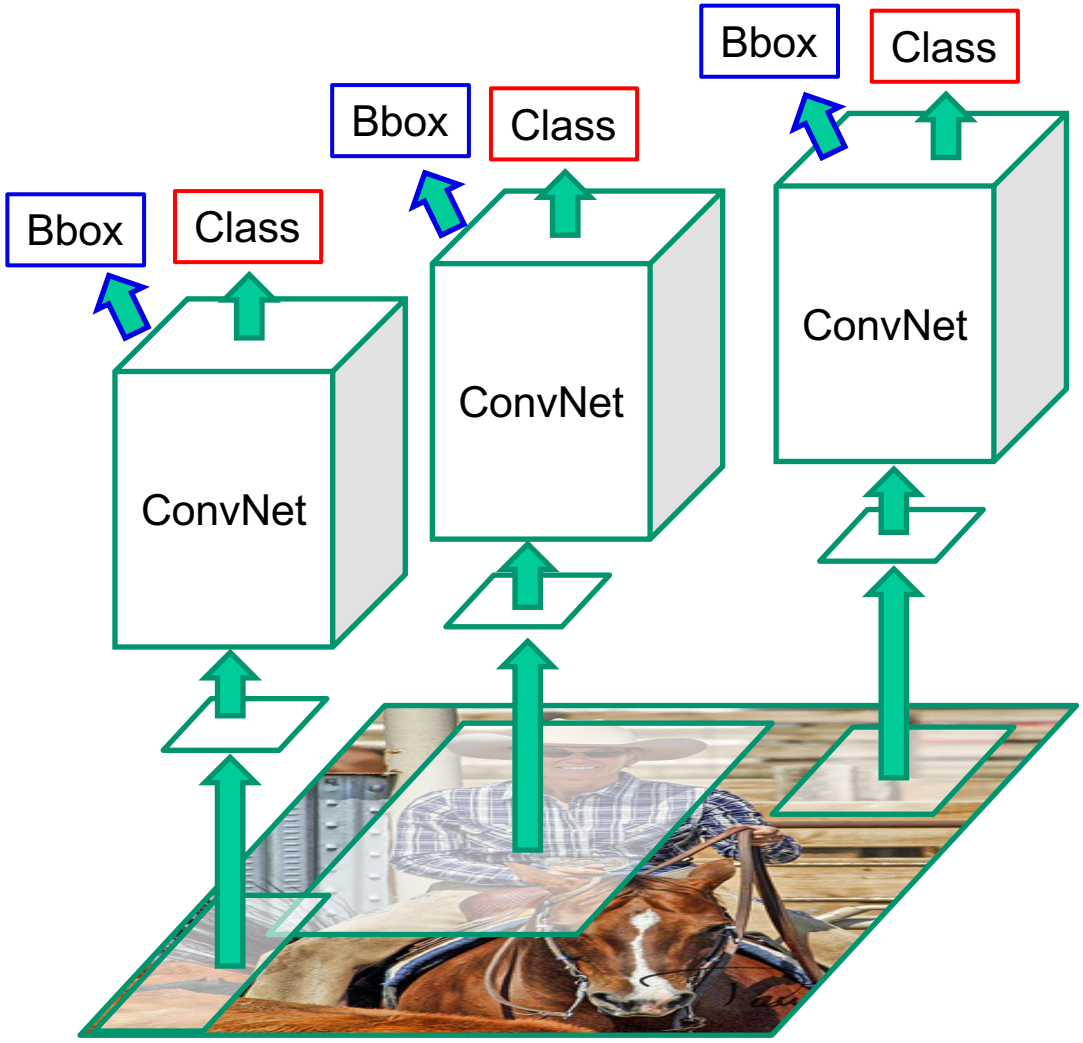
Sort the predictions by the confidence scores.

1. Select the next highest-score box,
2. Eliminate any current prediction if it has the same class and IoU > 0.5 with the current prediction.
3. Repeat step 1 until all predictions are checked.



Source: Justin Johnson

R-CNN: Problems



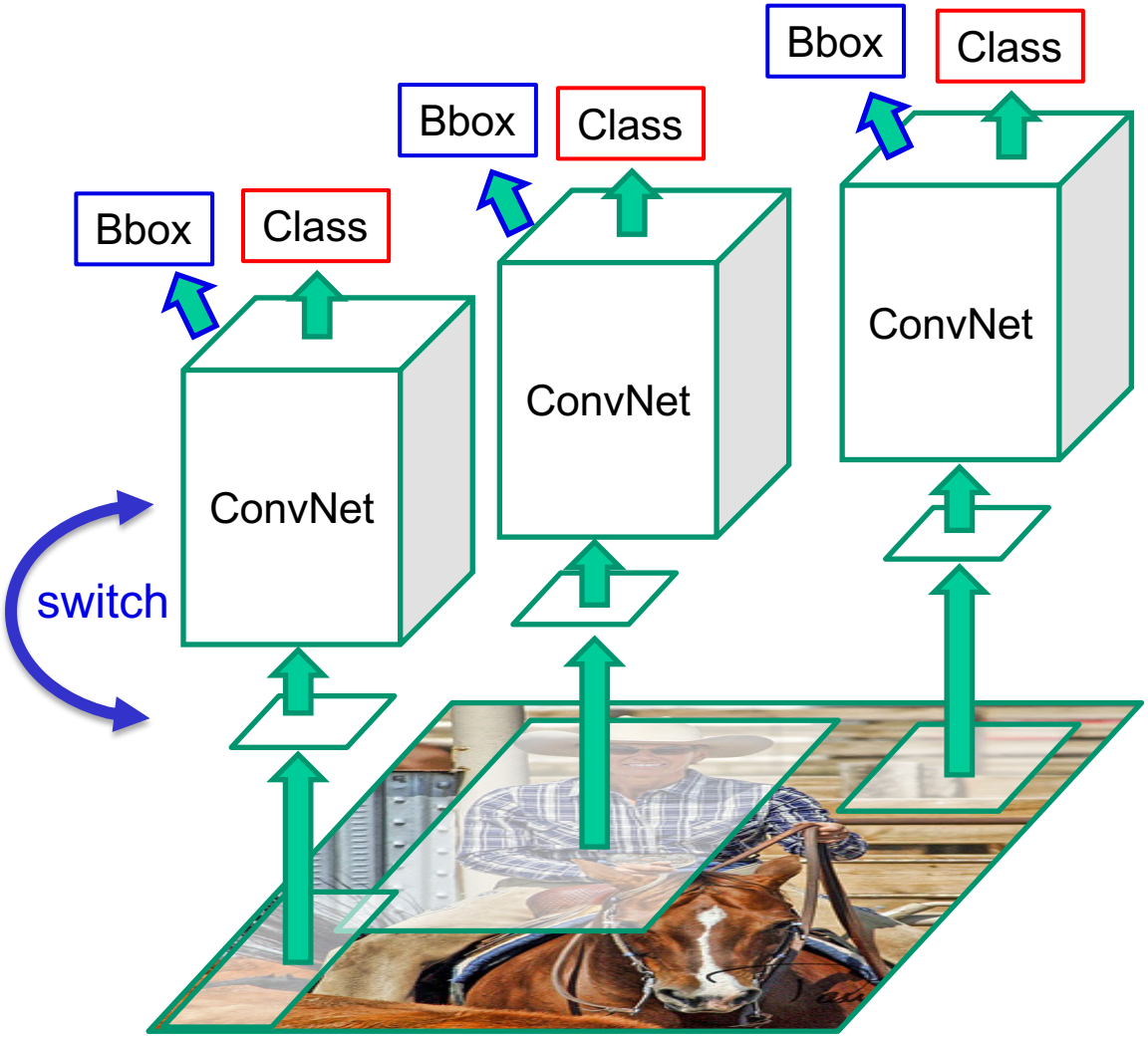
Problem: Very slow!

Need to run ~2000 forward passes for each image.

Waste calculations for overlapping RPs

Source: Justin Johnson

R-CNN: Problems



Problem: Very slow!

Need to run ~2000 forward passes for each image.

Waste calculations for overlapping RPs

Solution: Run CNN before cropping.

R. Girshick, Fast R-CNN, ICCV 2015

Source: Justin Johnson

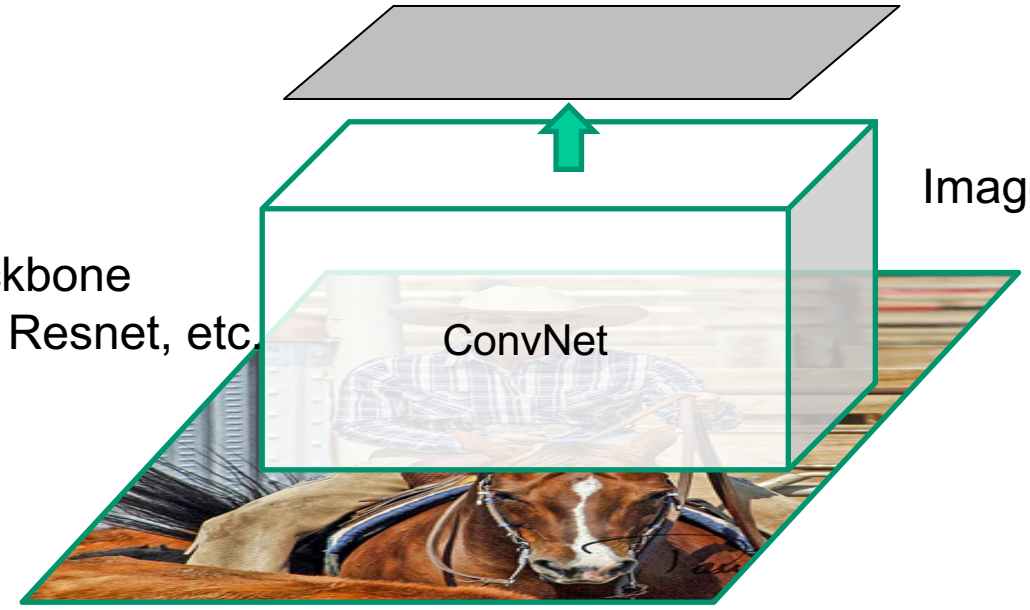
Fast R-CNN

Input image

Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN



Backbone
AlexNet / Resnet, etc.

Image features (512x20x15)

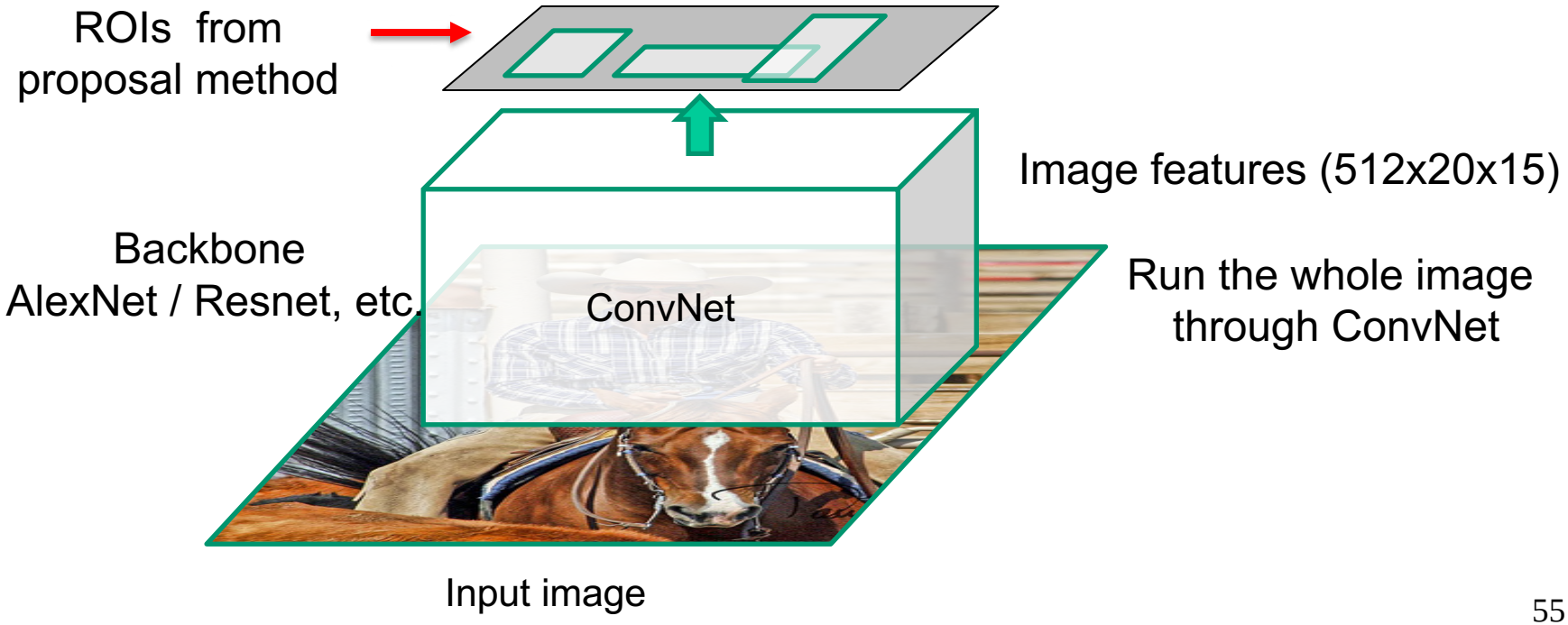
Run the whole image
through ConvNet

Input image

Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

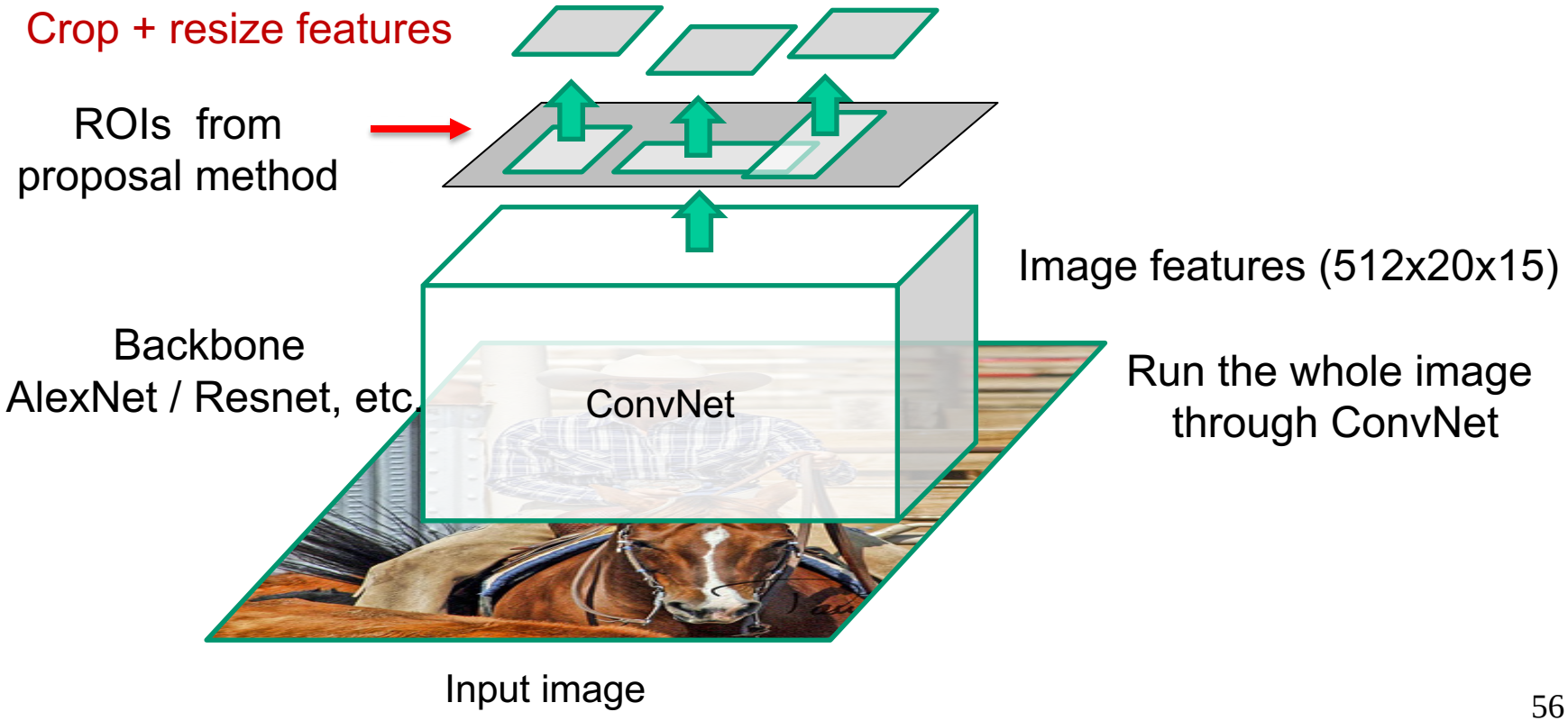
Fast R-CNN



Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

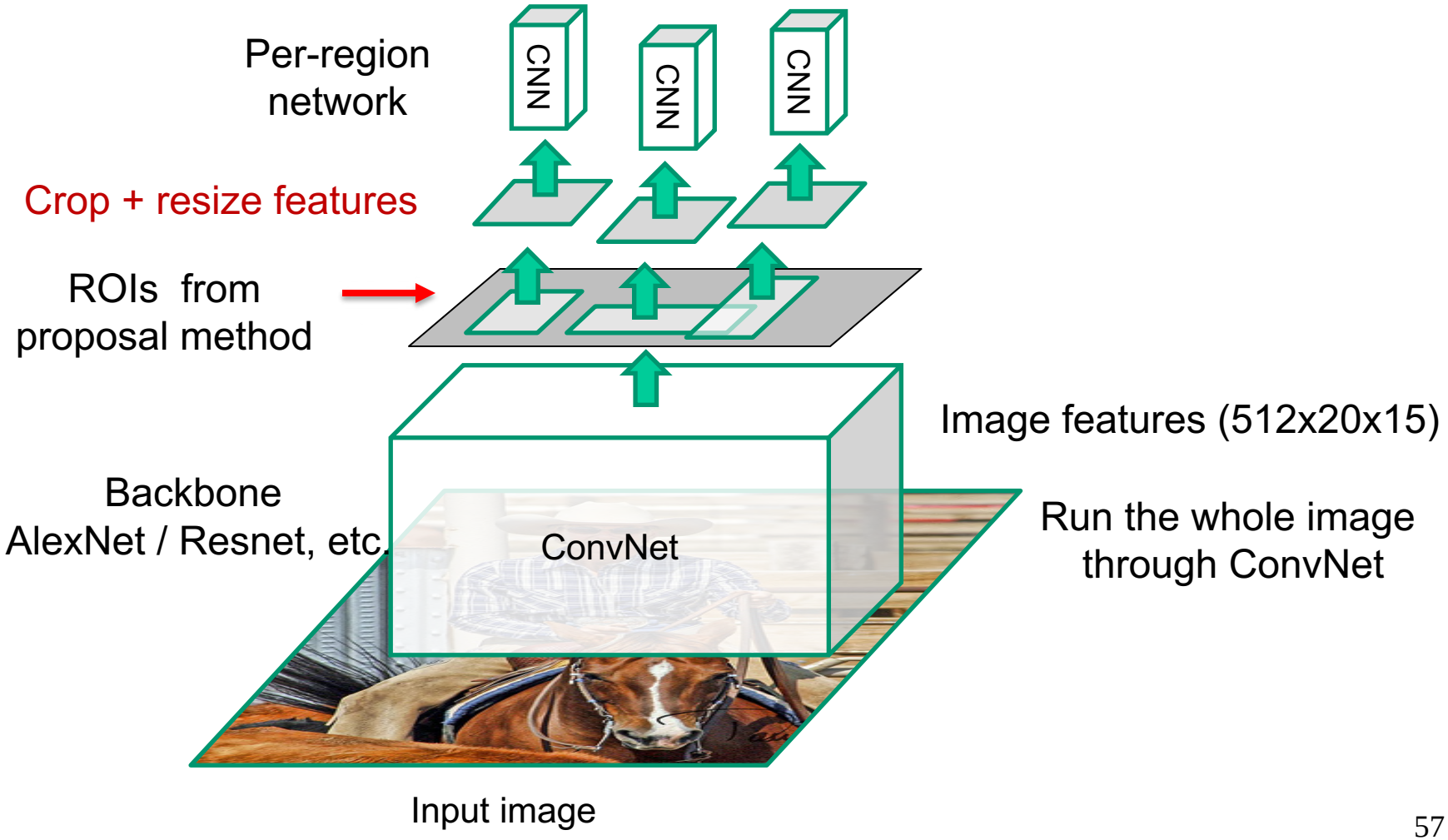
Fast R-CNN



Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

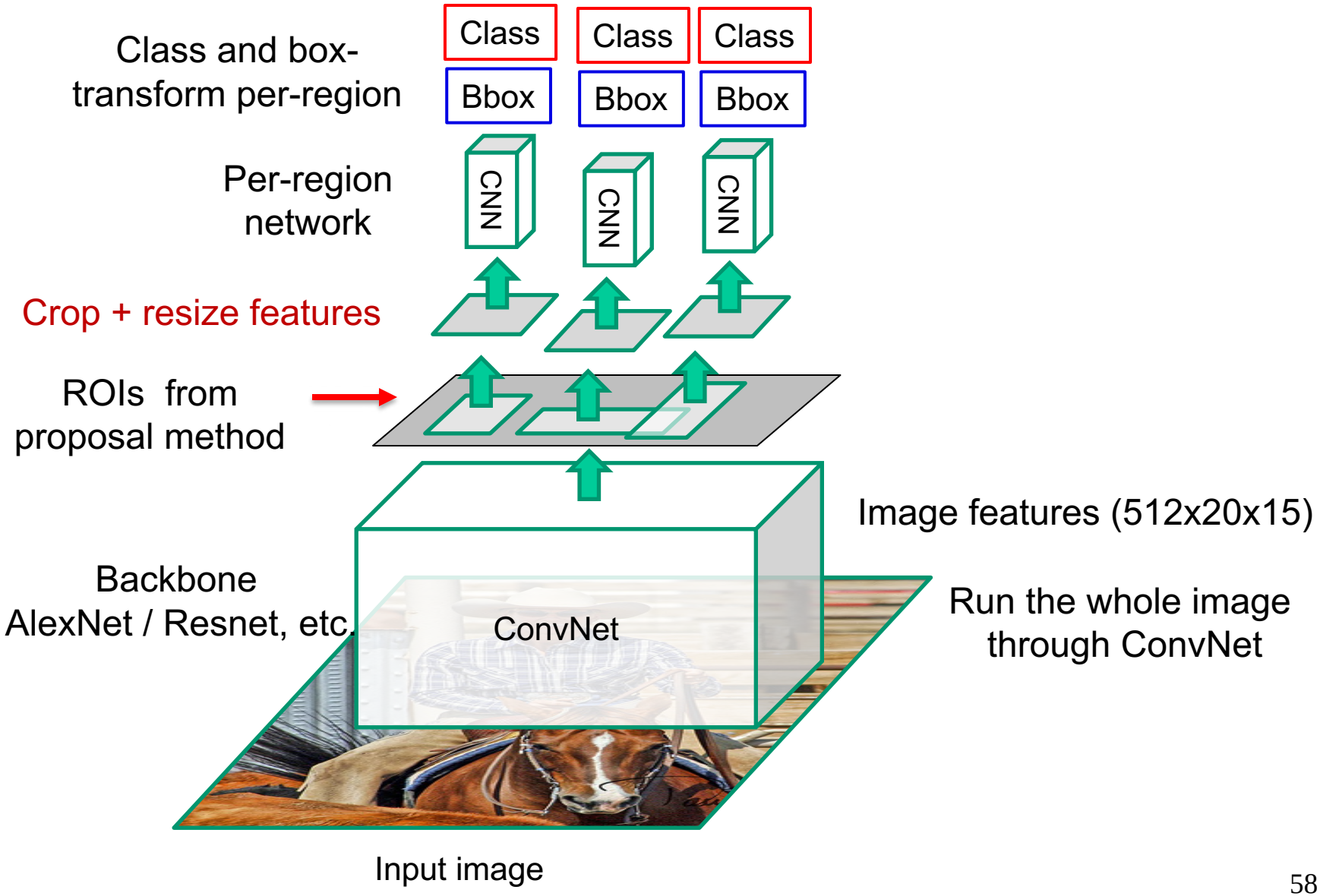
Fast R-CNN



Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

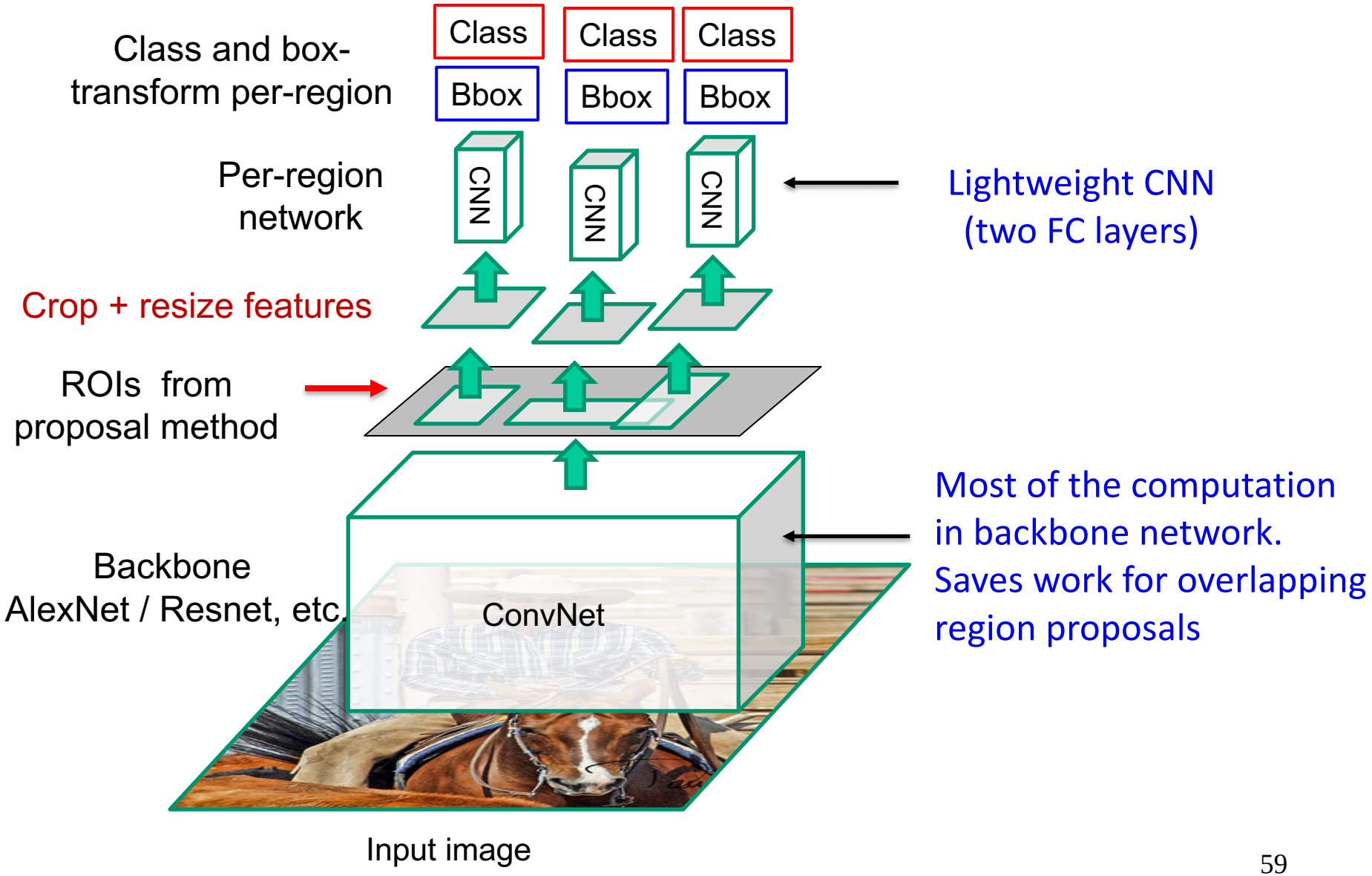
Fast R-CNN



Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

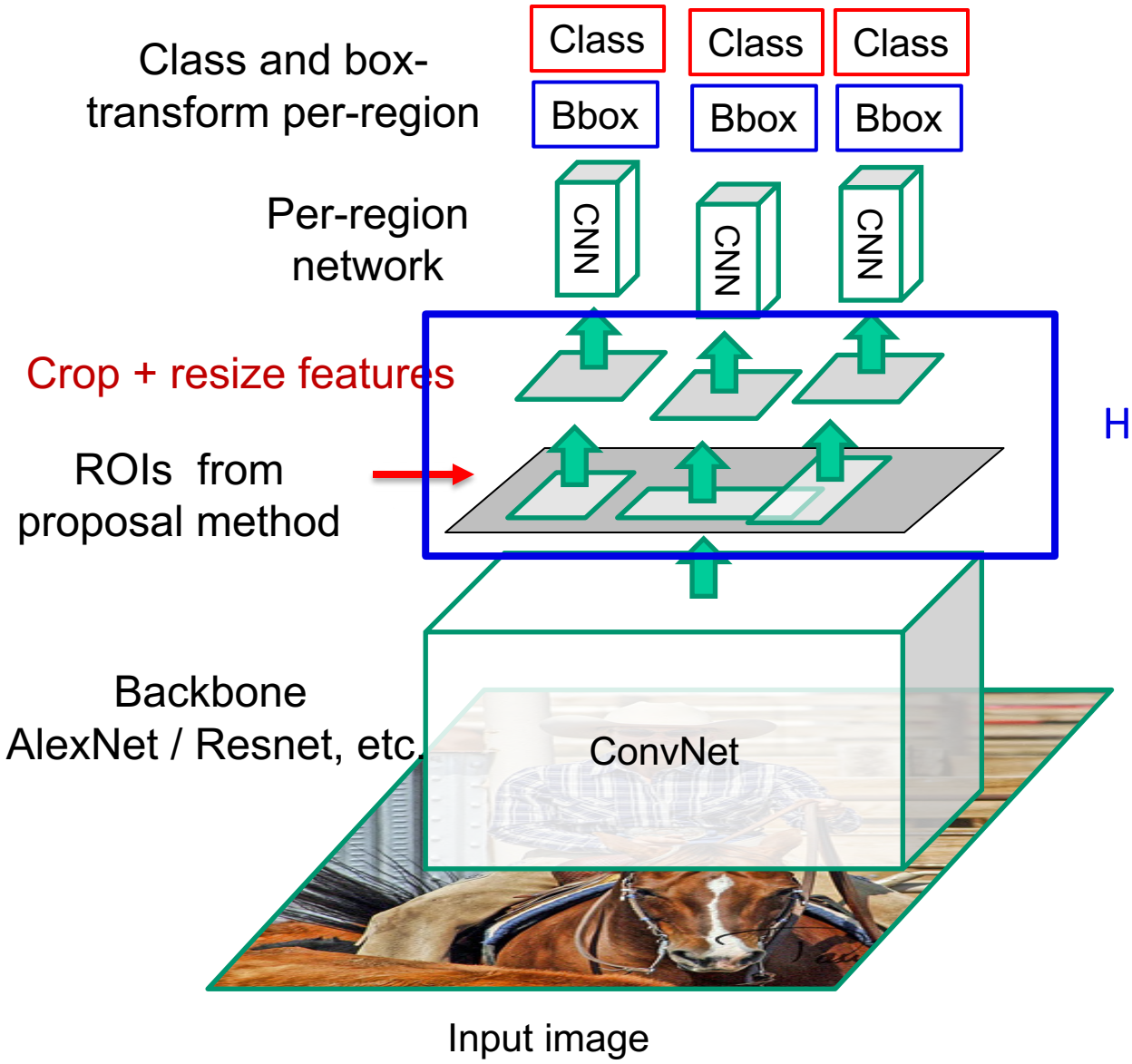
Fast R-CNN



Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN



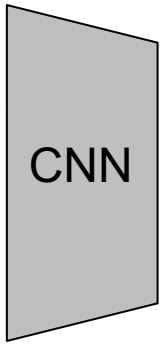
How to crop features?

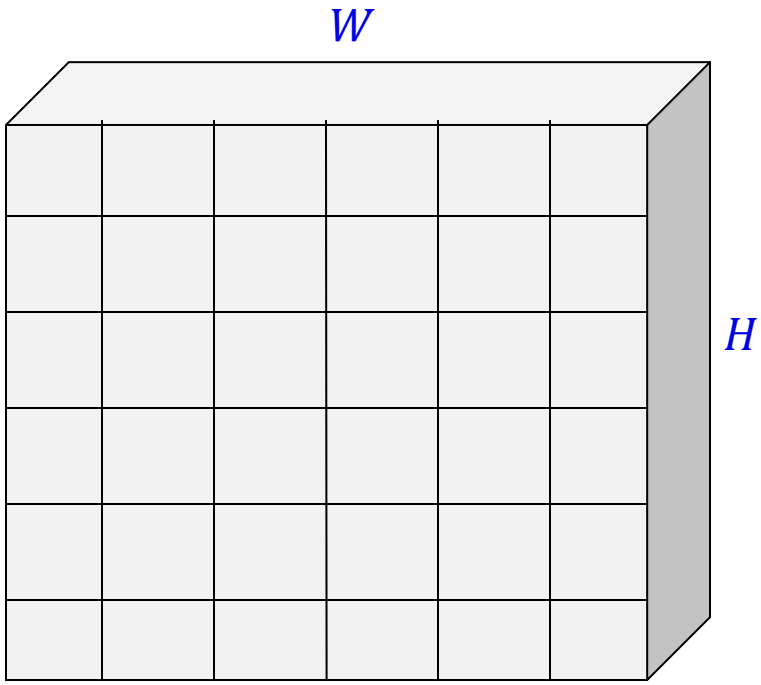
Source: Justin Johnson

R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN: Cropping using ROI Align

Input image
(e.g. 3x640x480)

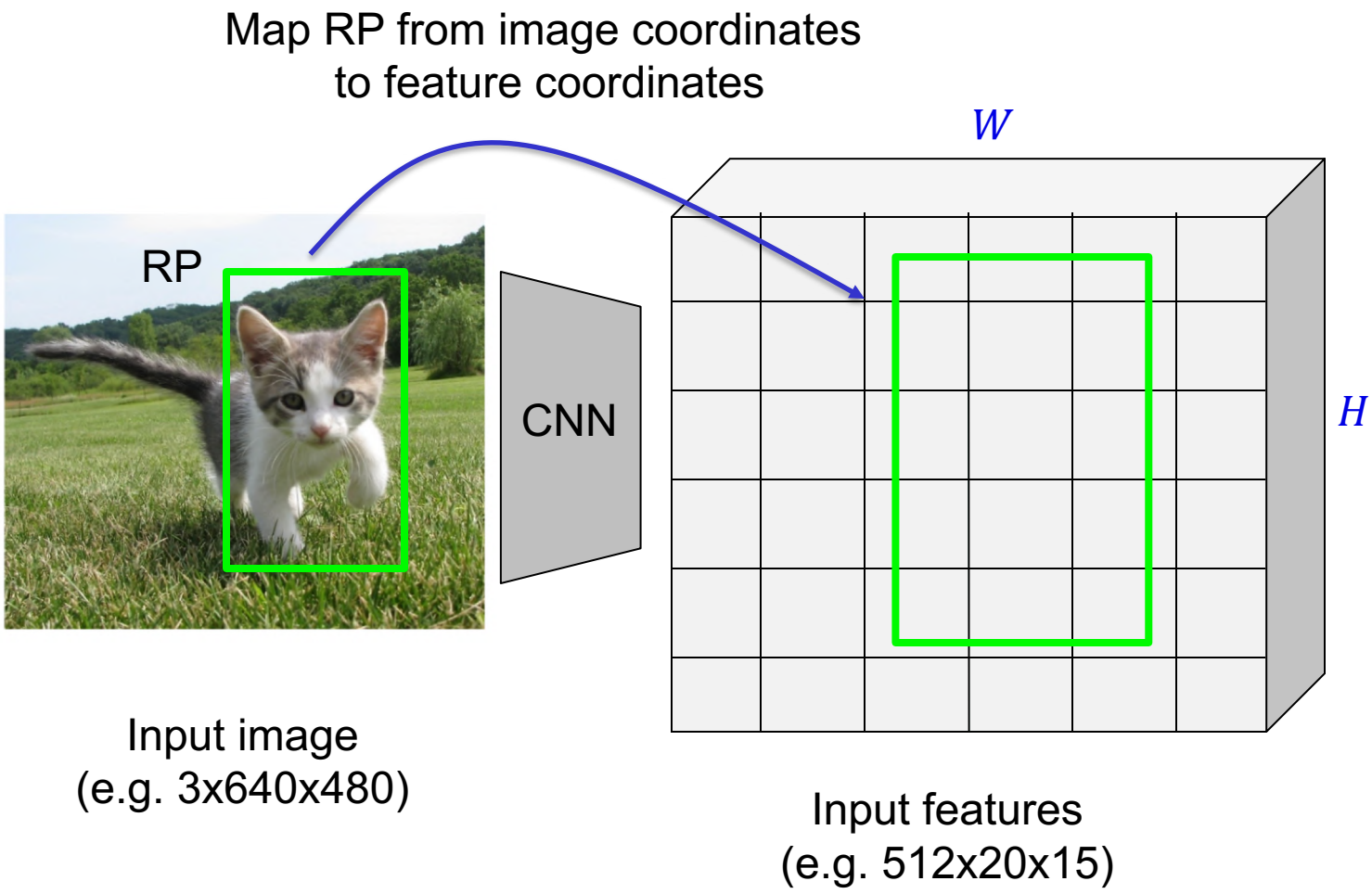




Input features
(e.g. 512x20x15)

Source: Justin Johnson

Fast R-CNN: Cropping using ROI Align

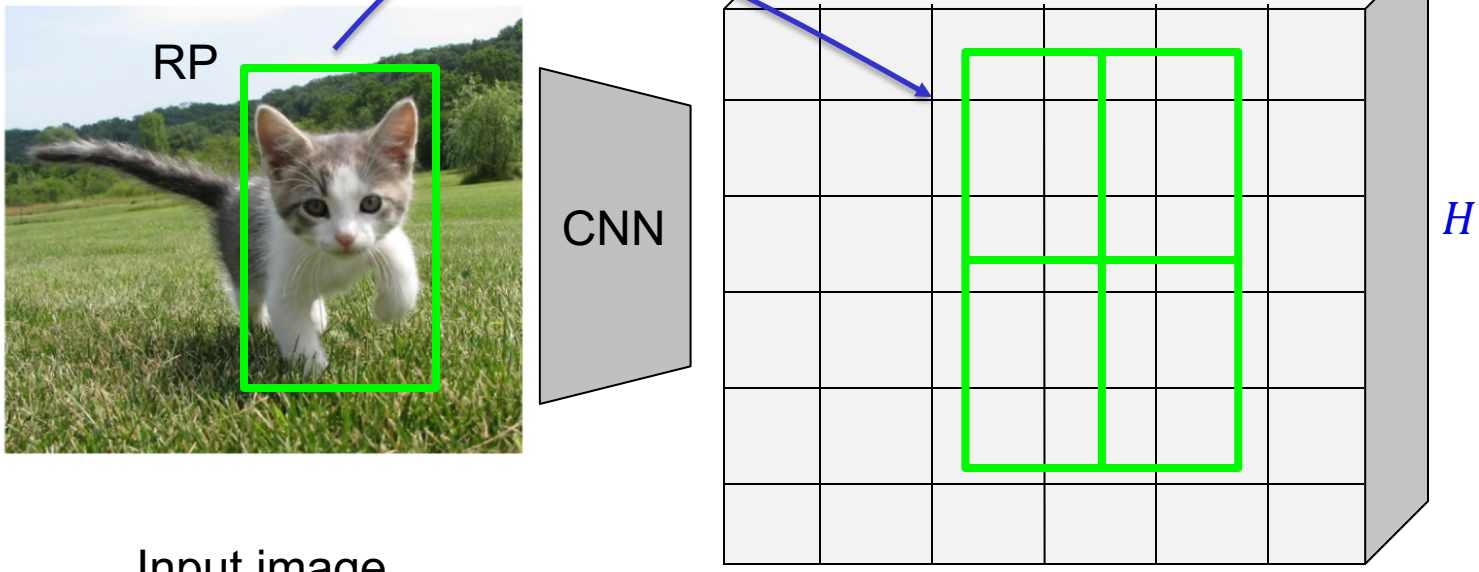


Source: Justin Johnson

Fast R-CNN: Cropping using ROI Align

Map RP from image coordinates to feature coordinates

Divide RP into kxk subregions (here 512x2x2, in practice 512x7x7)



Input image (e.g. 3x640x480)

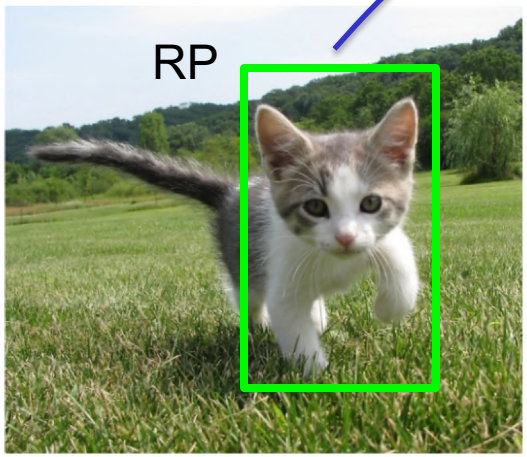
Input features (e.g. 512x20x15)

Source: Justin Johnson

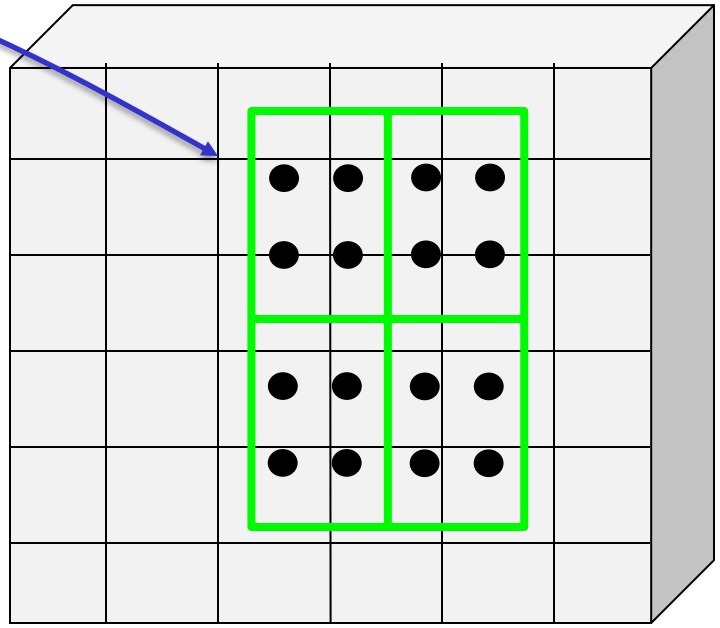
Fast R-CNN: Cropping using ROI Align

Map RP from image coordinates to feature coordinates

Divide RP into kxk subregions (here 512x2x2, in practice 512x7x7)



CNN



Sample each subregion with mxm points using **Bilinear Interpolation**

Input image (e.g. 3x640x480)

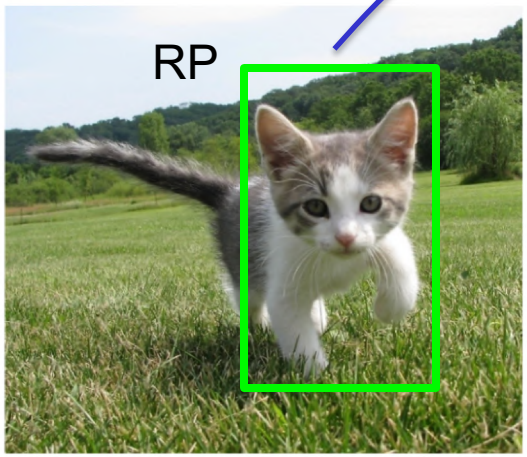
Input features (e.g. 512x20x15)

Source: Justin Johnson

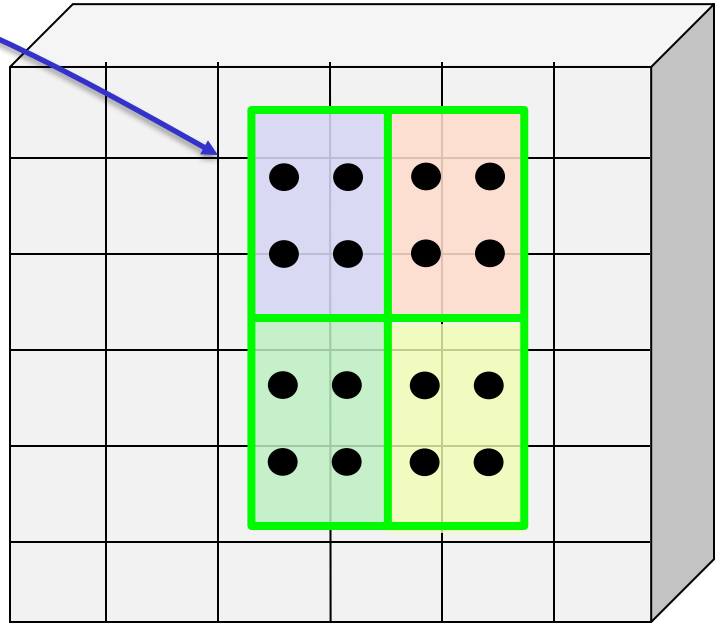
Fast R-CNN: Cropping using ROI Align

Map RP from image coordinates to feature coordinates

Divide RP into kxk subregions (here 512x2x2, in practice 512x7x7)



CNN



Sample each subregion with mxm points using **Bilinear Interpolation**

Max-pool within each subregion

Input image (e.g. 3x640x480)

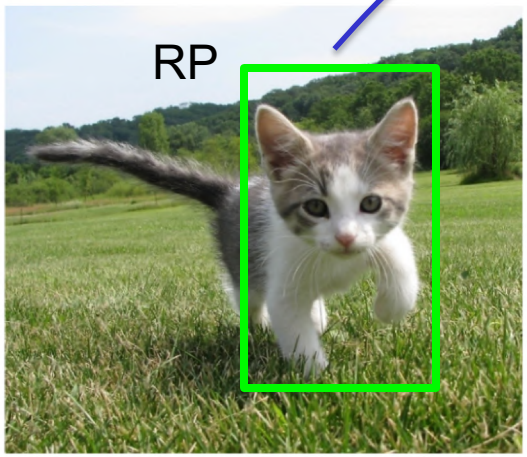
Input features (e.g. 512x20x15)

Source: Justin Johnson

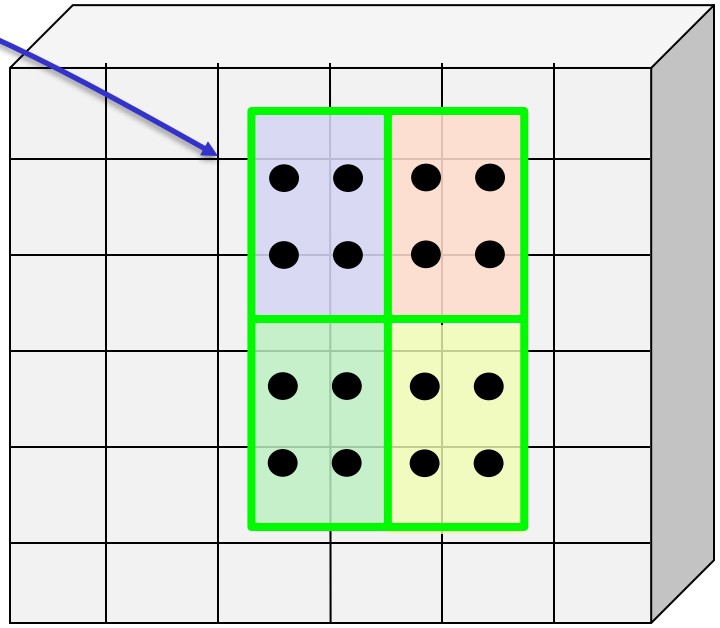
Fast R-CNN: Cropping using ROI Align

Map RP from image coordinates to feature coordinates

Divide RP into kxk subregions (here 512x2x2, in practice 512x7x7)



CNN

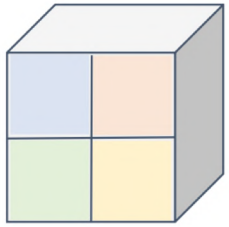


Sample each subregion with mxm points using **Bilinear Interpolation**

Input image (e.g. 3x640x480)

Input features (e.g. 512x20x15)

Max-pool within each subregion

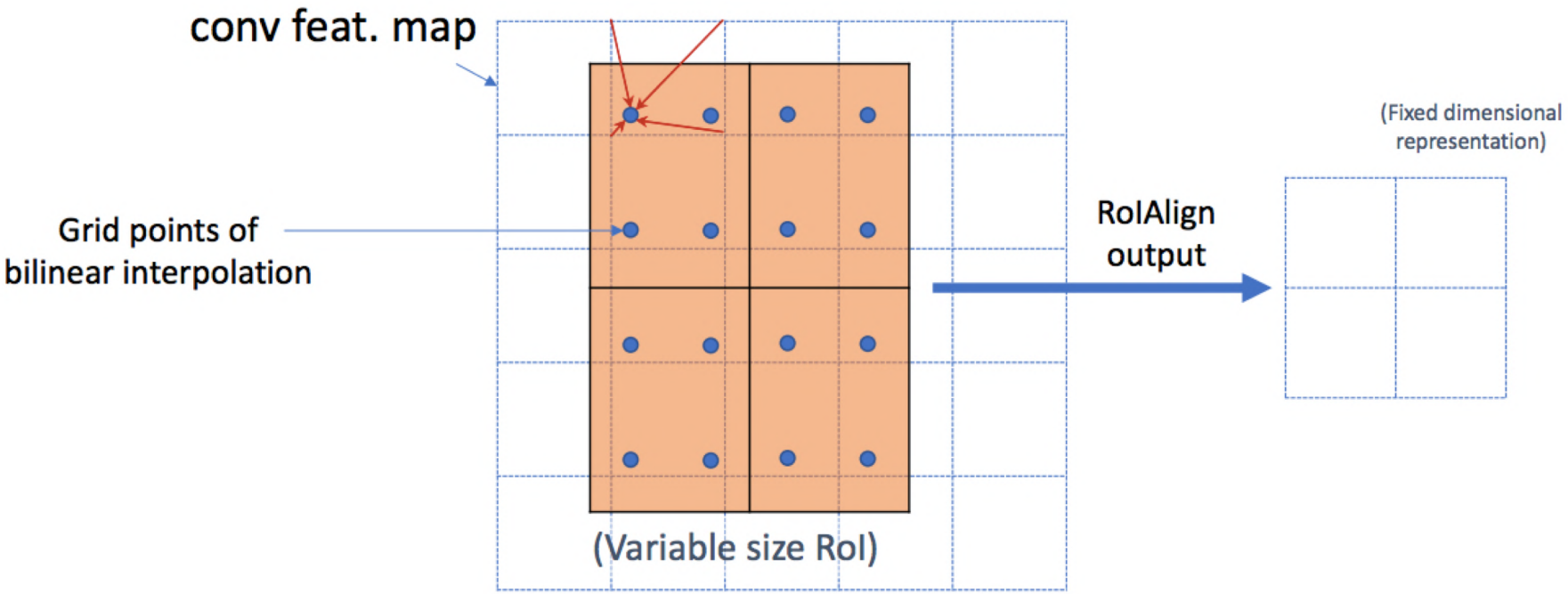


512x2x2

Source: Justin Johnson

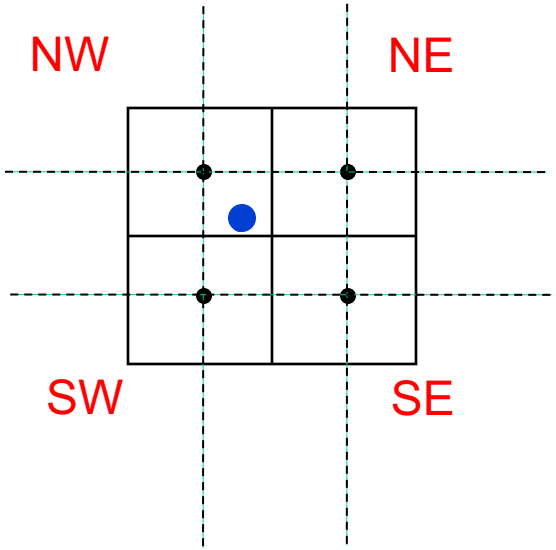
ROI Align

- Bilinear interpolation then Max-pooling

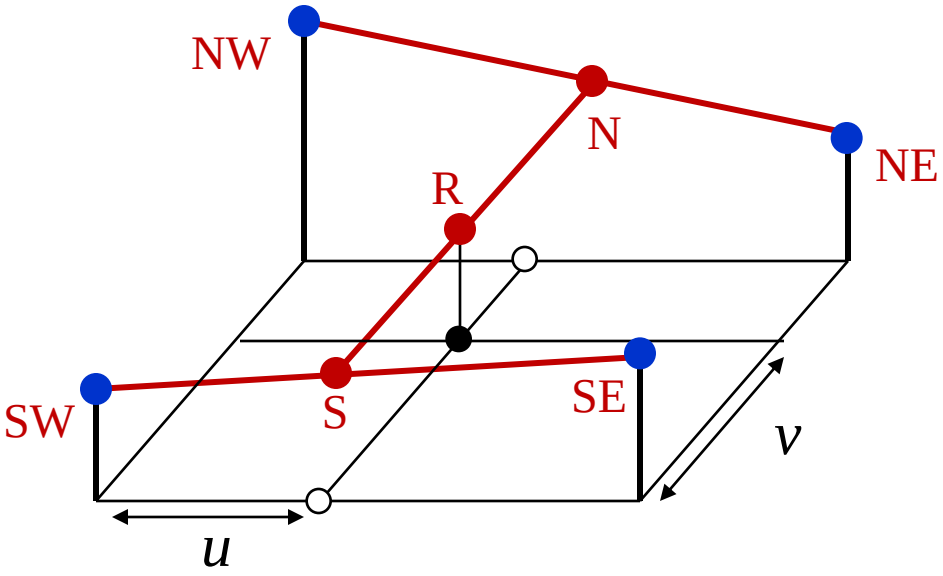


Bilinear interpolation

- Each sample has 4 nearest neighbor values.



Bilinear interpolation



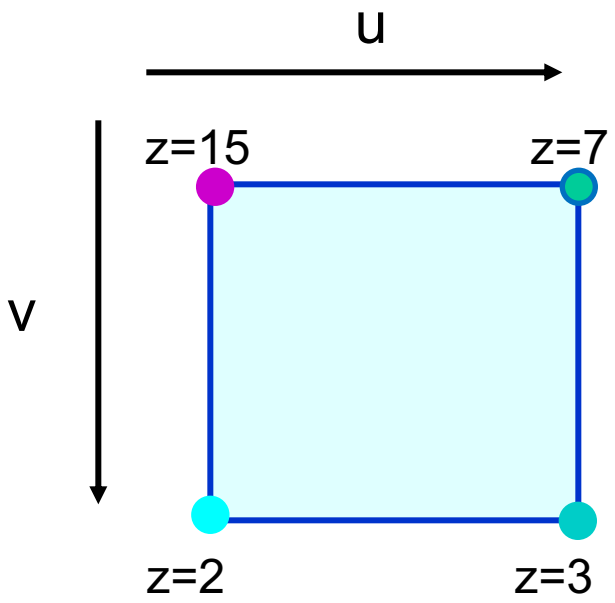
$$u, v \in [0,1]$$

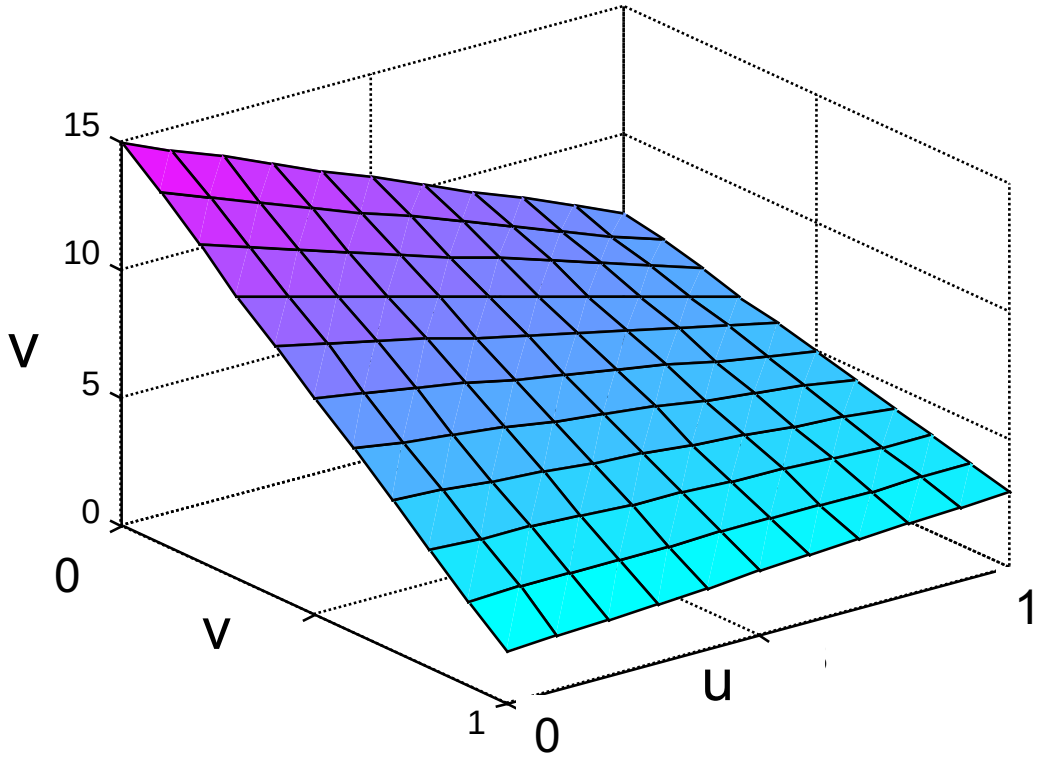
$$S(u) = u\,SE + (1-u)\,SW$$

$$N(u) = u\,NE + (1-u)\,NW$$

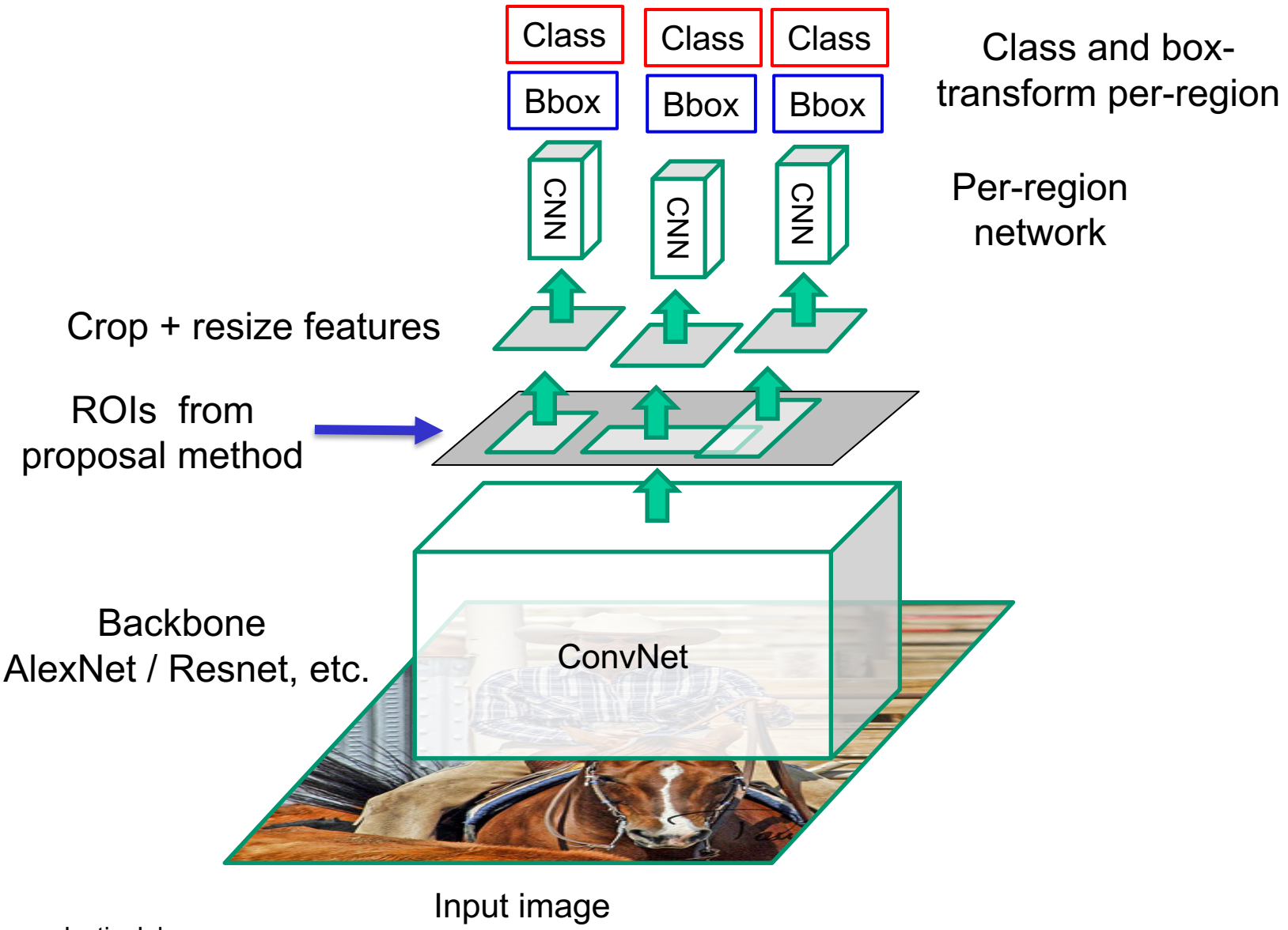
$$R(u,v) = v\,N(u) + (1-v)\,S(u)$$

Bilinear interpolation: Example



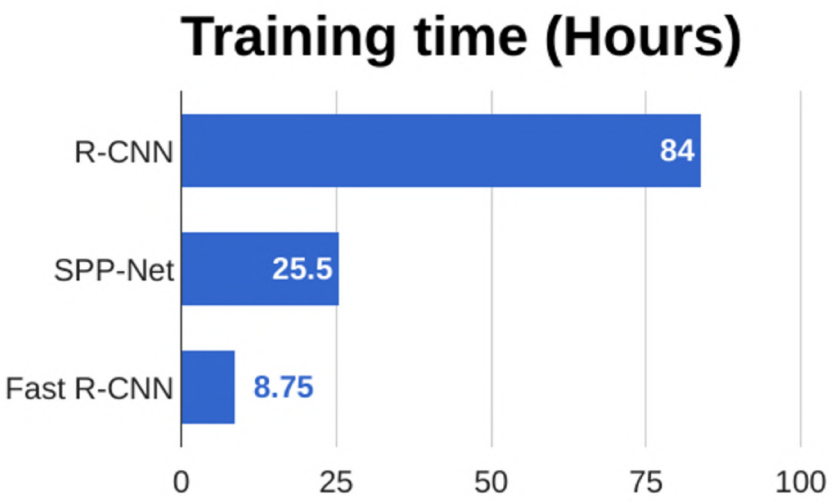


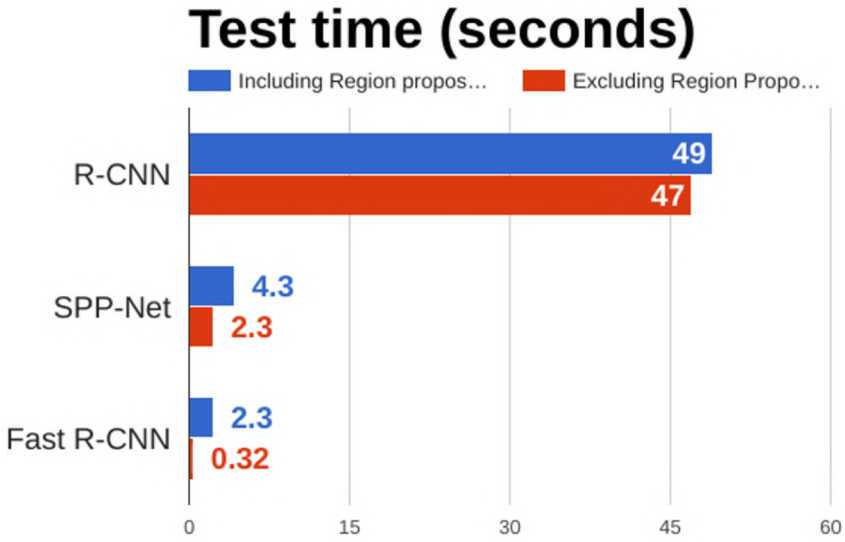
Fast R-CNN



Source: Justin Johnson

R-CNN vs. Fast R-CNN





What about the performance?

Source: Justin Johnson

Evaluating a detector

Test image (previously unseen)

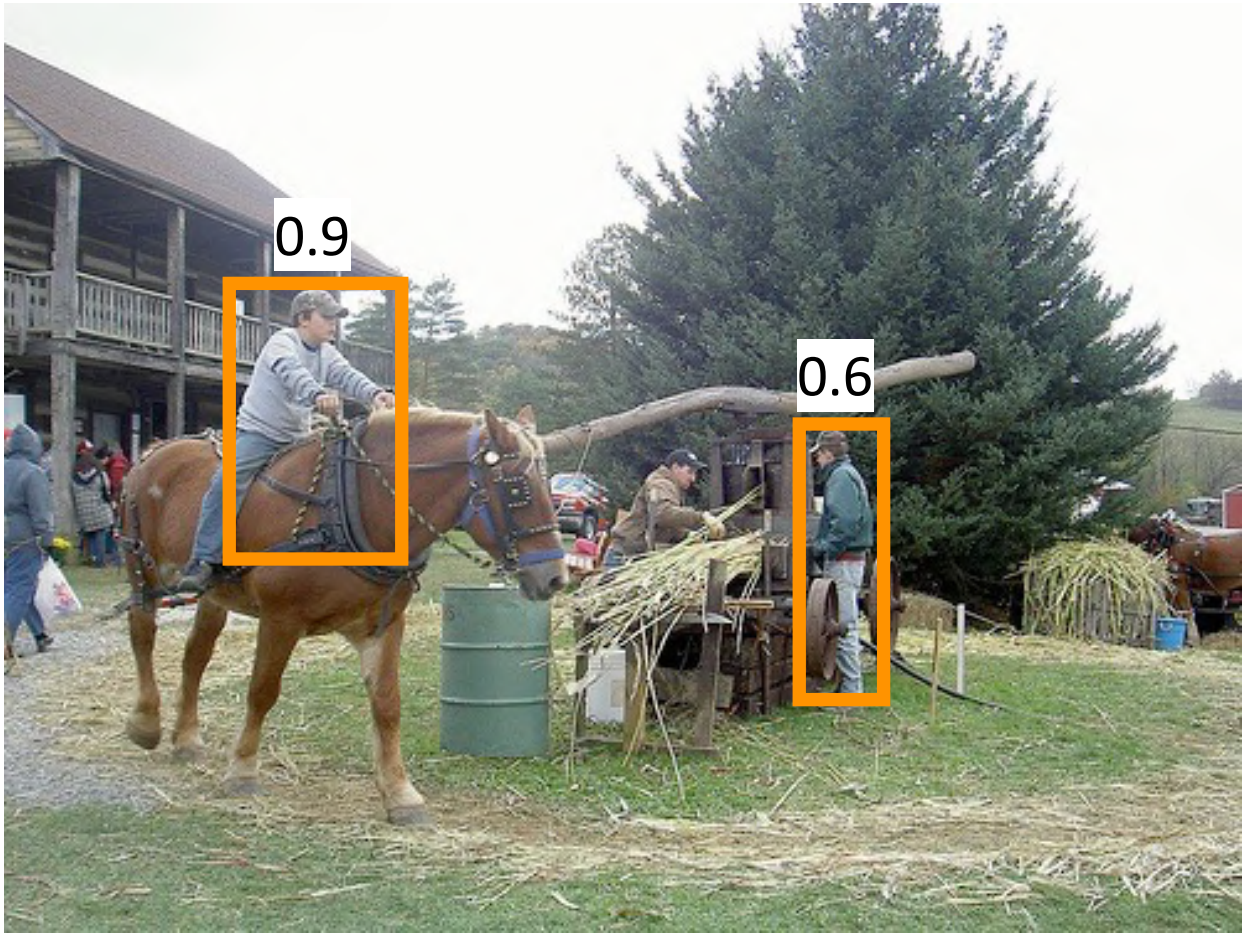
Evaluating a detector

First Detection

‘person’ detector predictions

Evaluating a detector

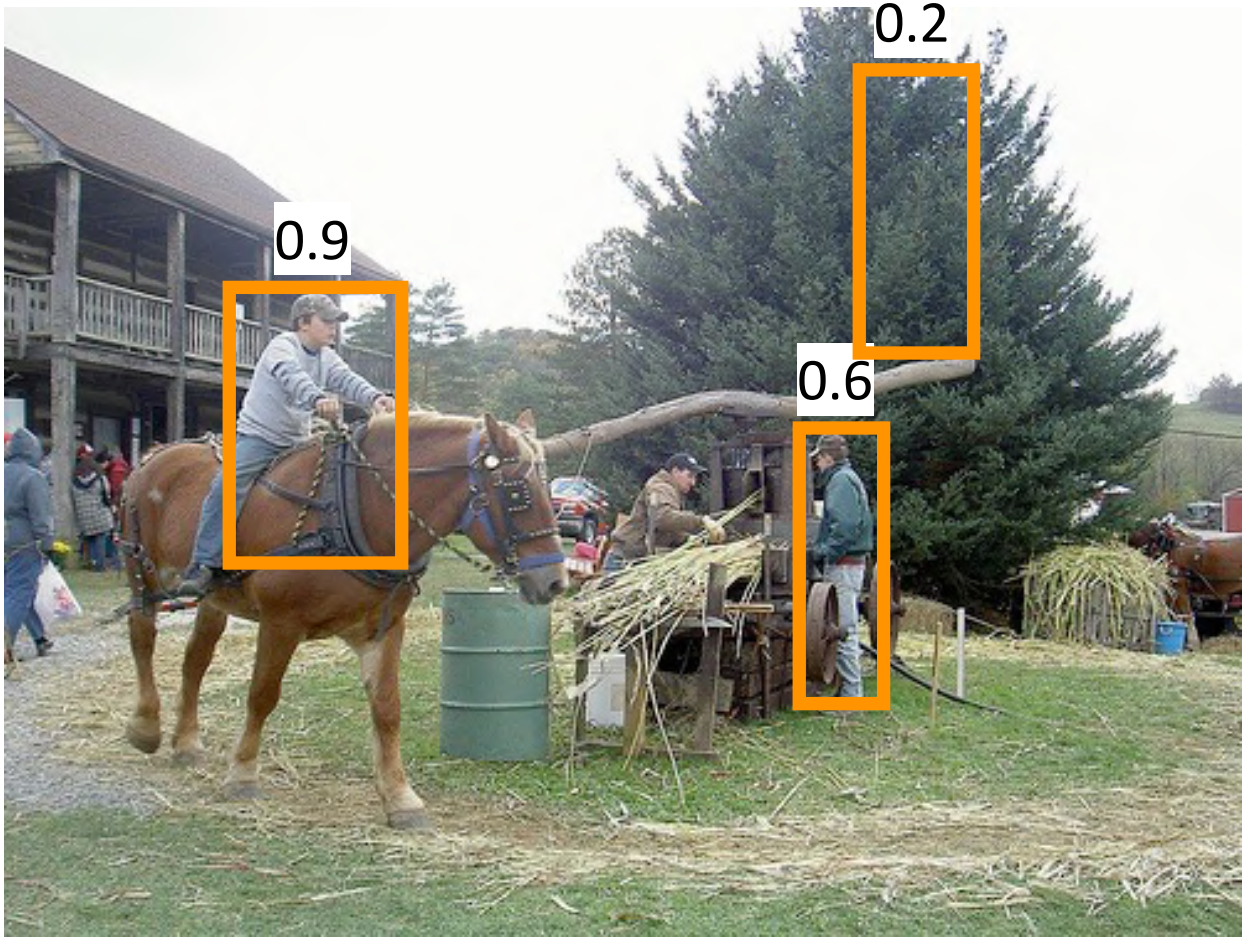
Second Detection



‘person’ detector predictions

Evaluating a detector

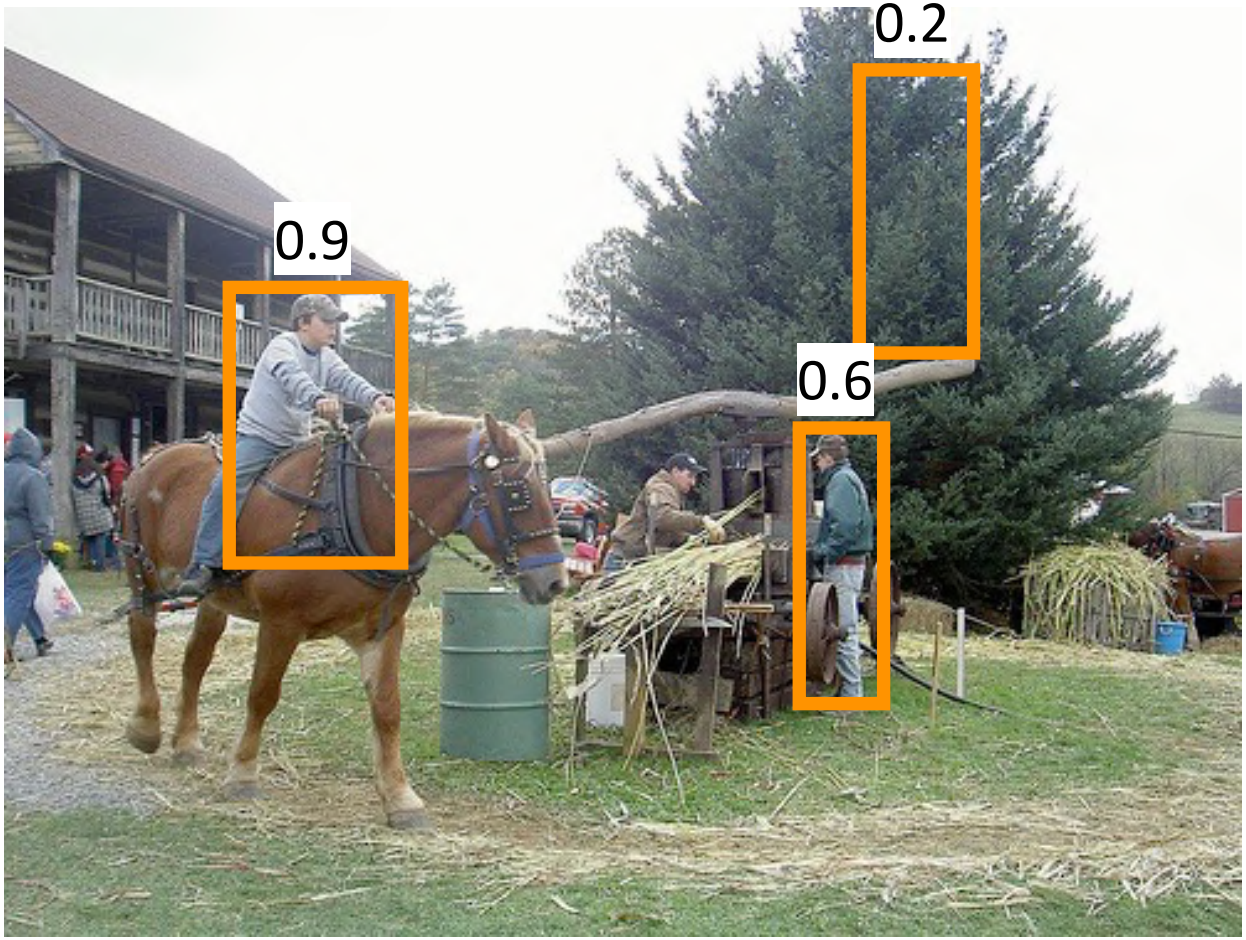
Third Detection



‘person’ detector predictions

Evaluating a detector

Compared to GT

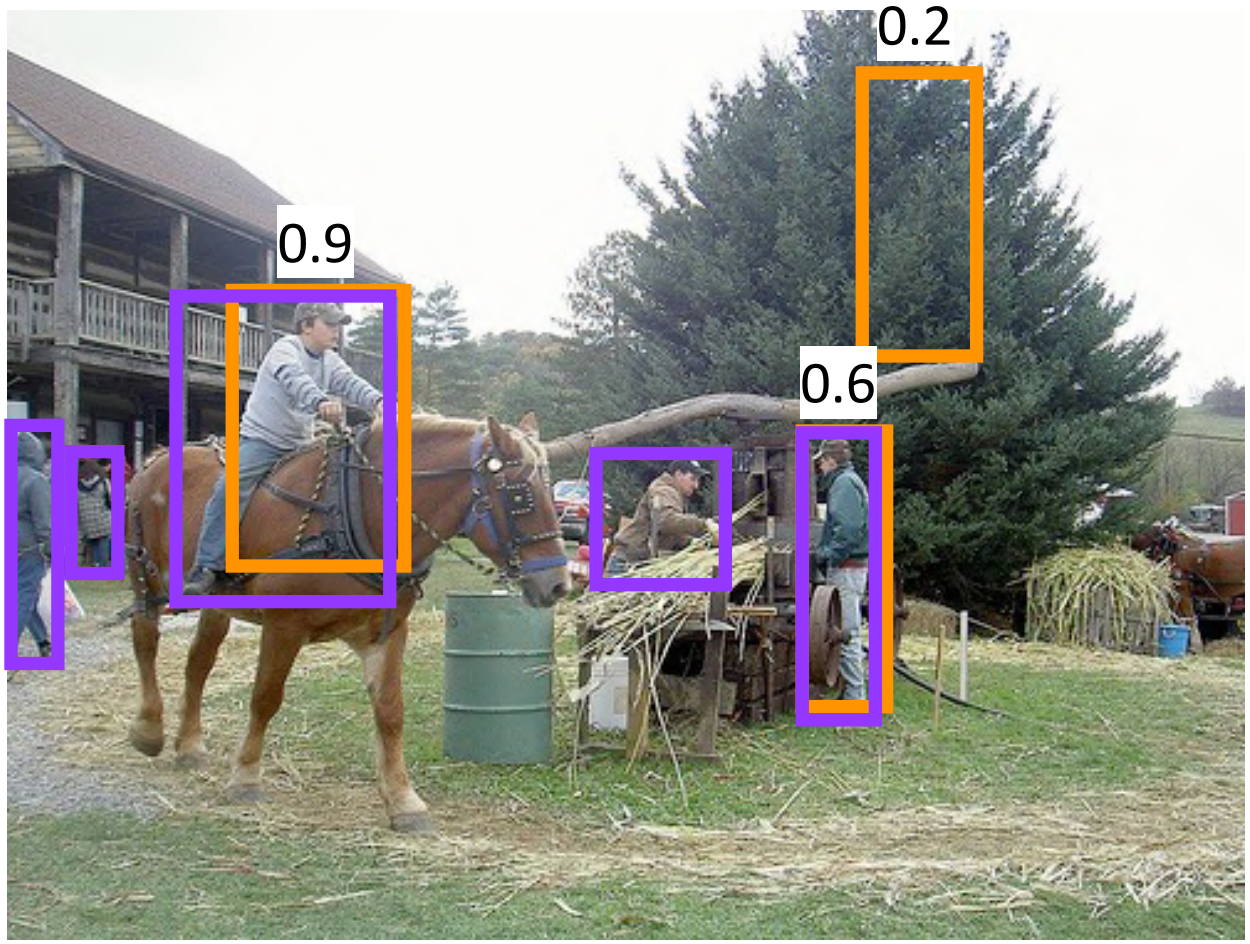


‘person’ detector predictions

ground truth ‘person’ boxes

Evaluating a detector

Compared to GT

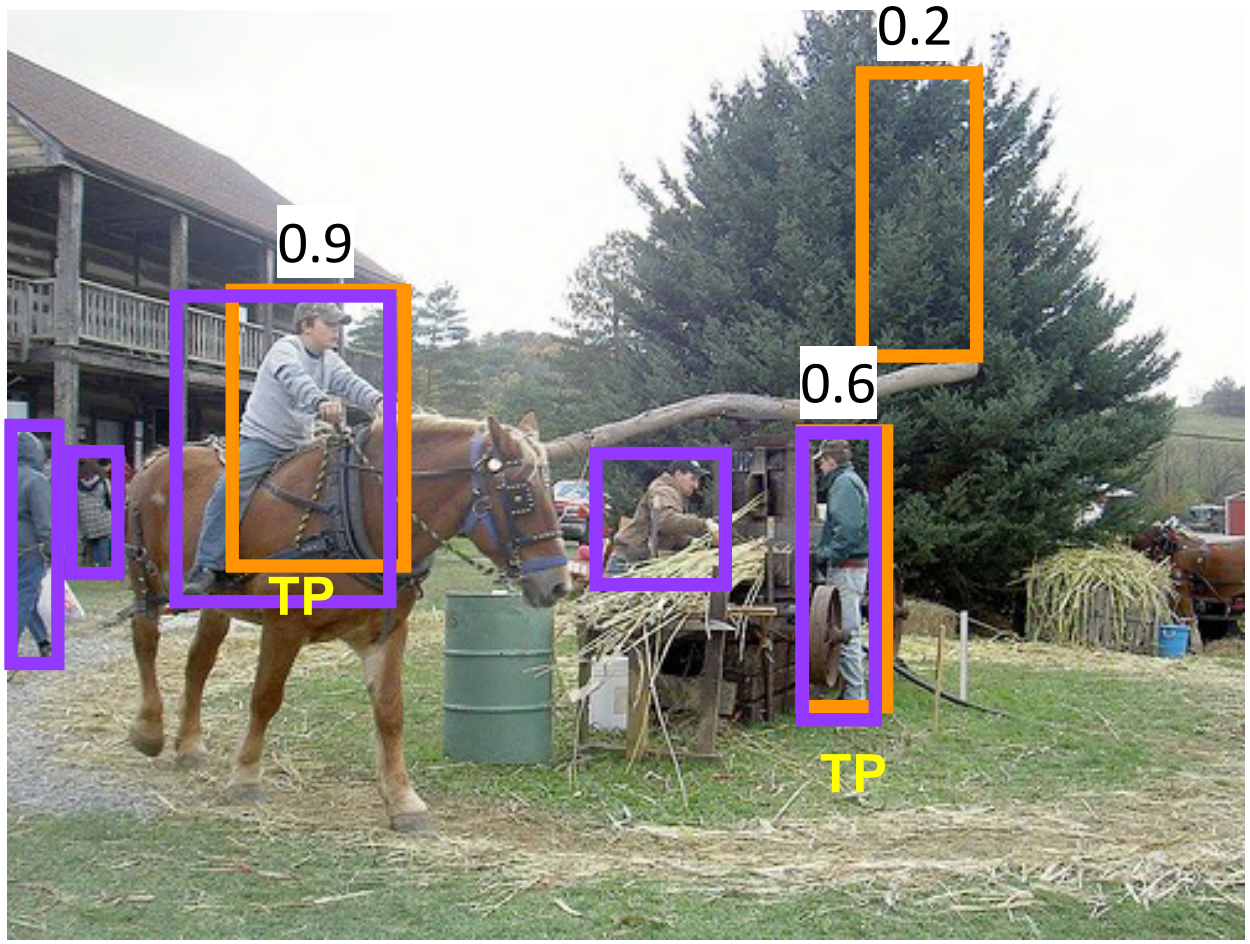


‘person’ detector predictions

ground truth ‘person’ boxes

Evaluating a detector

Compared to GT

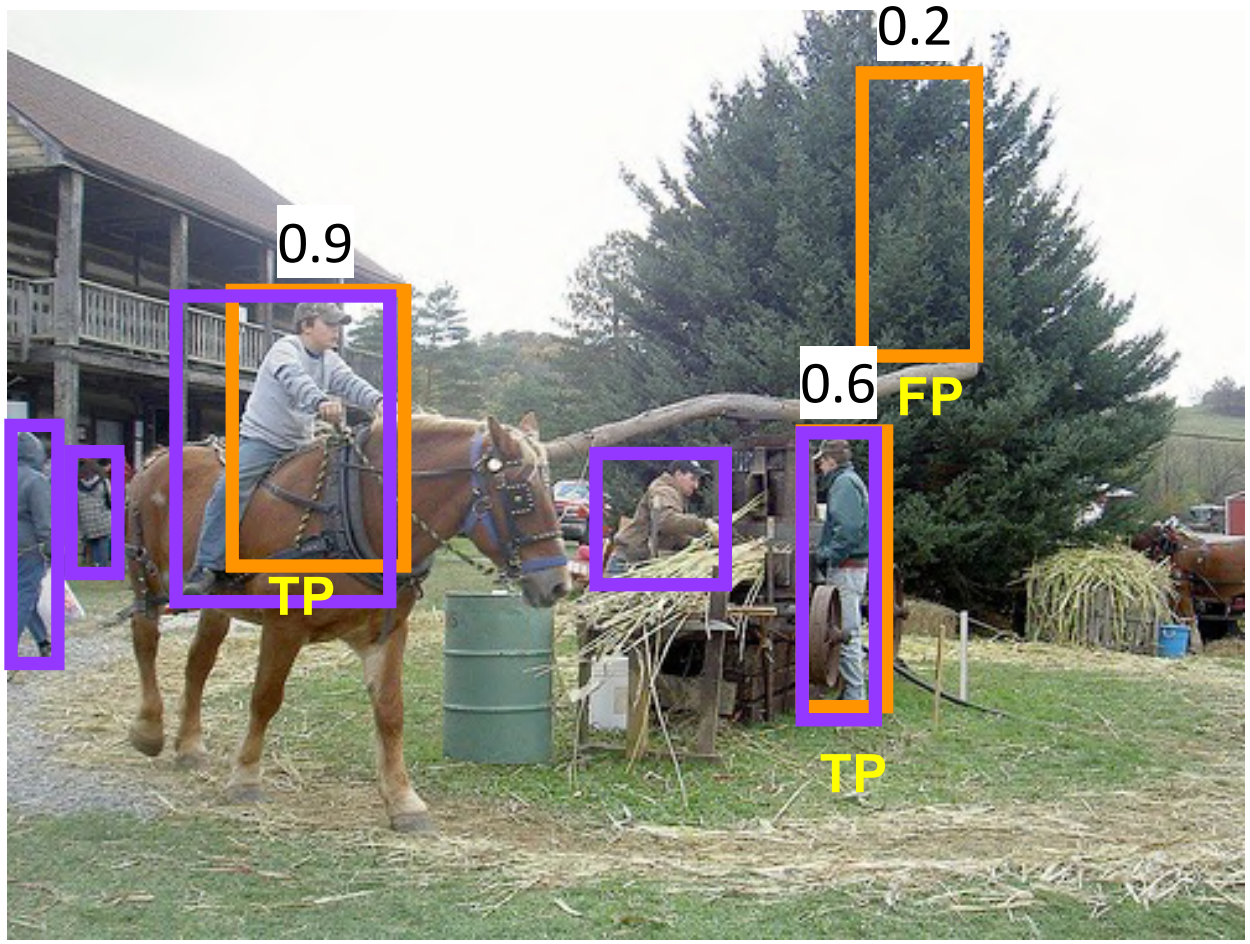


‘person’ detector predictions

ground truth ‘person’ boxes

Evaluating a detector

Compared to GT

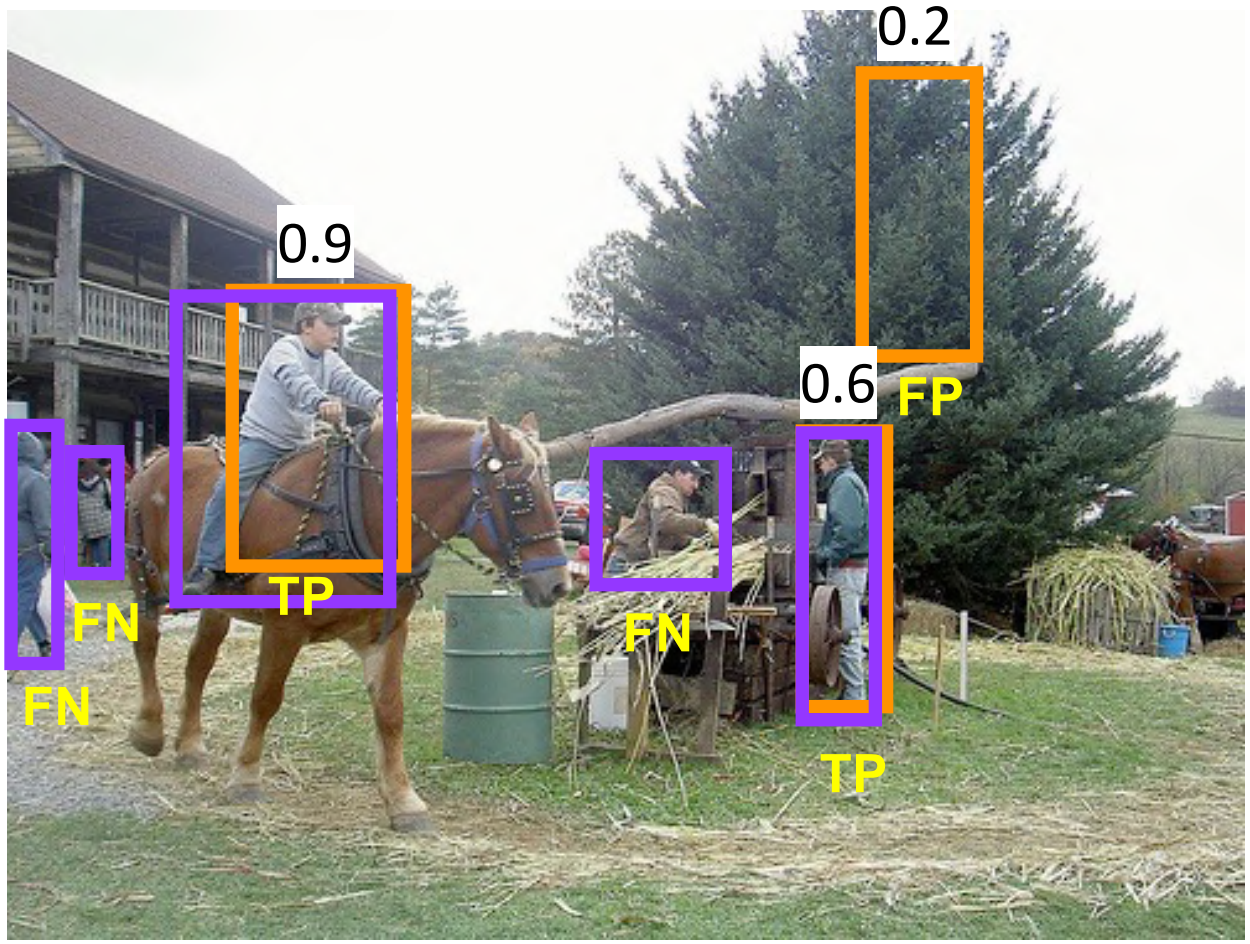


‘person’ detector predictions

ground truth ‘person’ boxes

Evaluating a detector

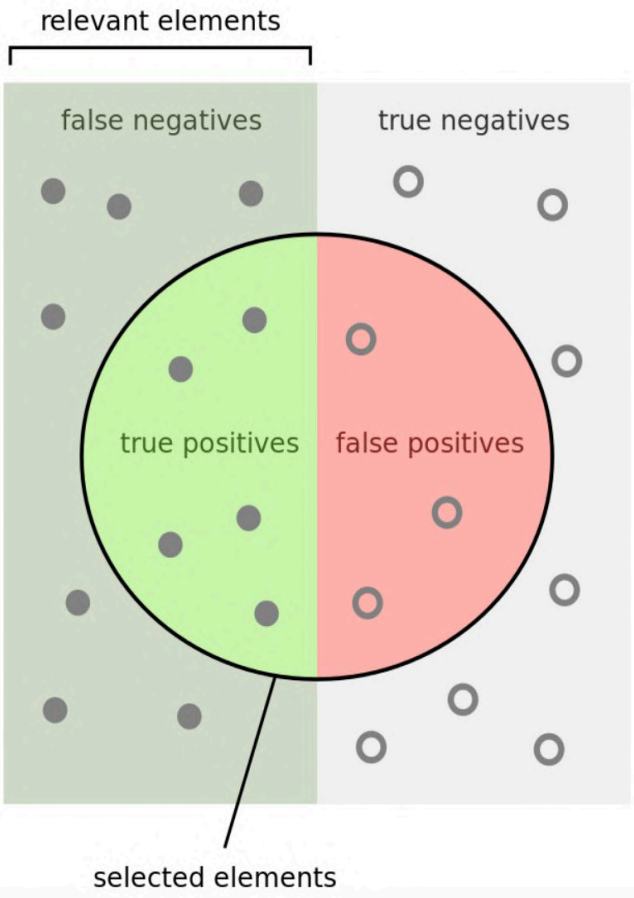
Compared to GT



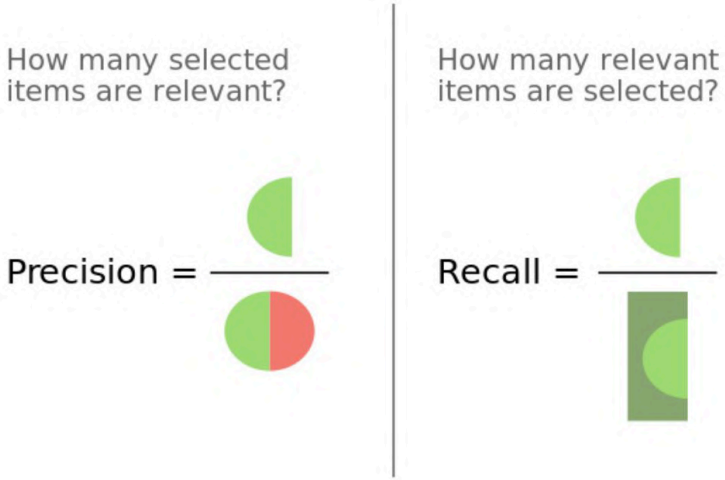
‘person’ detector predictions

ground truth ‘person’ boxes

Evaluating a detector



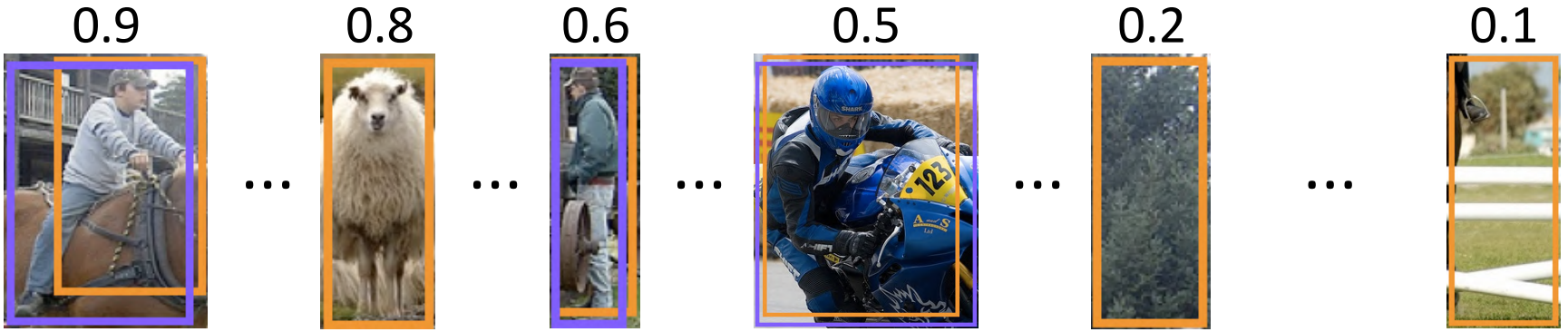
Precision vs. Recall



$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

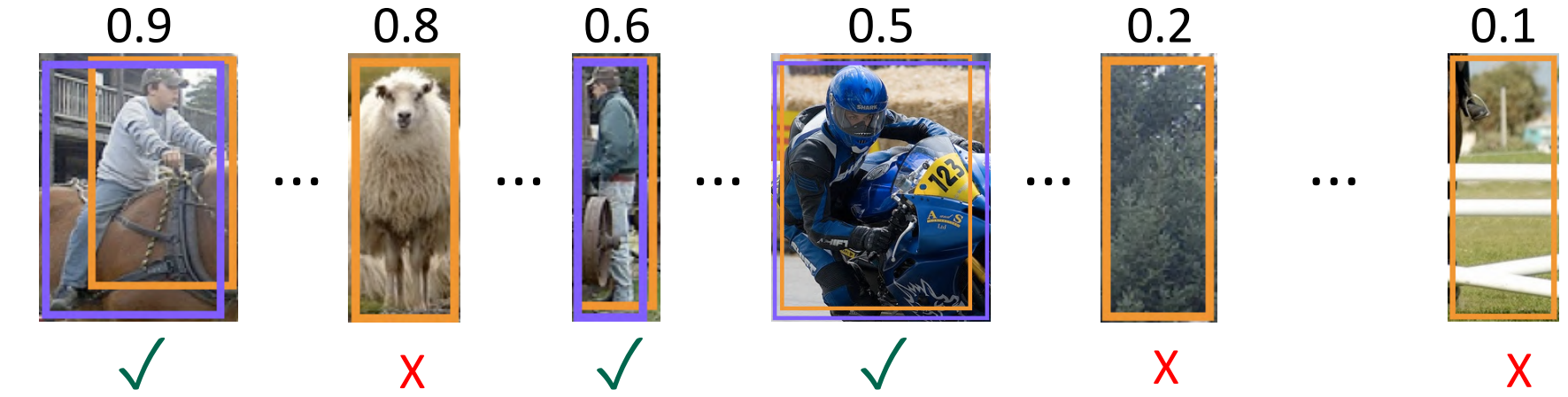
Evaluating a detector

- Sort predictions by confidence
- TP = IoU(prediction, some GT) ≥ 0.5
- FP = IoU(prediction, any GT) < 0.5



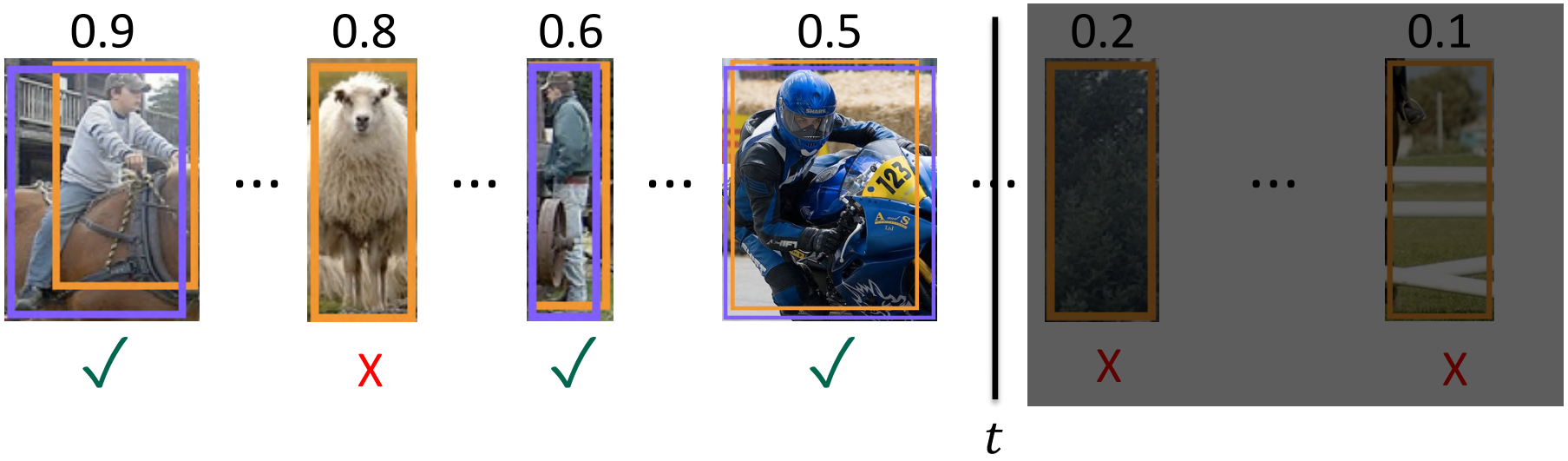
Evaluating a detector

- Sort predictions by confidence



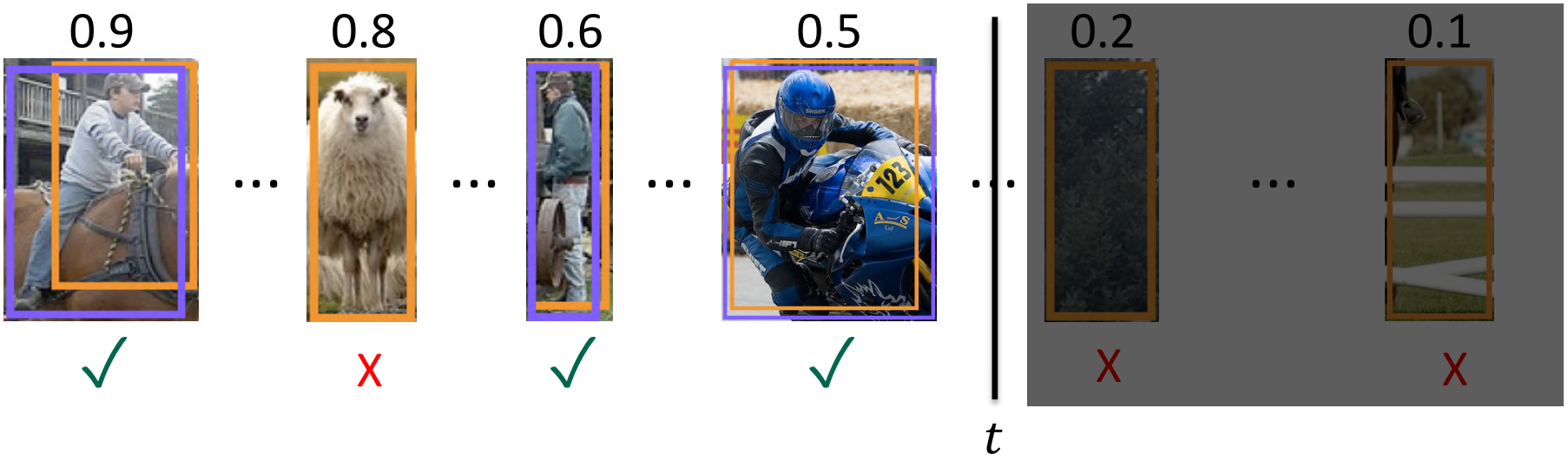
Evaluating a detector

- Sort predictions by confidence
- Choose a confidence threshold



Evaluating a detector

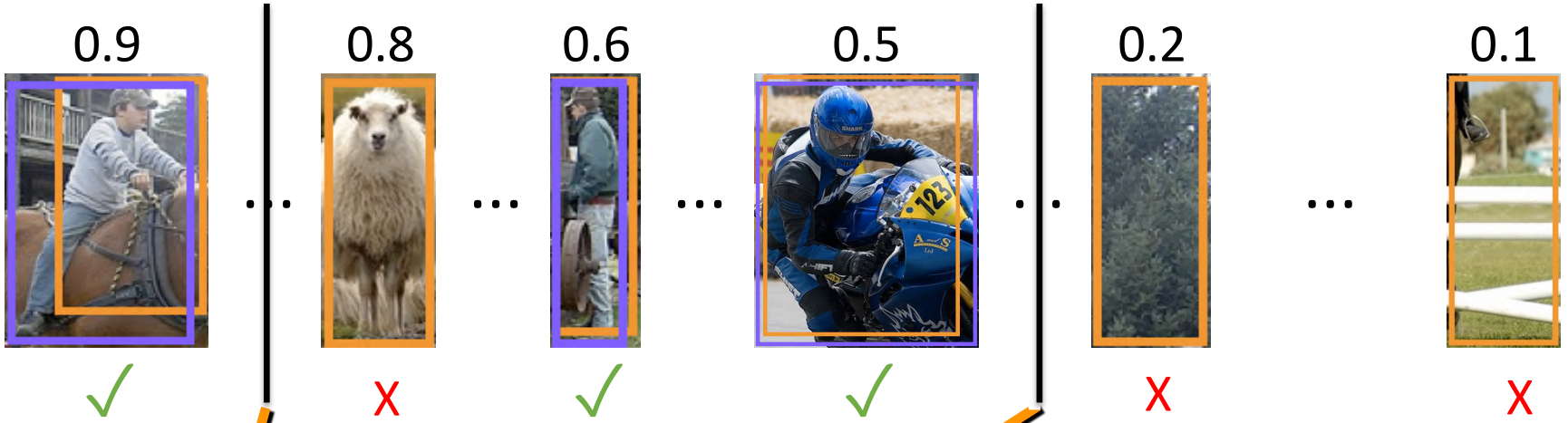
- Sort predictions by confidence
- Choose a confidence threshold
- Evaluate metric

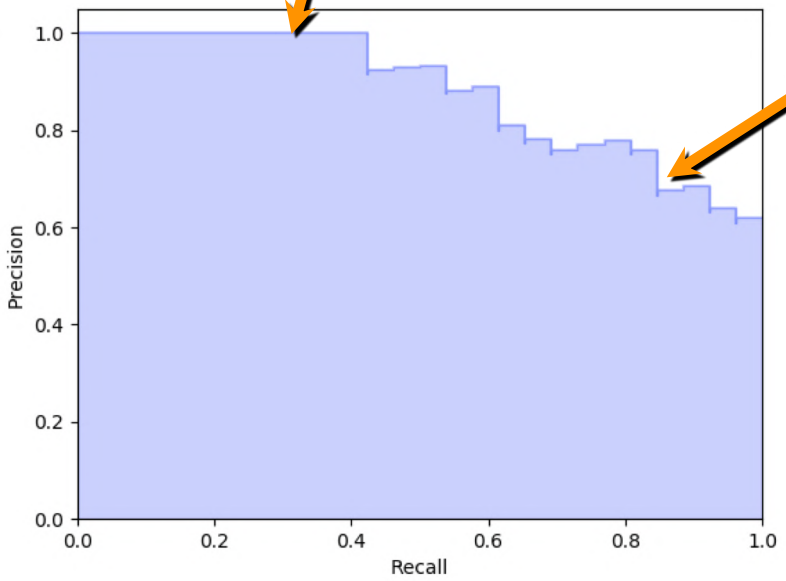


$$precision(t) = \frac{TP(t)}{TP(t) + FP(t)} = \frac{\checkmark}{\checkmark + \text{X}}$$

$$recall(t) = \frac{TP(t)}{GT} = \frac{TP(t)}{TP(t) + FN(t)} = \frac{\checkmark}{\square}$$

Evaluating a detector





Average Precision (AP):

The area under the PR curve

0% is worst

100% is best

mean AP (**mAP**):

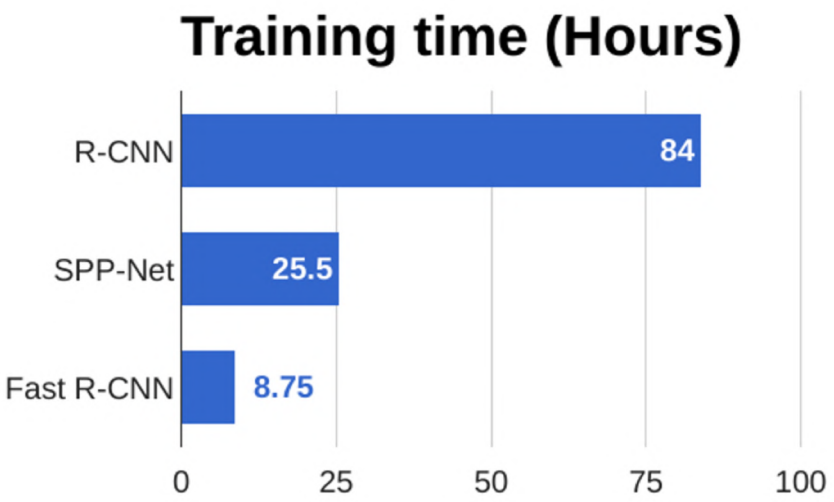
The mean of AP over all classes

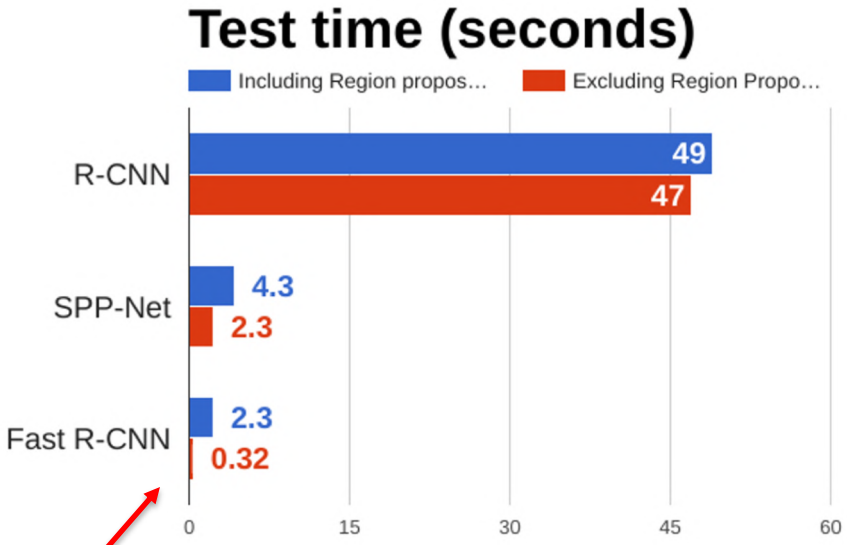
mAP: Fast R-CNN vs. R_CNN

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

Source: Justin Johnson

R-CNN vs. Fast R-CNN





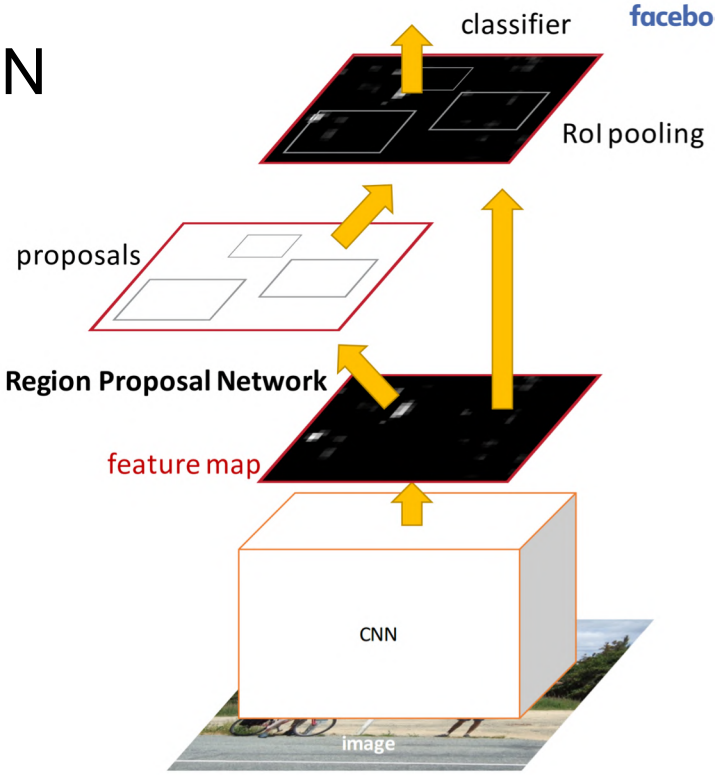
Problem:
Runtime is dominated by the Region Proposals

Source: Justin Johnson

Faster R-CNN

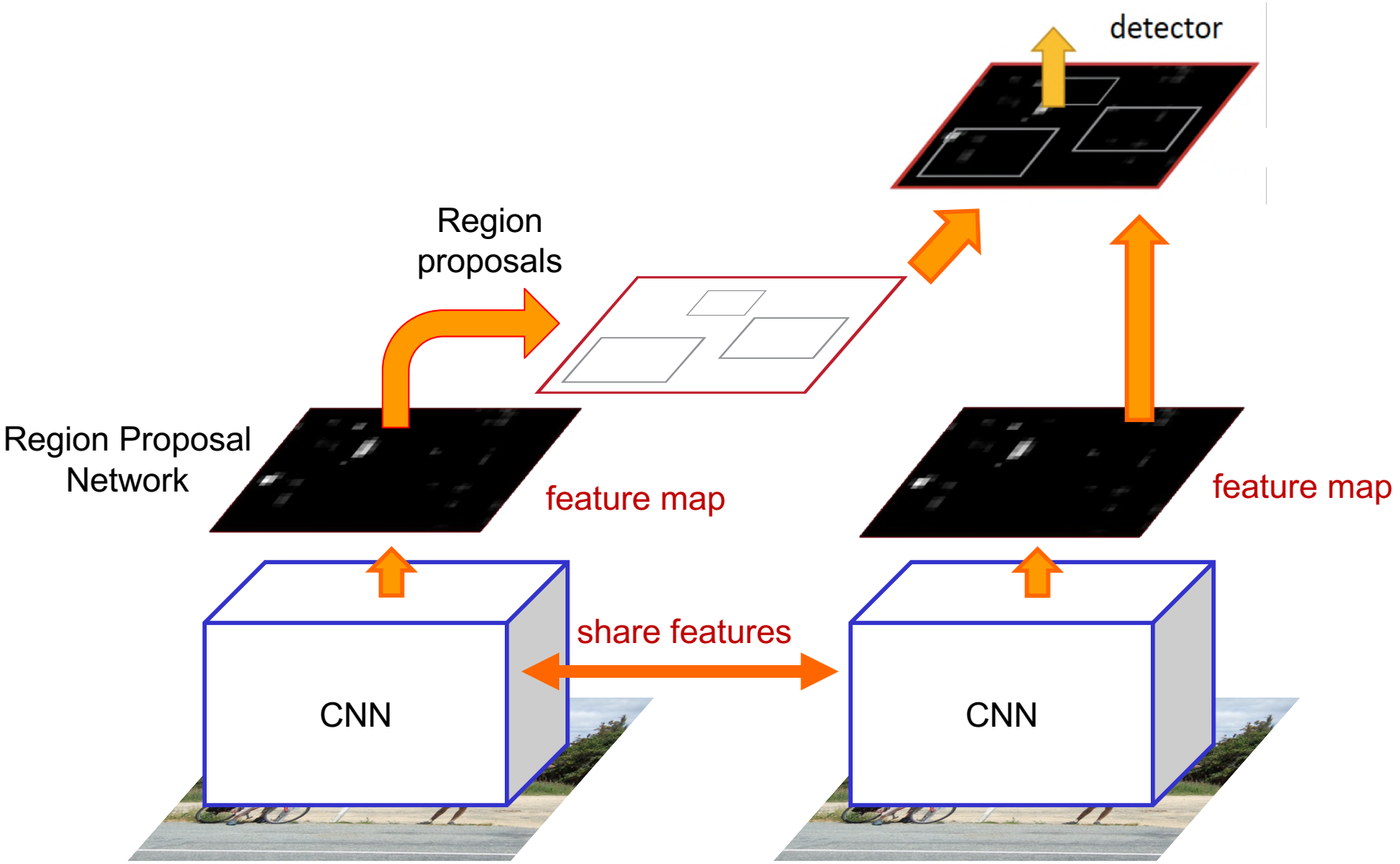
Main idea:

- Use a region proposal network (RPN) to predict RP from CNN features.
- Use the same features for the RPN and for class/box regressions.



S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS15

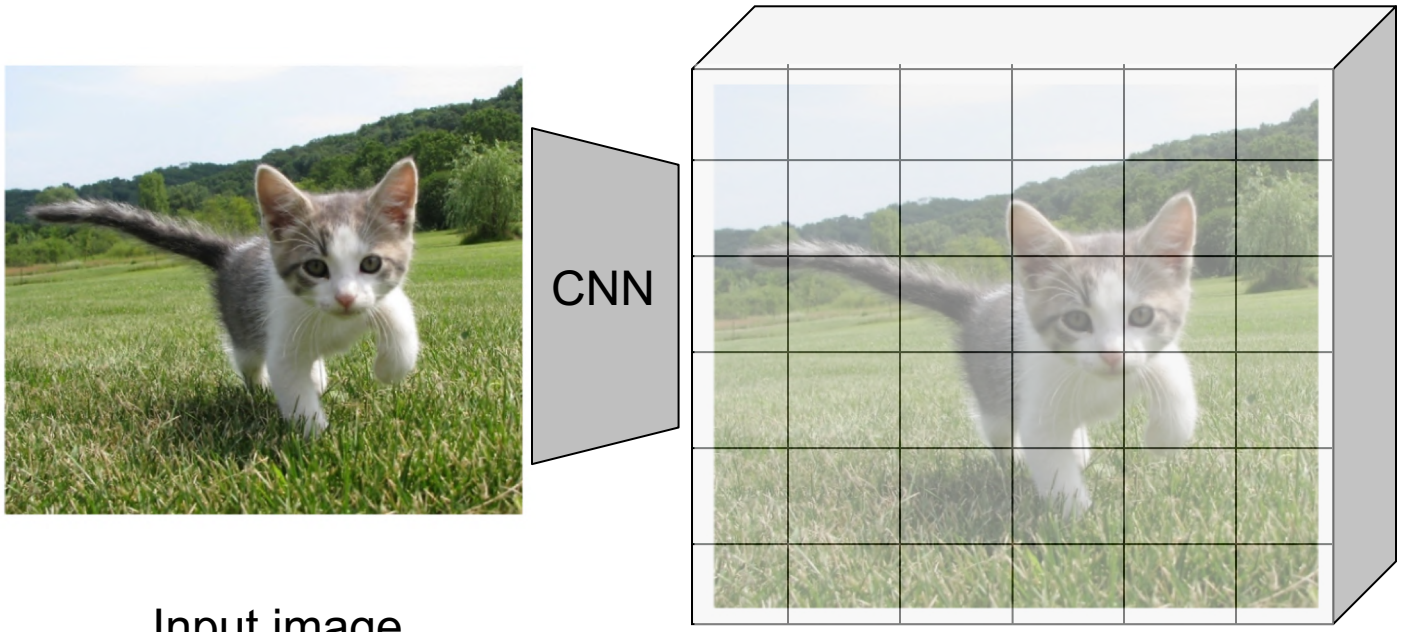
Faster R-CNN



S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS15

Faster R-CNN: RPN

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



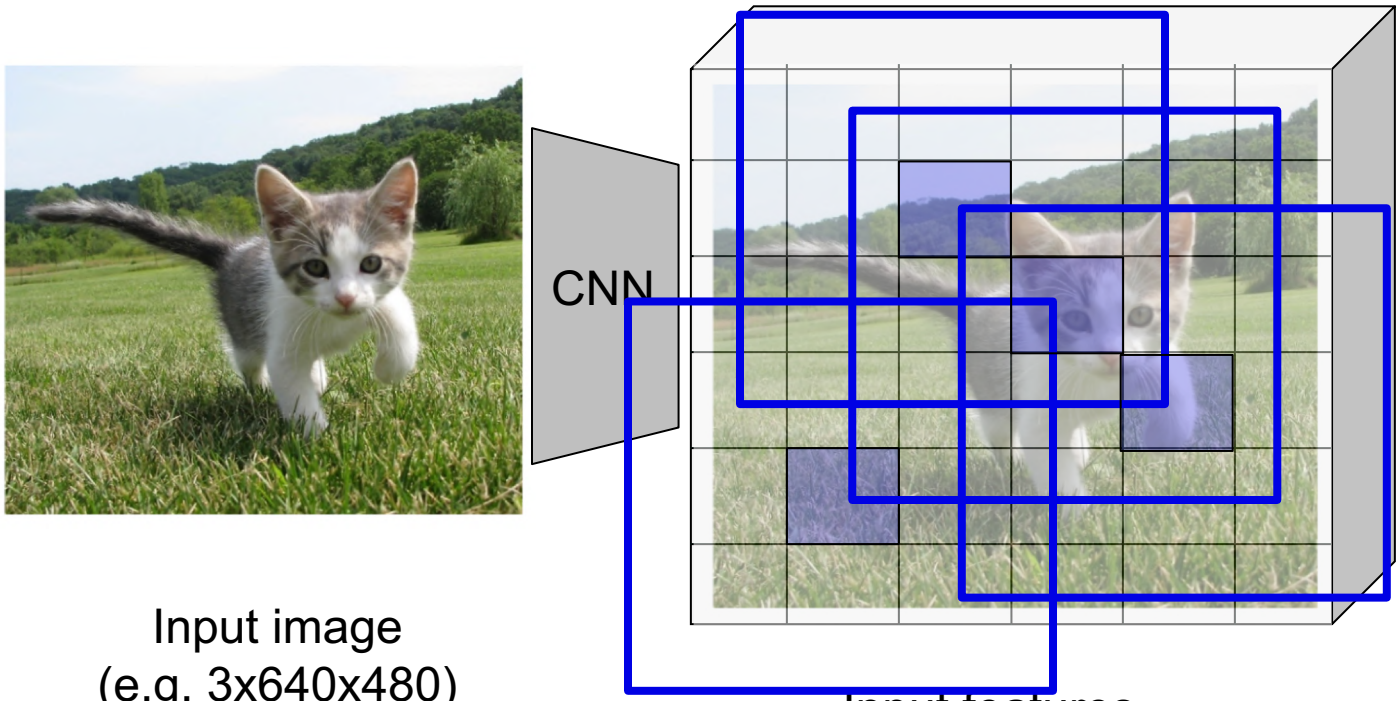
Input image
(e.g. 3x640x480)

Input features
(e.g. 512x20x15)

Source: Justin Johnson

Faster R-CNN: RPN

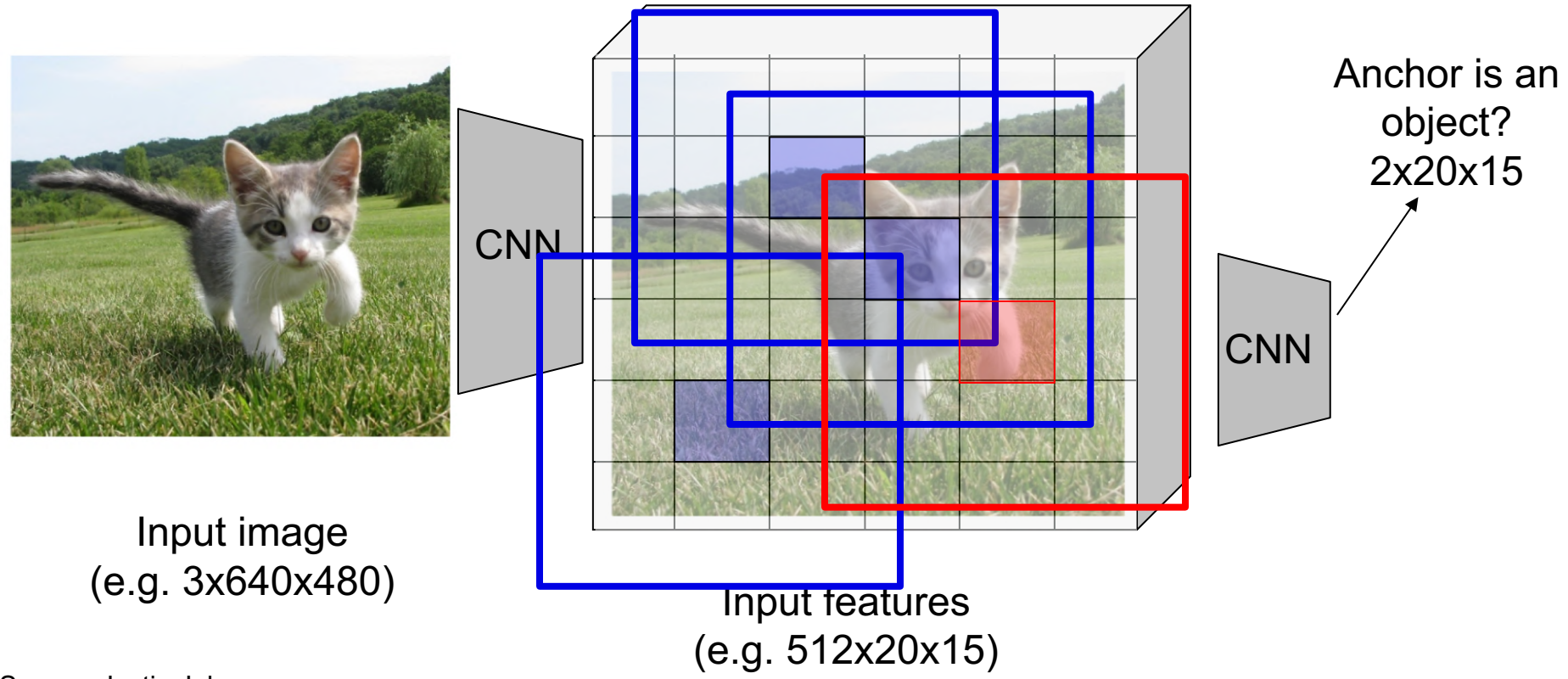
- Imagine an **anchor box** of fixed size at each point in the feature map.



Source: Justin Johnson

Faster R-CNN: RPN

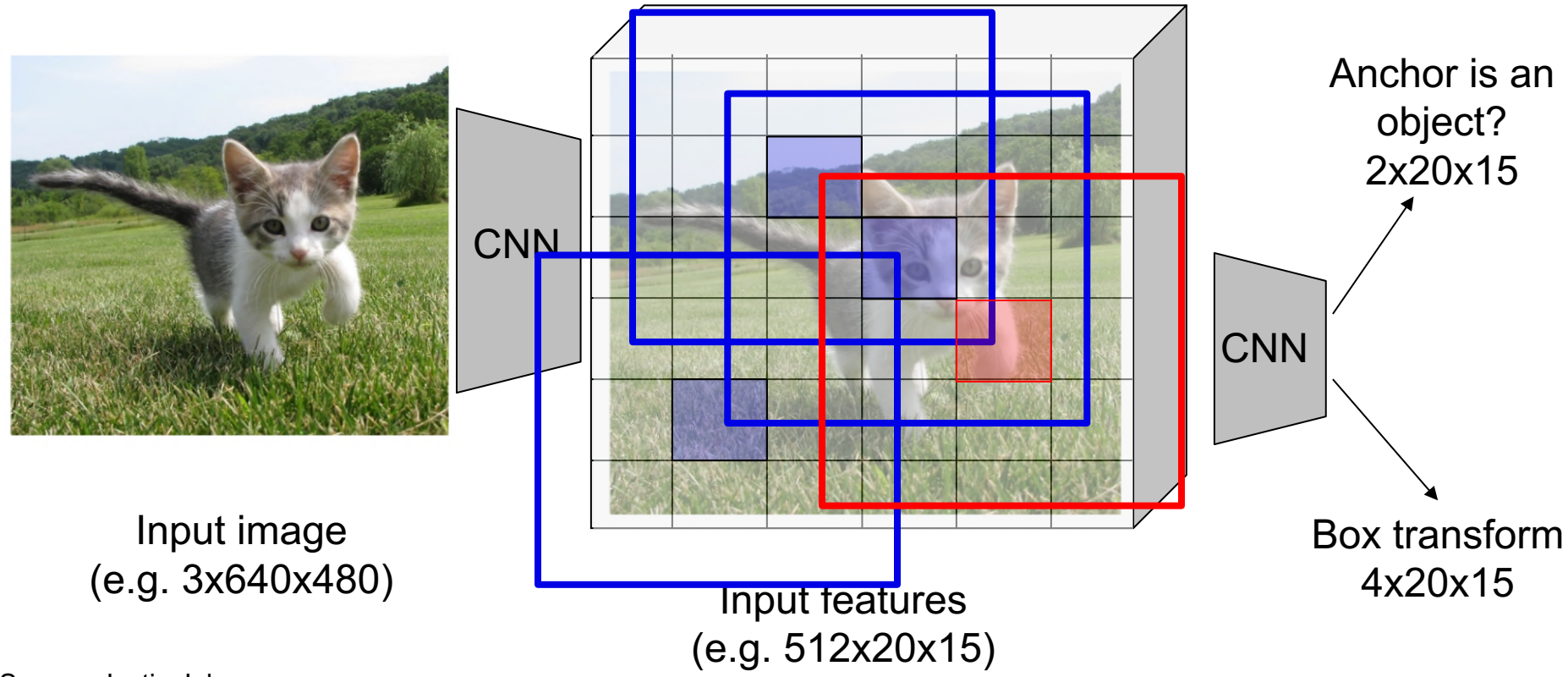
- Imagine an **anchor box** of fixed size at each point in the feature map.



Source: Justin Johnson

Faster R-CNN: RPN

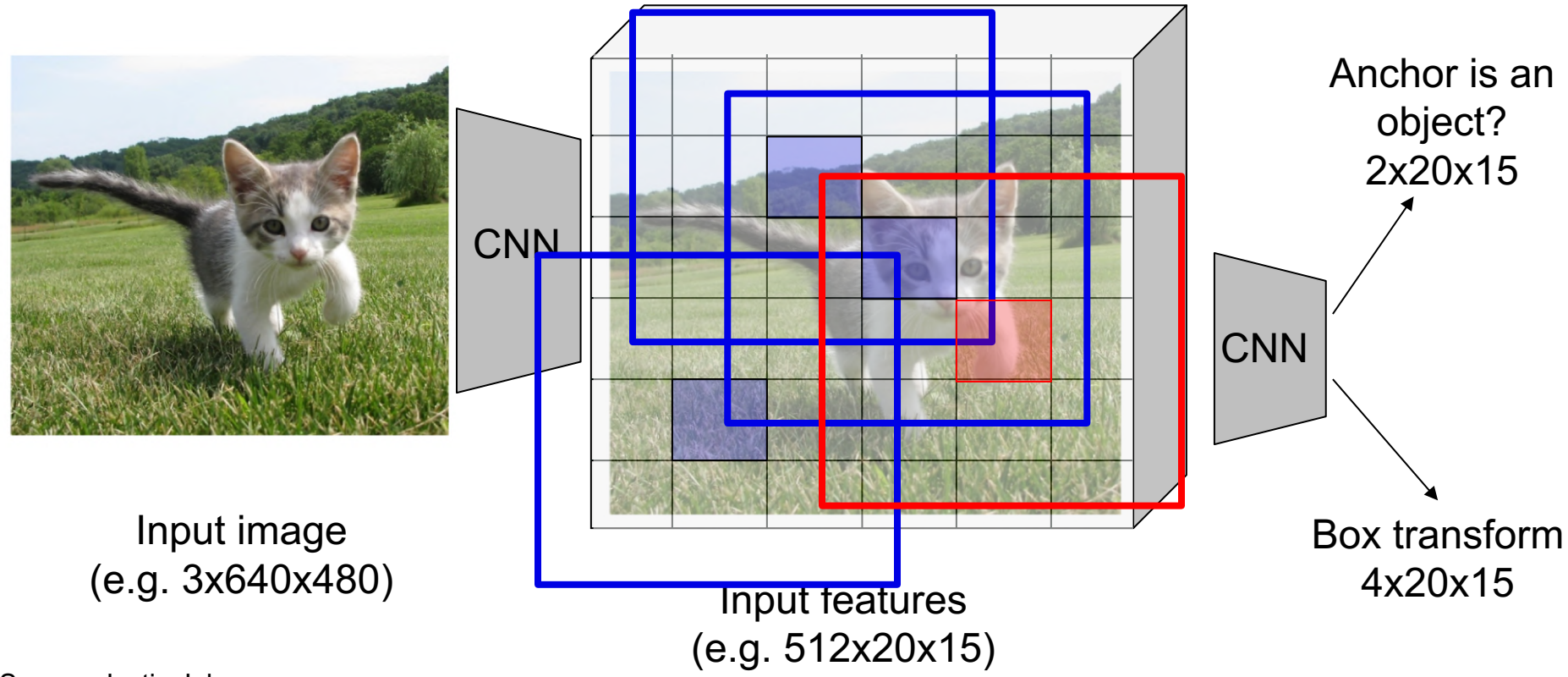
- Imagine an **anchor box** of fixed size at each point in the feature map.



Source: Justin Johnson

Faster R-CNN: RPN

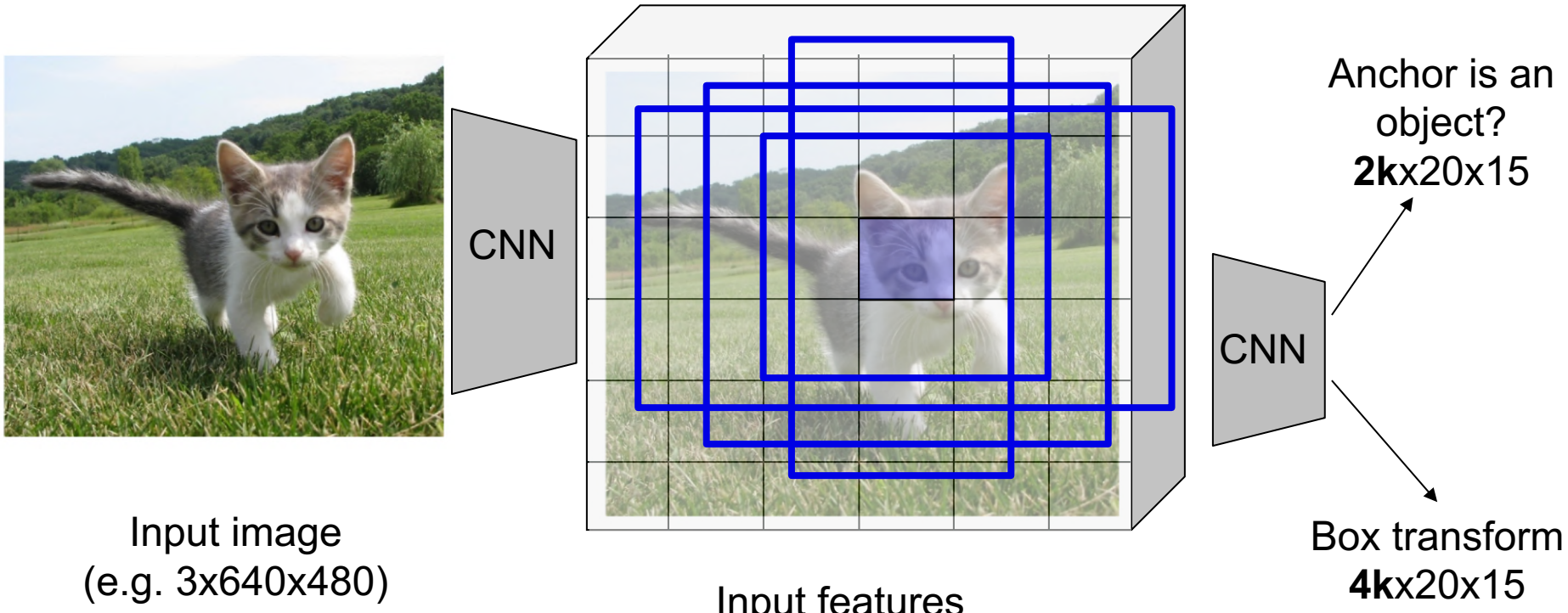
- **Problem**: Anchor box may have the wrong size / shape
- **Solution**: Use k different anchor boxes at each point.



Source: Justin Johnson

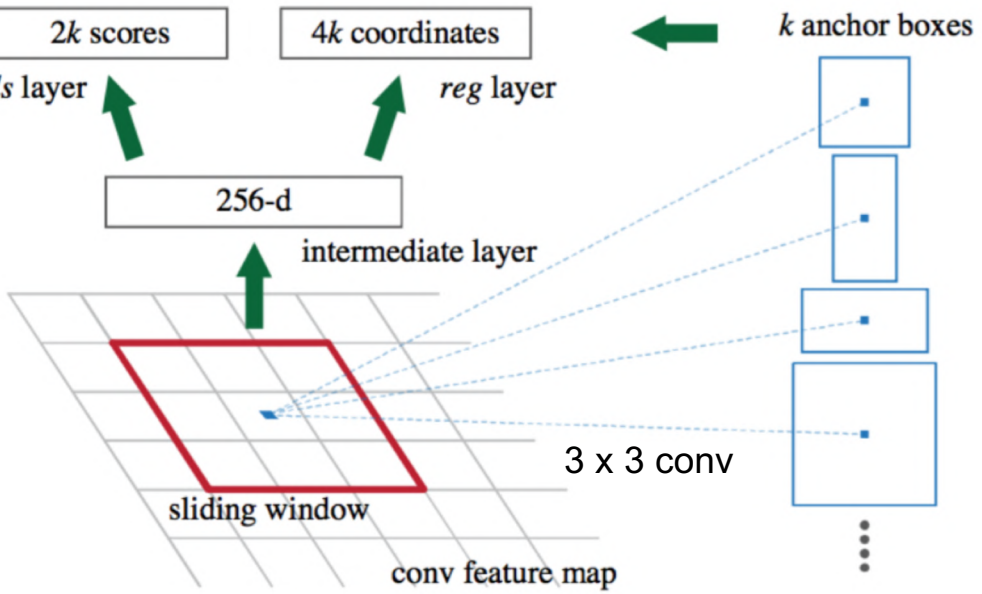
Faster R-CNN: RPN

- **Problem**: Anchor box may have the wrong size / shape
- **Solution**: Use **k different anchor boxes** at each point.



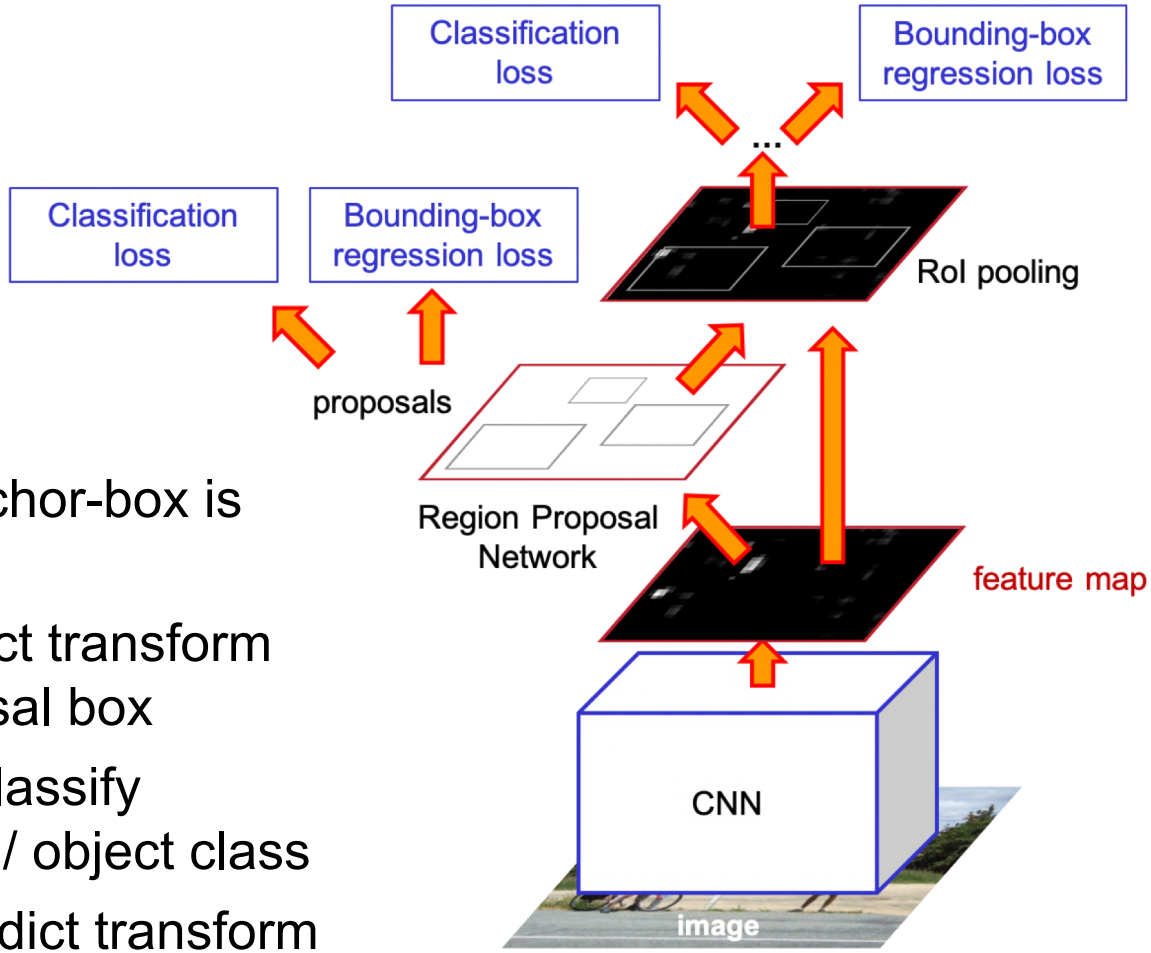
Source: Justin Johnson

Faster R-CNN: RPN



- At each location, hypothesize the k anchor boxes
- Build a small network for:
  - classifying object/non-object
  - regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window

Faster R-CNN: Training



Jointly train with 4 losses:

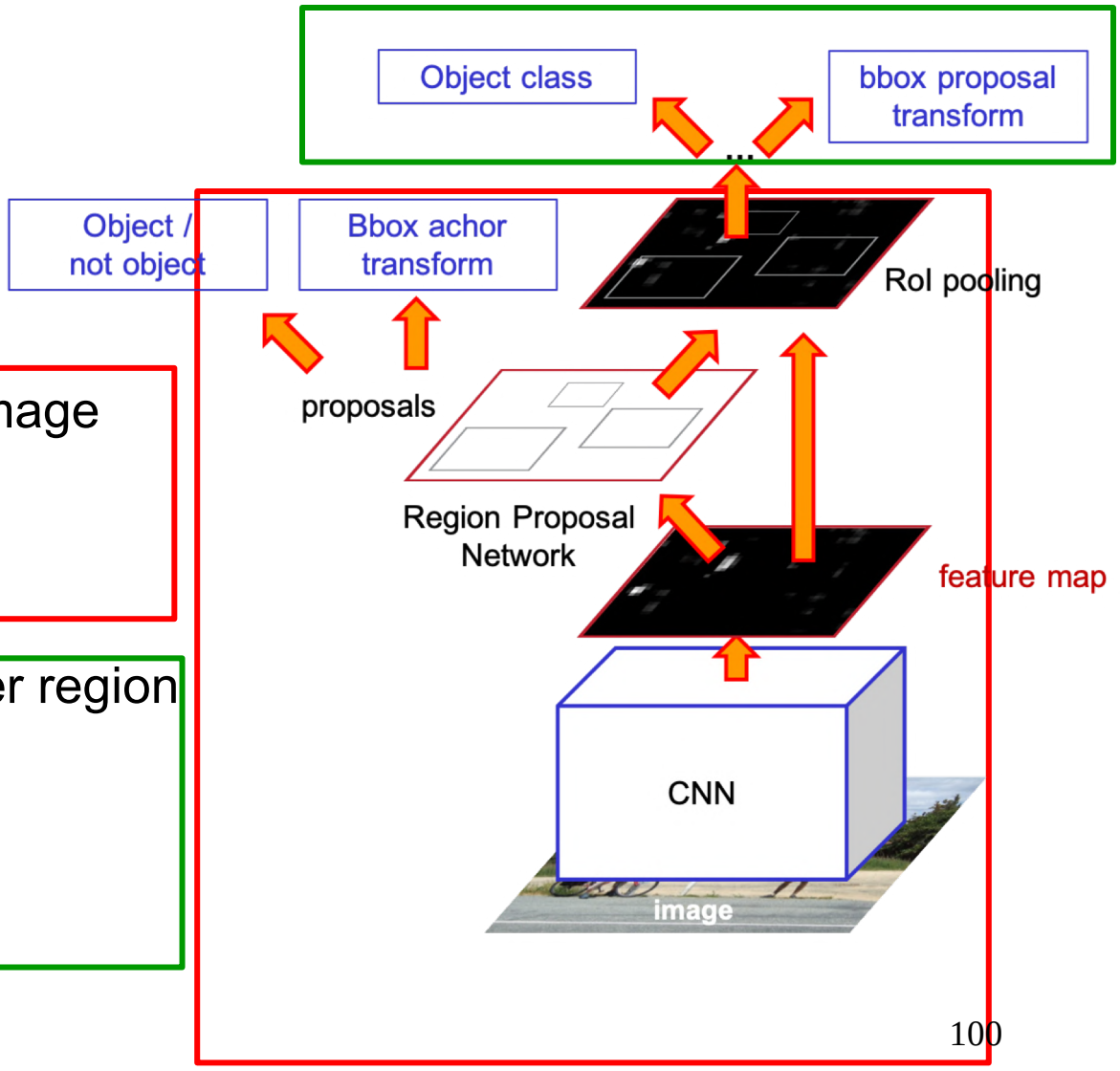
1. **RPN Classification**: anchor-box is object /not an object
2. **RPN Regression**: Predict transform from ancor box to proposal box
3. **Object classification**: classify proposal as background / object class
4. **Object Regression**: predict transform from proposal box to object box

Faster R-CNN: Testing

Two stage object detector:

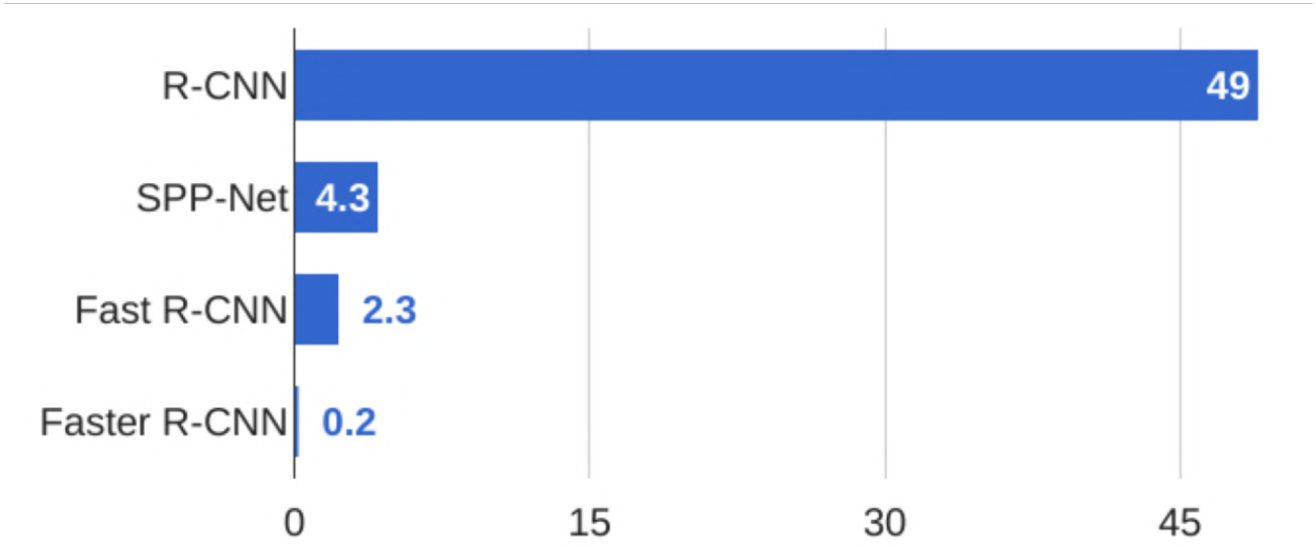
First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop feature / ROI align
- Predict object class
- Predict bbox offest



Faster R-CNN: Results

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

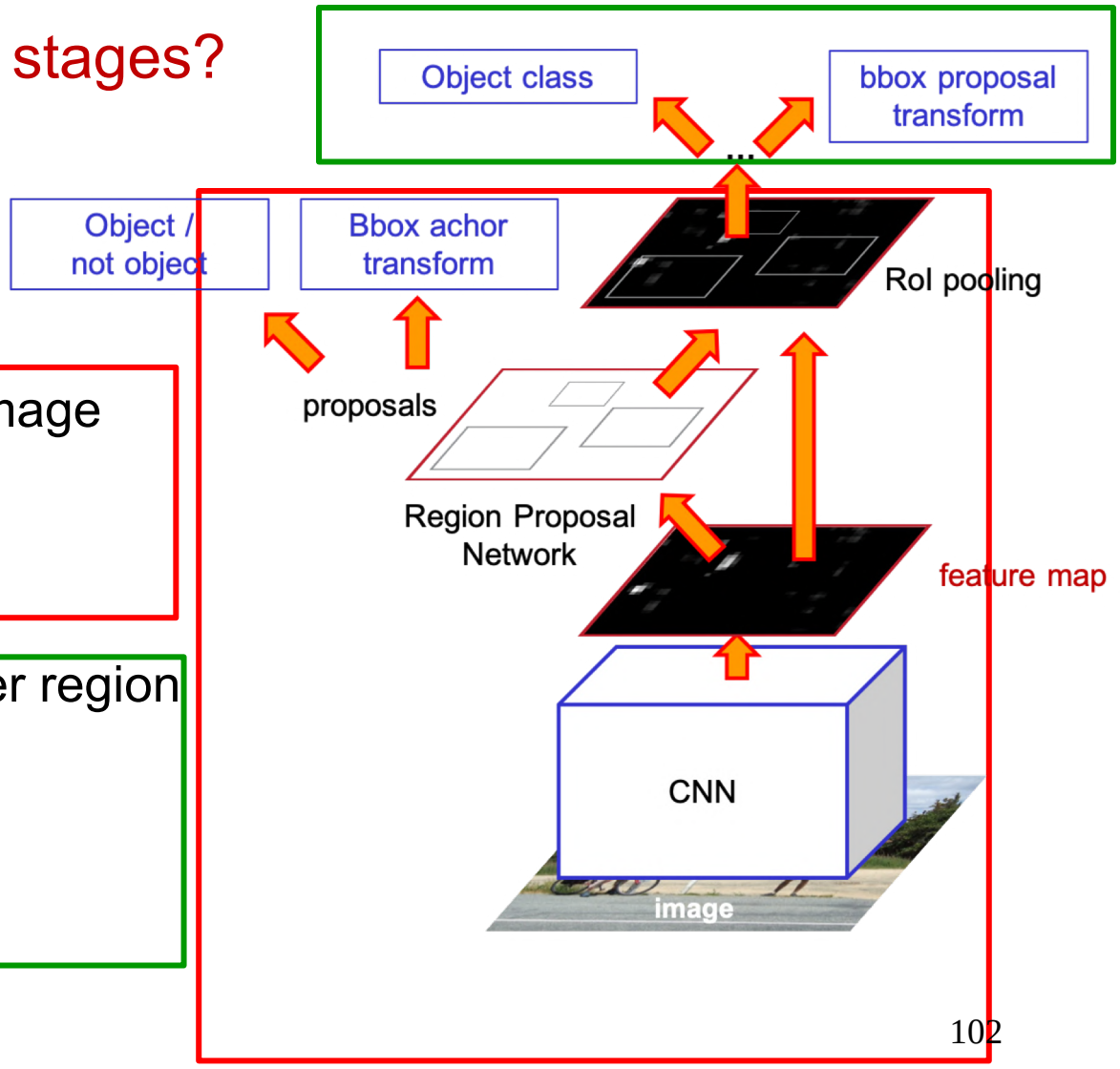


R_CNN testing-time speed

Source: Justin Johnson

Single Stage Object Detection

Do we really need two stages?



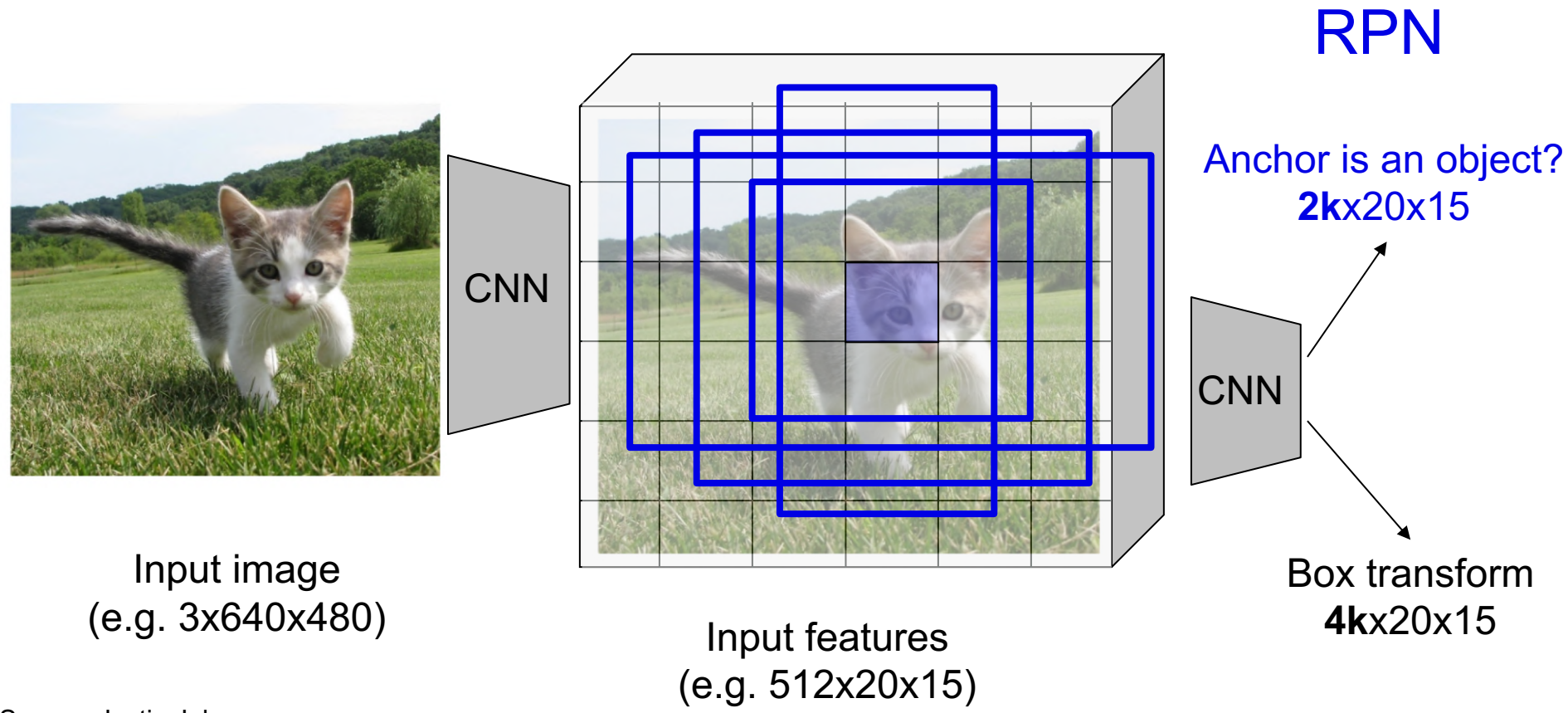
Two stage object detector:

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

Second stage: Run once per region
- Crop feature / ROI align
- Predict object class
- Predict bbox offest

Single Stage Object Detection

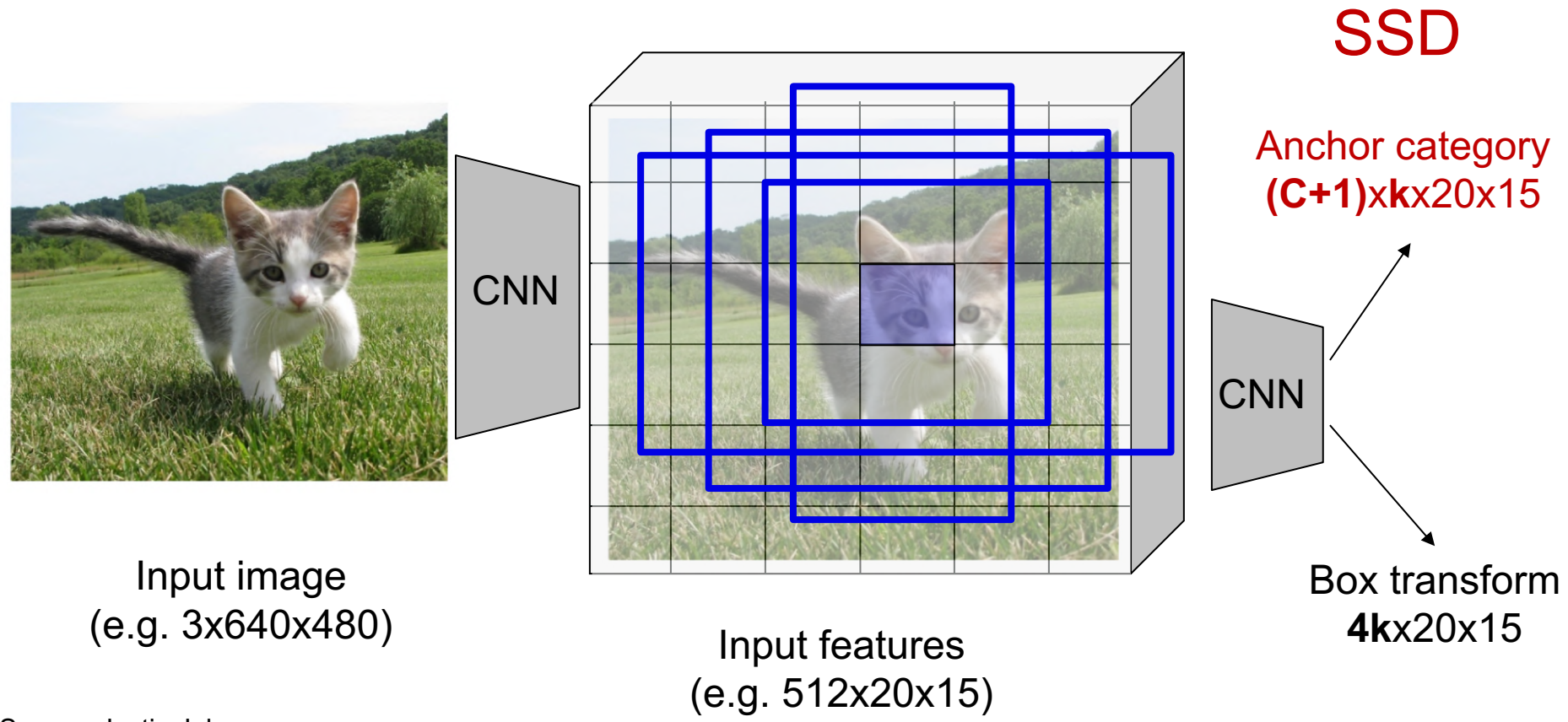
- **RPN:** Classify each anchor box as object / non-object
- **Single stage detector**: Classify each anchor box as one of C categories (or background)



Source: Justin Johnson

Single Stage Object Detection

- **RPN:** Classify each anchor box as object / non-object
- **Single stage detector**: Classify each anchor box as one of C categories (or background)



Source: Justin Johnson

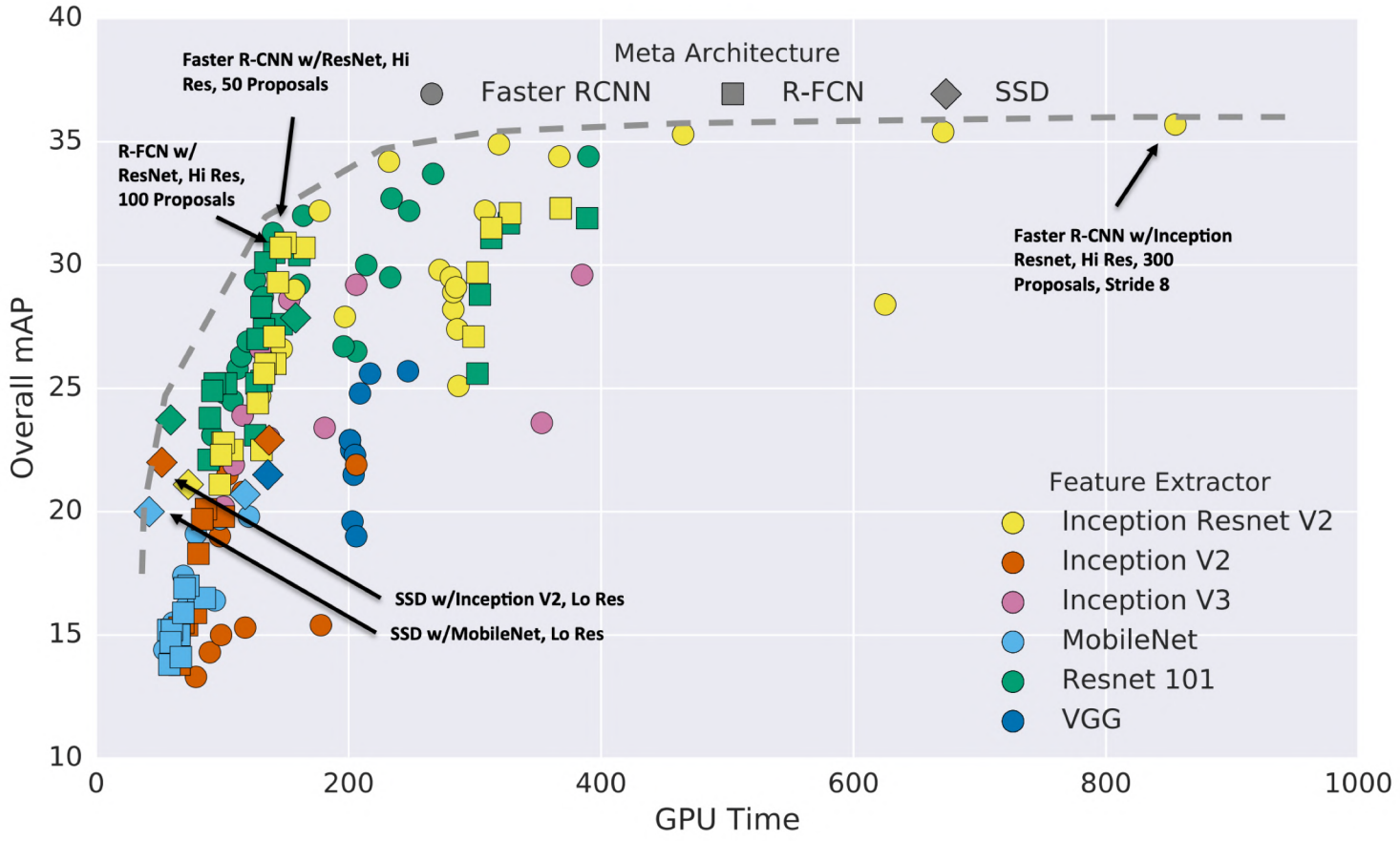
Detectors without proposal

1. YOLO (You Only Look once)
   - Object detection system targeted for real-time processing
   - Divides the input image into an **S×S** grid and look at each grid cell a single object.
2. SSD (Single Shot Detector)
   - Similar to YOLO
   - Hypothesize k anchor boxes at each cell

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, SSD: Single Shot MultiBox Detector, ECCV 2016.

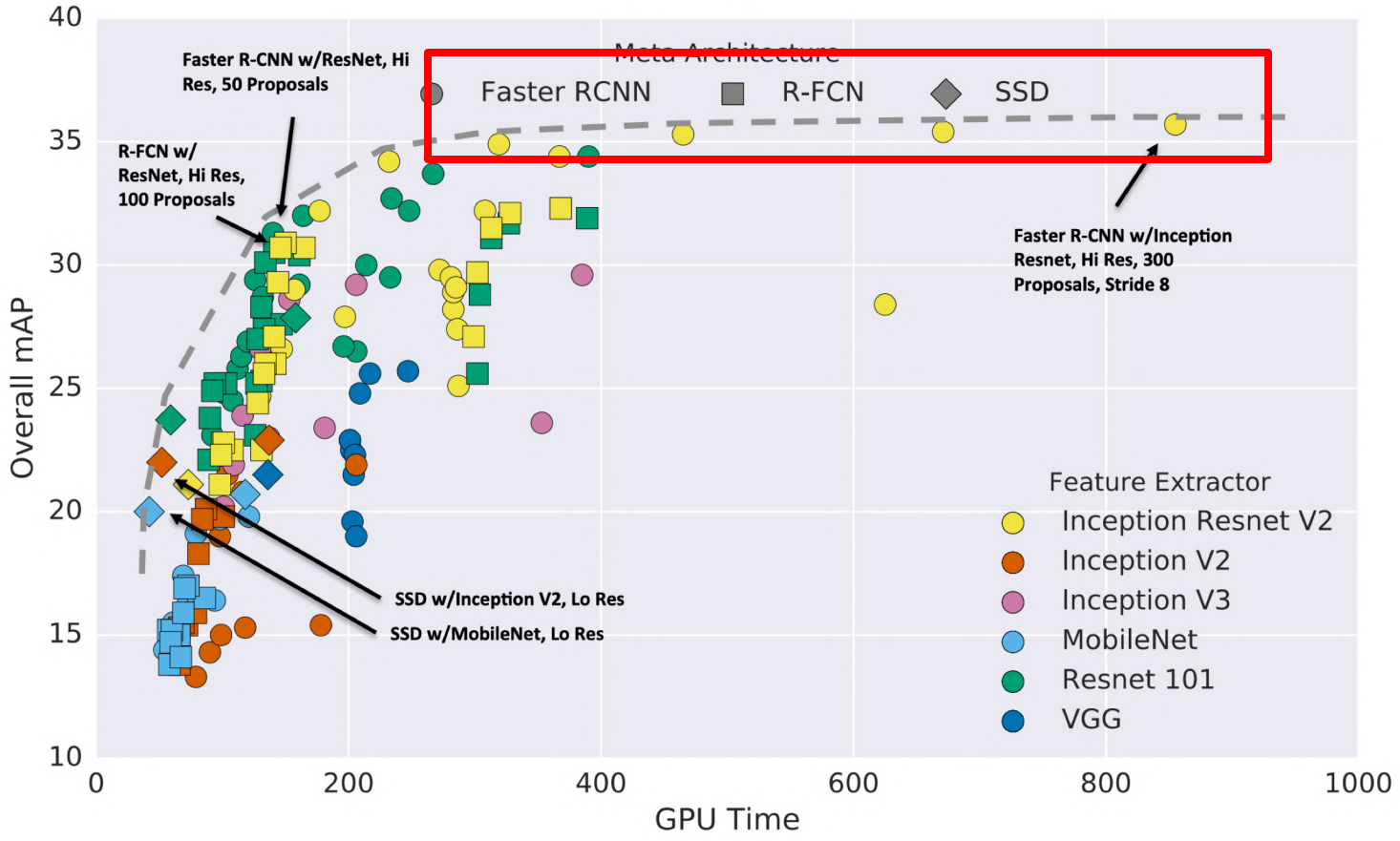
J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

Object Detectors



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

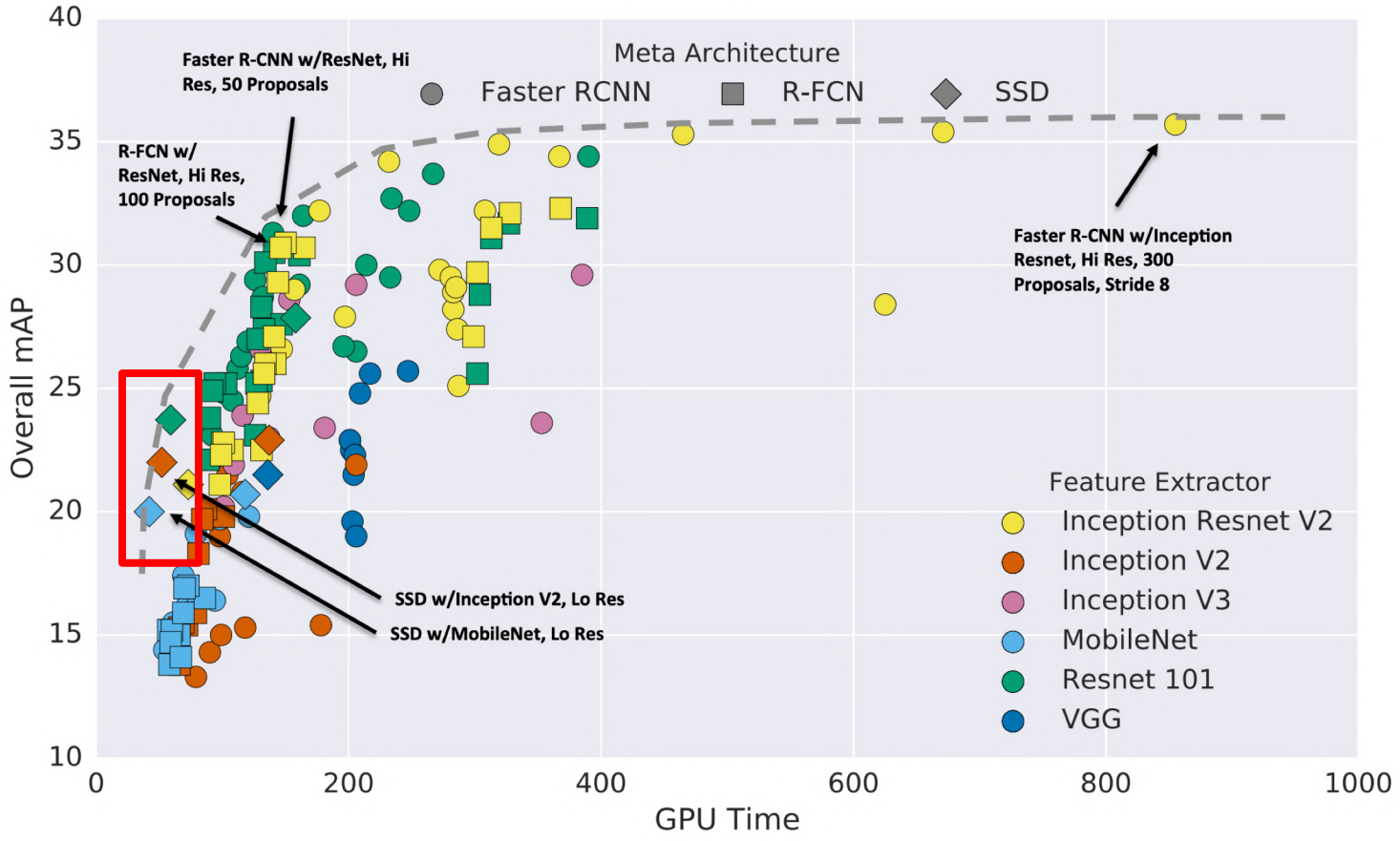
Object Detectors



- Two stages detectors get the best accuracy, but slower

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

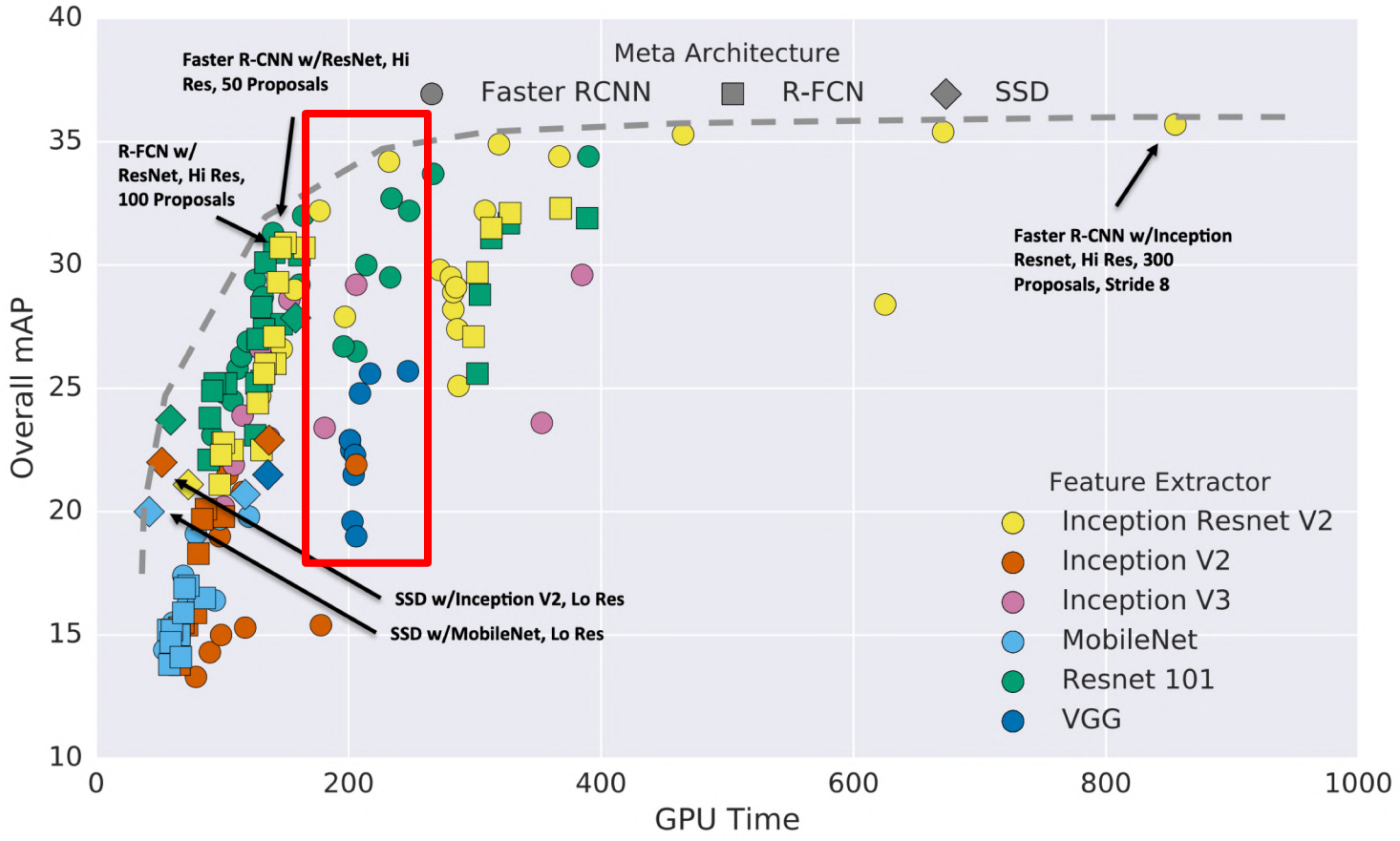
Object Detectors



- One stage detectors are faster, but don't perform well

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

Object Detectors

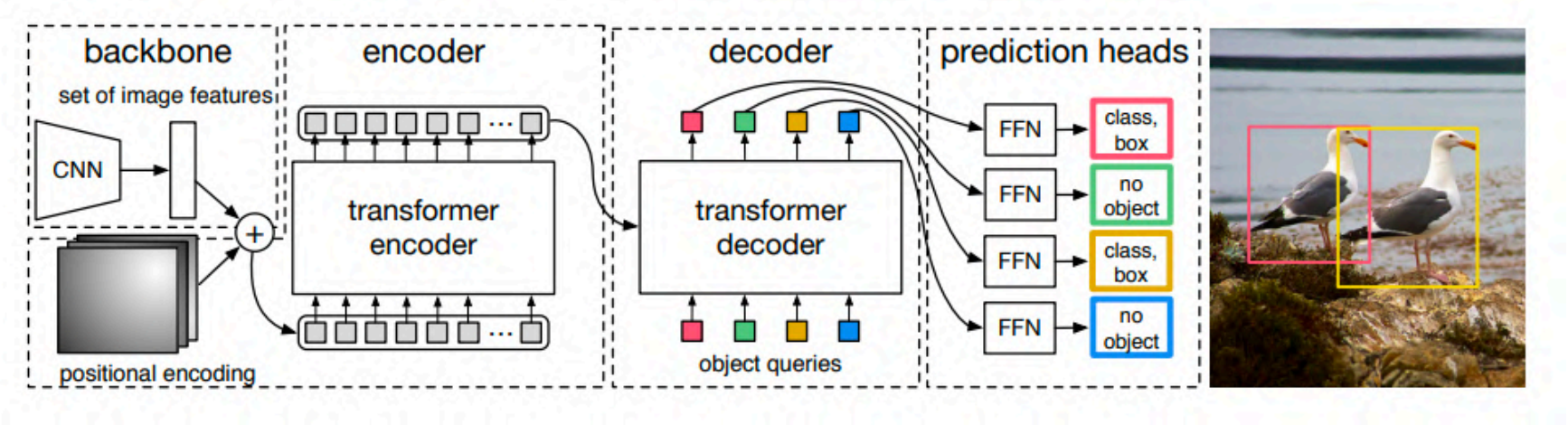


- Bigger backbone improve performance, but slower

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

Object Detectors

- These results are few years old…
- Nowadays the best object detectors use **Transformer based backbone**



Carion, et al. **End-to-End Object Detection with Transformers**

THE END