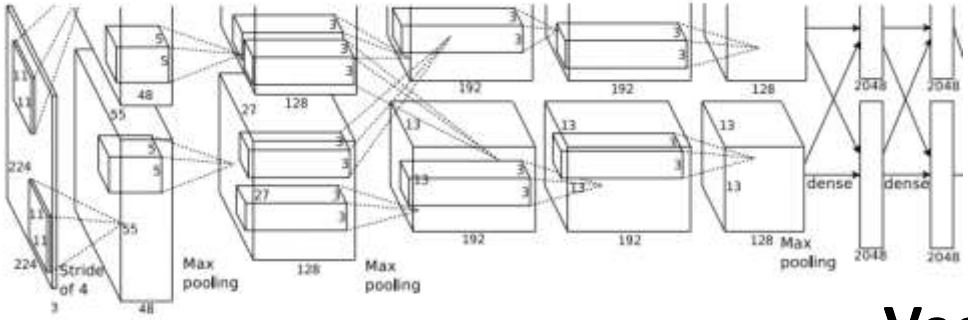
Object Detection

Notes based on
CS231n, Stanford University, and
EECS 498-007 / 598-005, University of Michigan
with permission from Justin Johnson

So far: Image Classification



Vector: 4096

Fully-Connected: 4096 to 1000

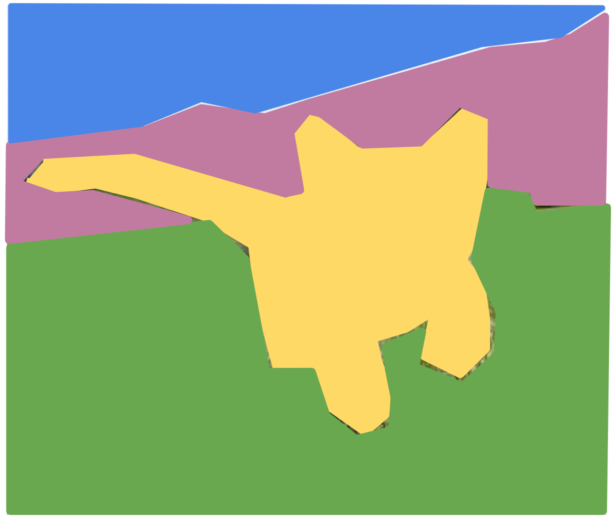
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Computer Vision Tasks

Classification

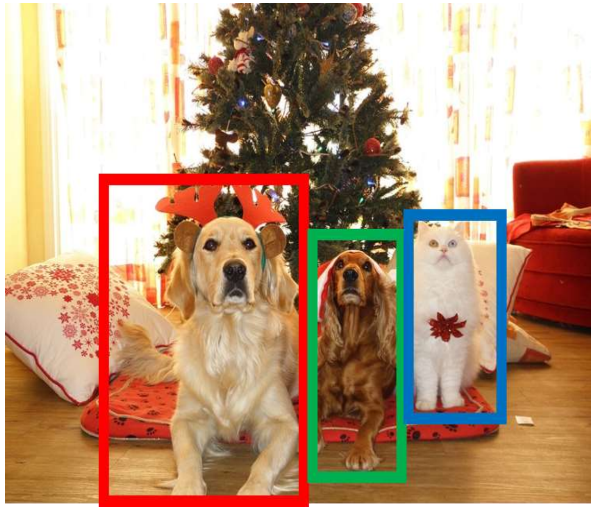
CAT

Semantic Segmentation



GRASS, CAT, TREE, SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

No spatial extent

No objects, just pixels

Multiple Objects

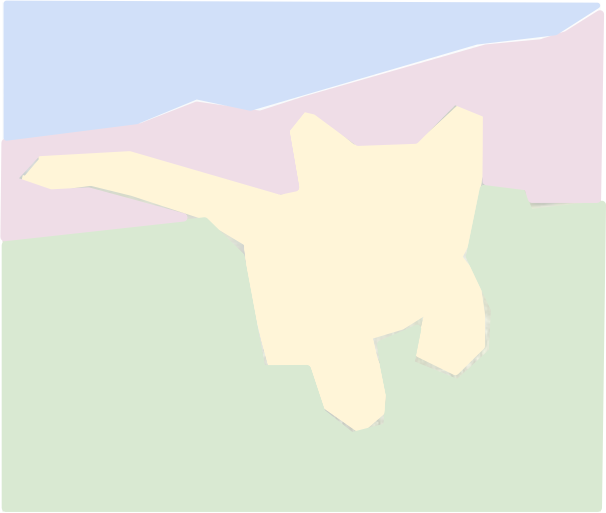
Today: Object Detection

Classification

CAT

No spatial extent

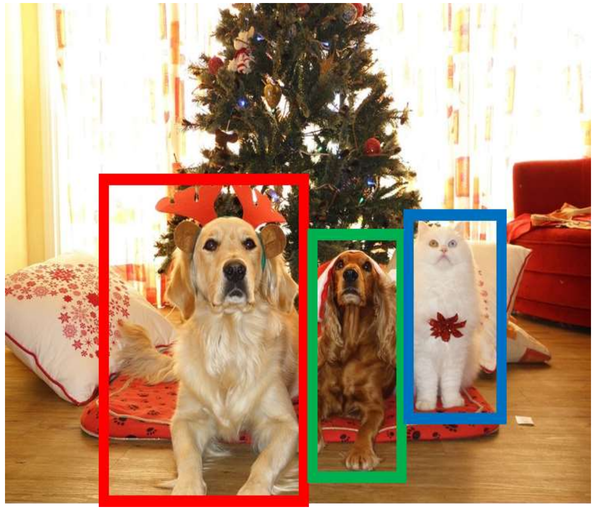
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

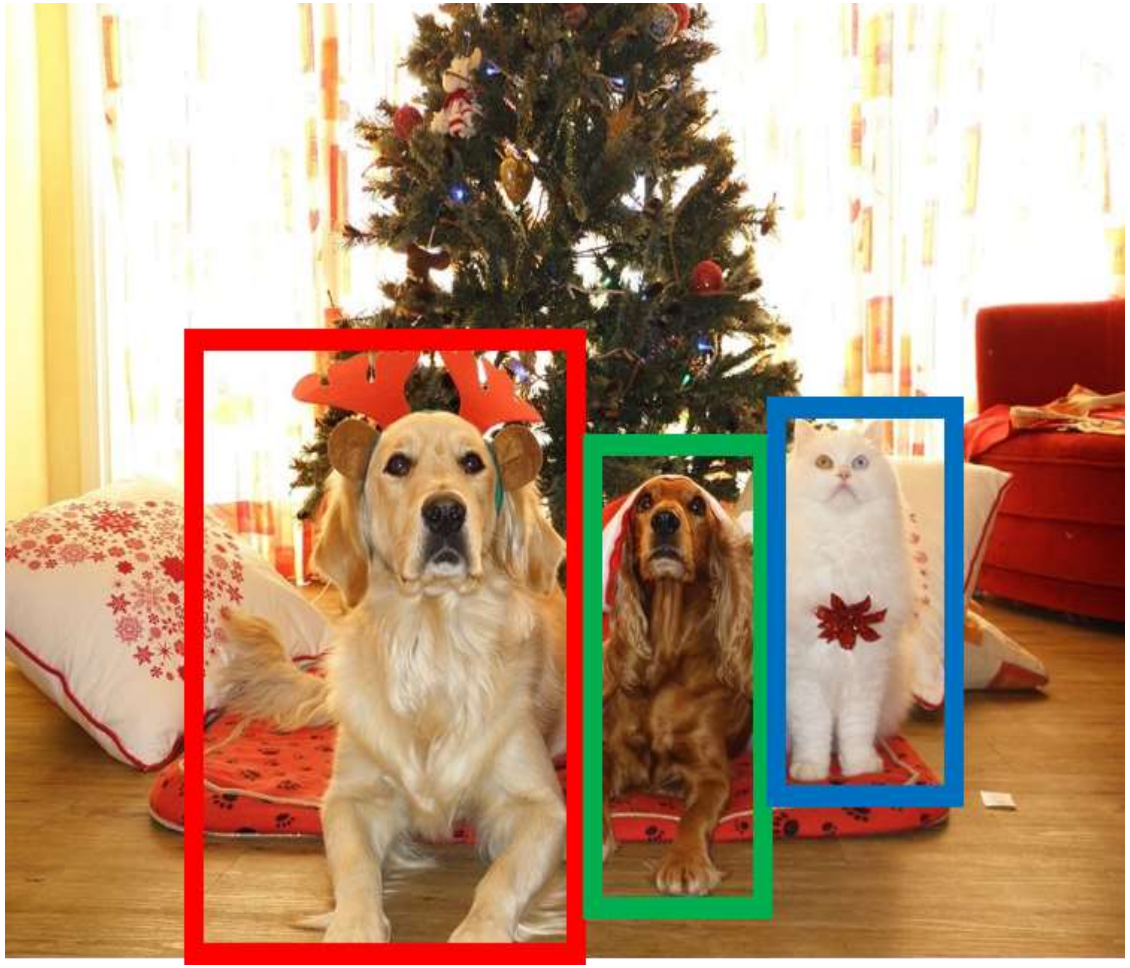
Multiple Objects

Object Detection: Task Definition

Input: Single RGB Image

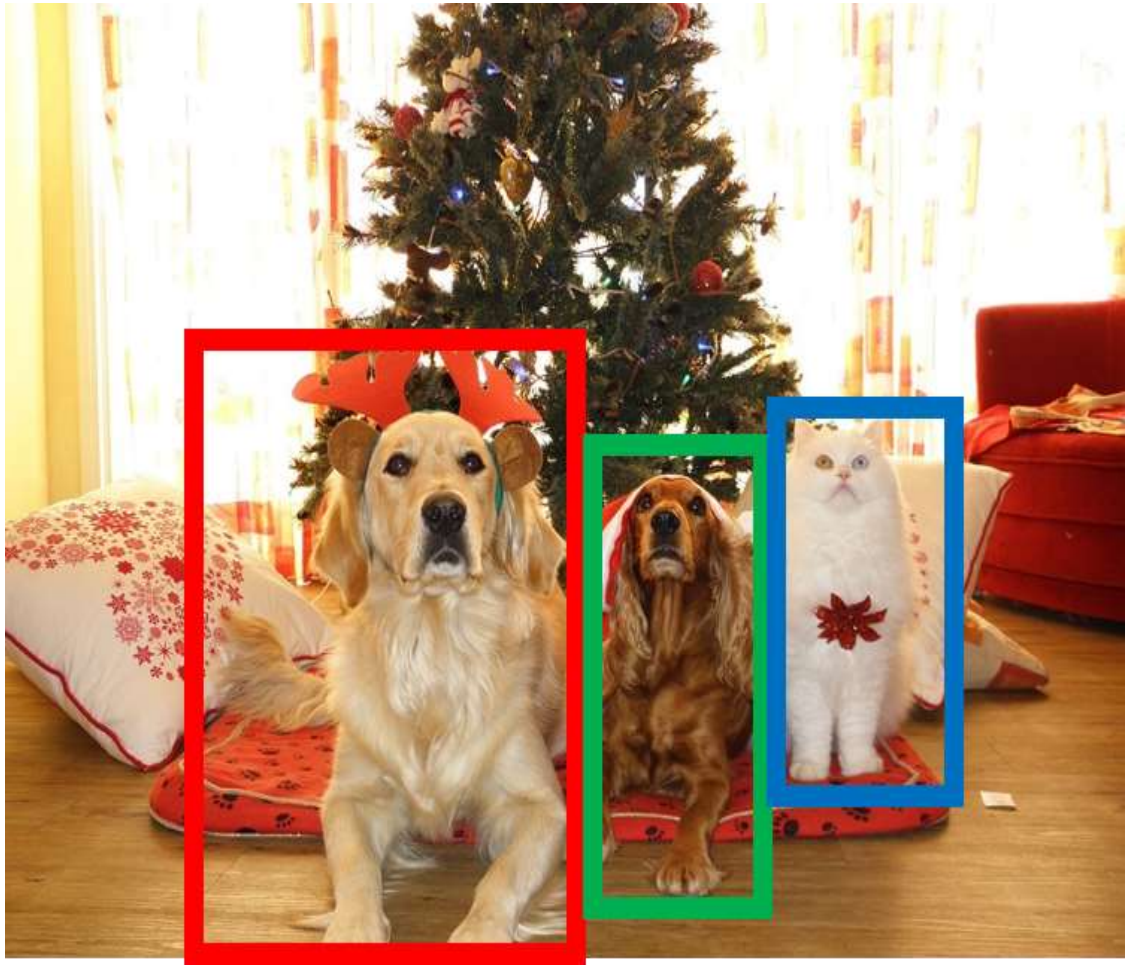
Output: A set of detected objects;
For each object predict:

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

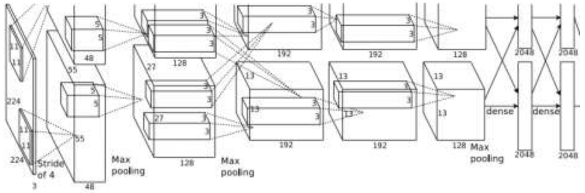


Object Detection: Challenges

- **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

Detecting a single object

"What"

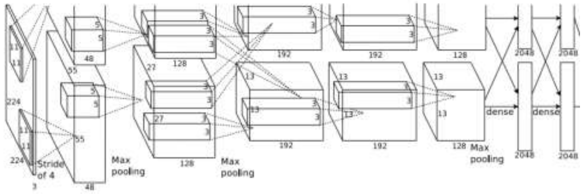
Correct label:
Cat

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

Softmax Loss

Fully Connected:
4096 to 1000

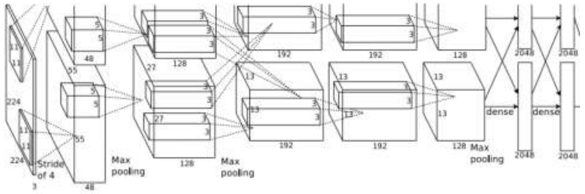
[This image](#) is [CC0 public domain](#)



Vector:
4096

Detecting a single object

“What”



Vector:
4096

Fully Connected:
4096 to 1000

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

Correct label:
Cat

Softmax Loss

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

"What"

Correct label:
Cat

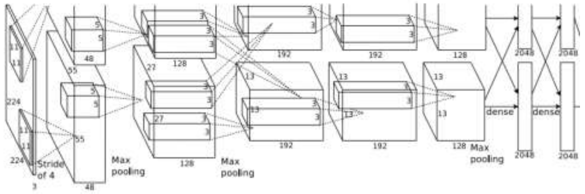
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Fully Connected:
4096 to 1000

Softmax Loss

Weighted Sum

Loss



Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Treat localization as a regression problem!

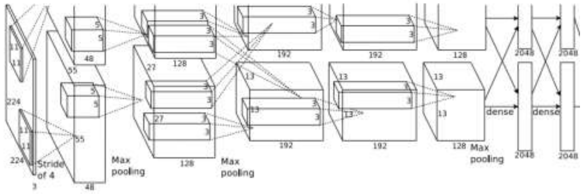
"Where"

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

Detecting a single object

“What”

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

...

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’)

“Where”

Detecting a single object

"What"

Correct label:
Cat

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

Softmax Loss

Multitask Loss

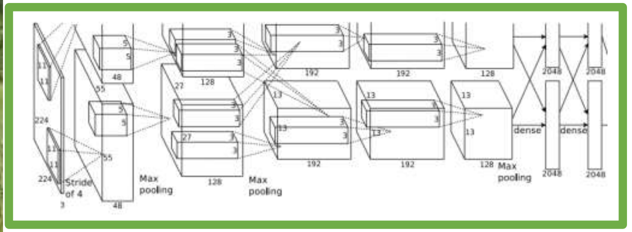
Weighted Sum

Loss

Often pretrained on ImageNet (Transfer learning)

Fully Connected: 4096 to 1000

This image is CC0 public domain



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!

"Where"

Detecting a single object

"What"

Correct label:
Cat

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

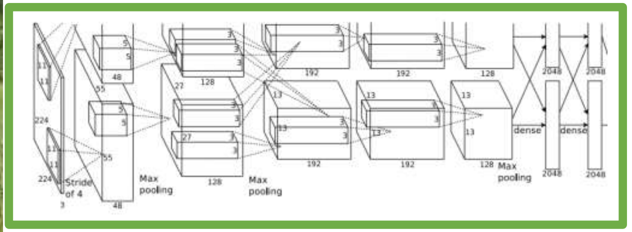
Softmax Loss

Multitask Loss

Often pretrained on ImageNet (Transfer learning)

Fully Connected:
4096 to 1000

This image is CC0 public domain



Vector:
4096

Weighted Sum → **Loss**

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x’, y’, w’, h’)

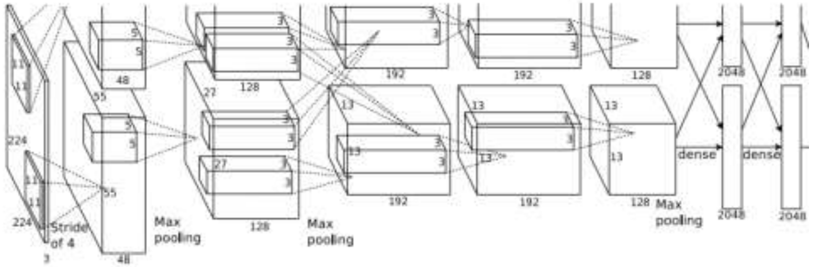
Treat localization as a regression problem!

Problem: Images can have more than one object!

"Where"

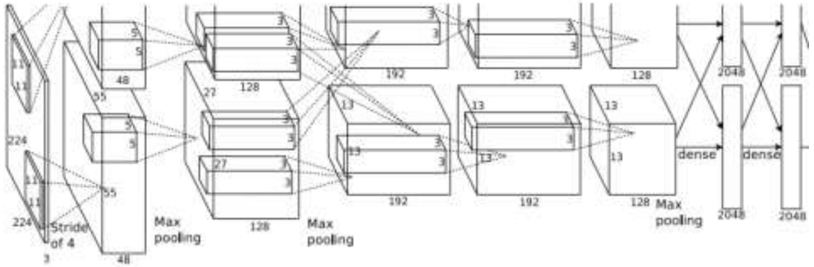
Detecting Multiple Objects

Need different numbers of outputs per image



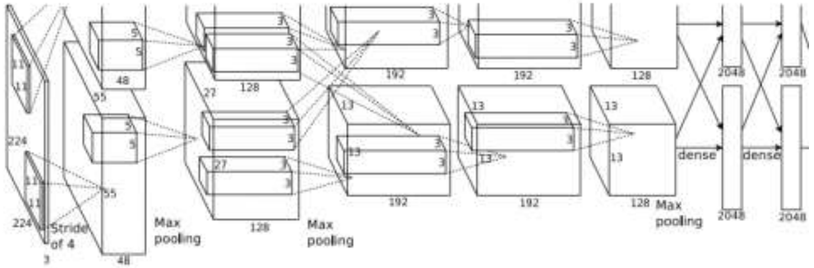
CAT: (x, y, w, h)

4 numbers



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

12 numbers

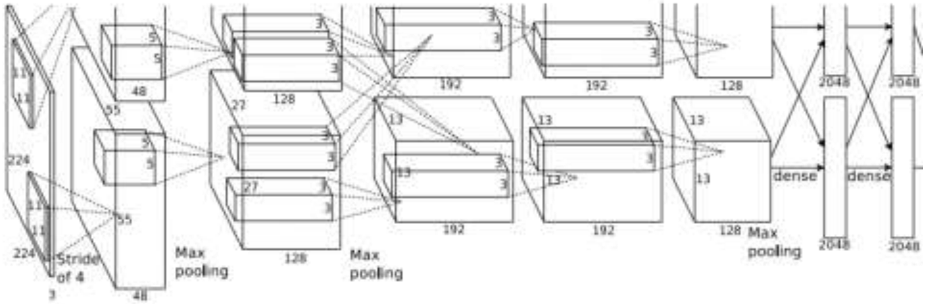


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detecting Multiple Objects: **Sliding Window**

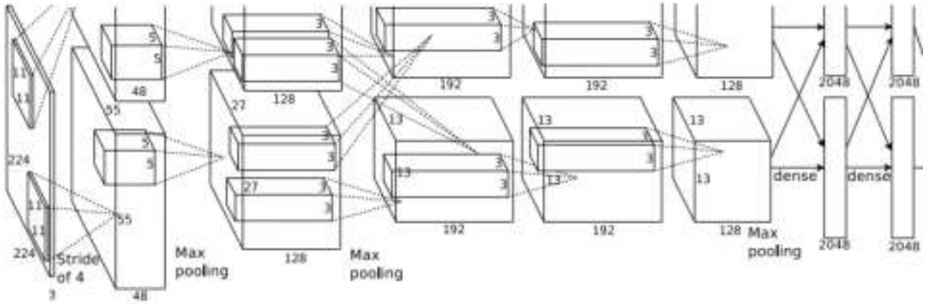
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: **Sliding Window**

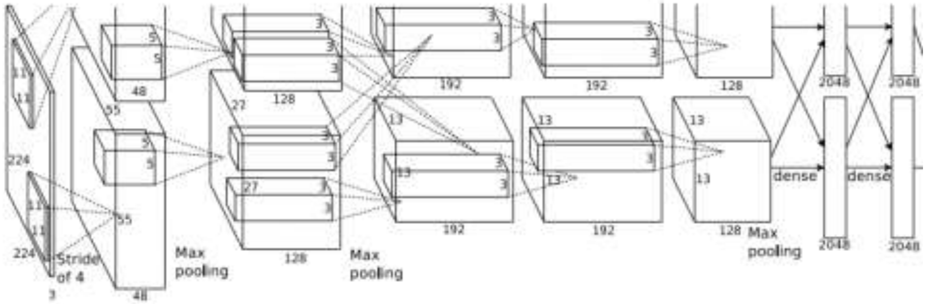
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

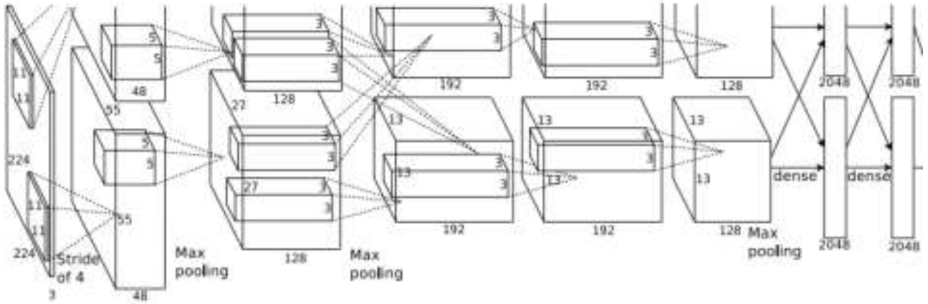
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



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

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)

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?

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:

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

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

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)

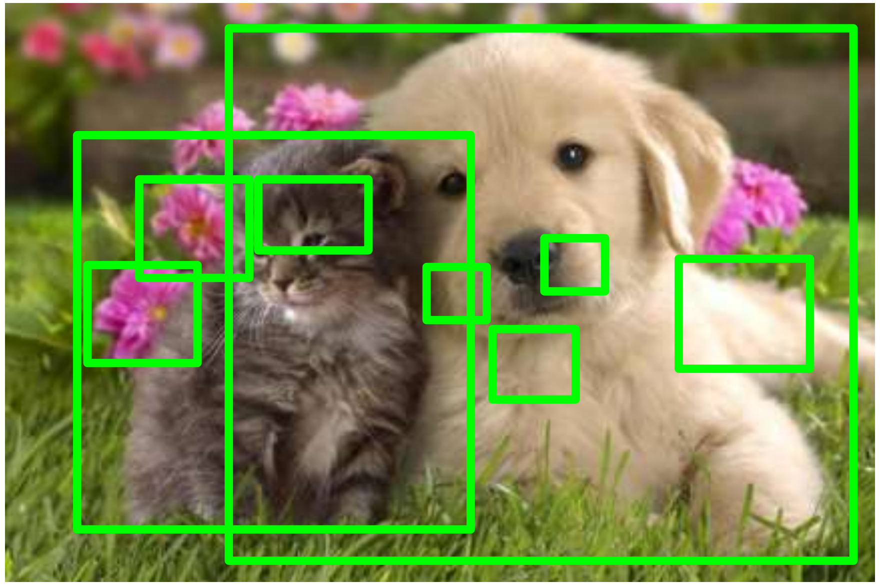
Total possible boxes:

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

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

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.

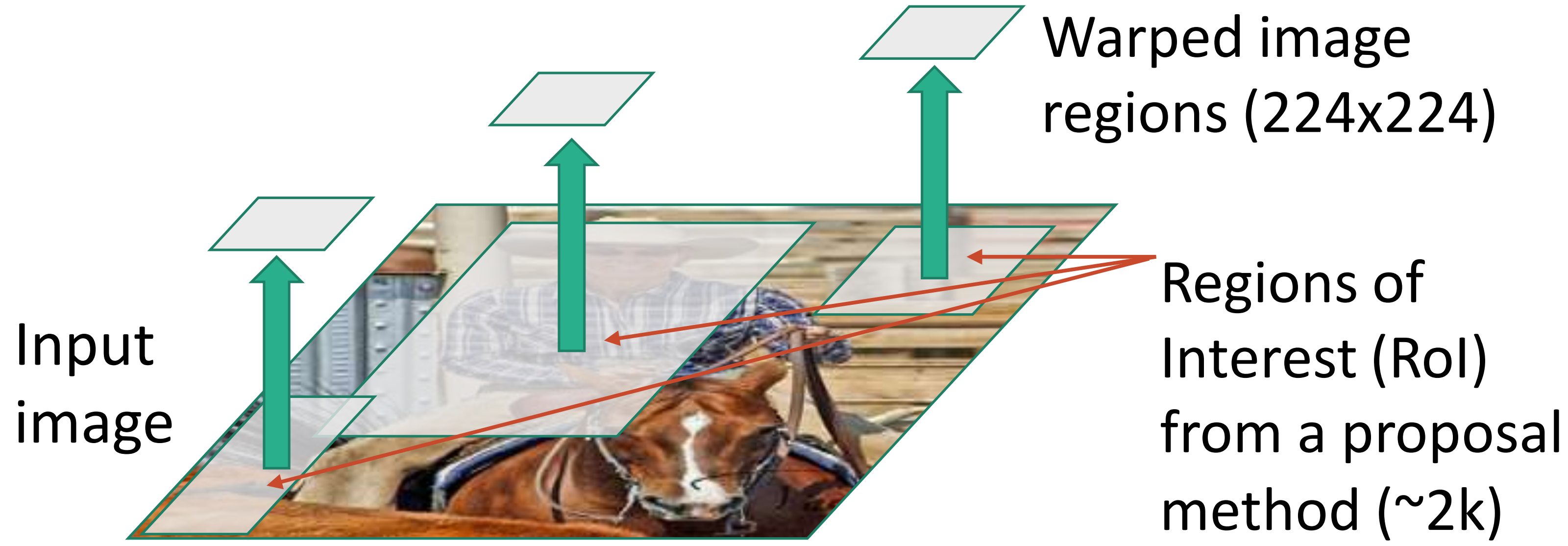
R-CNN: Region-Based CNN

Input
image

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

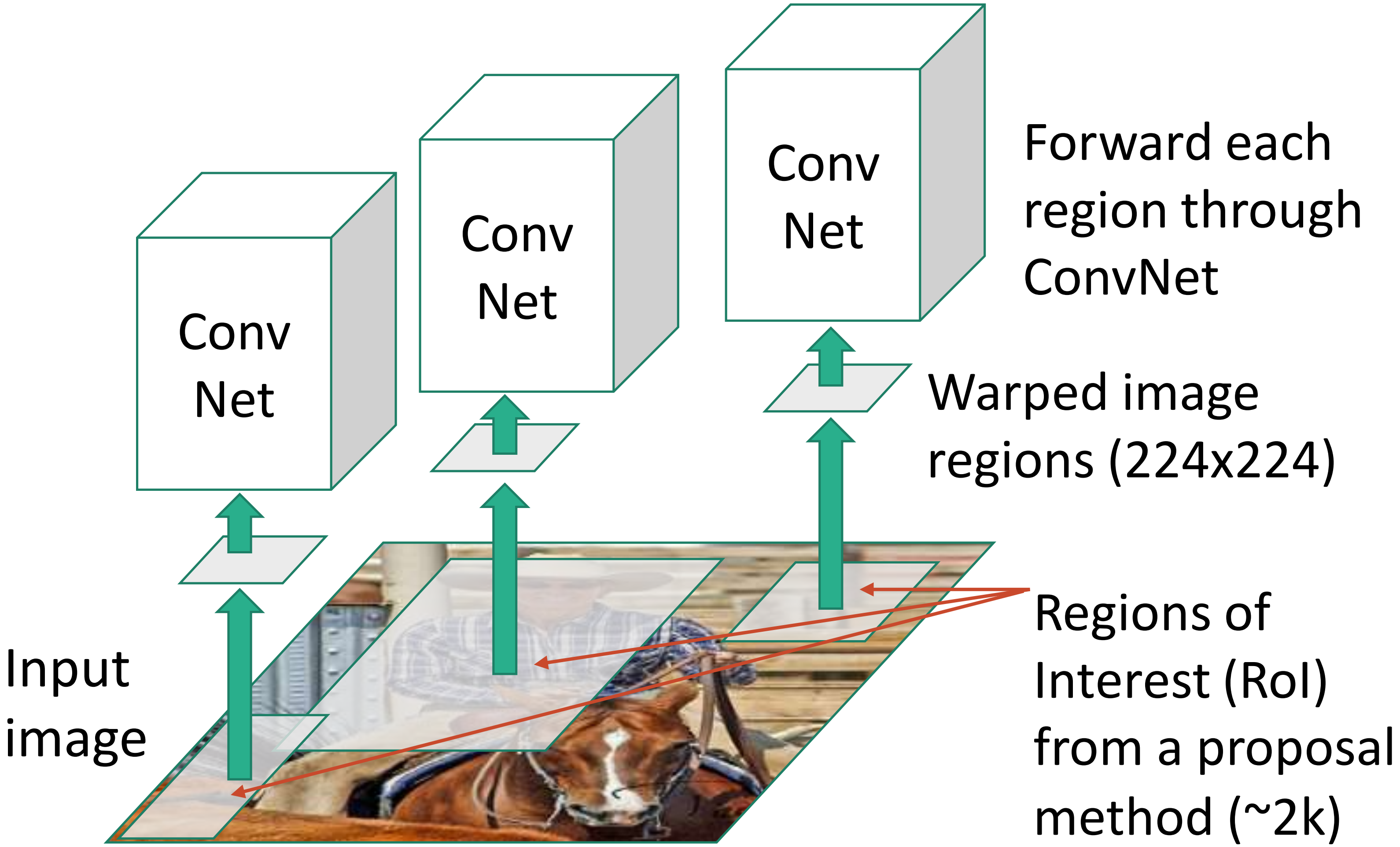
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

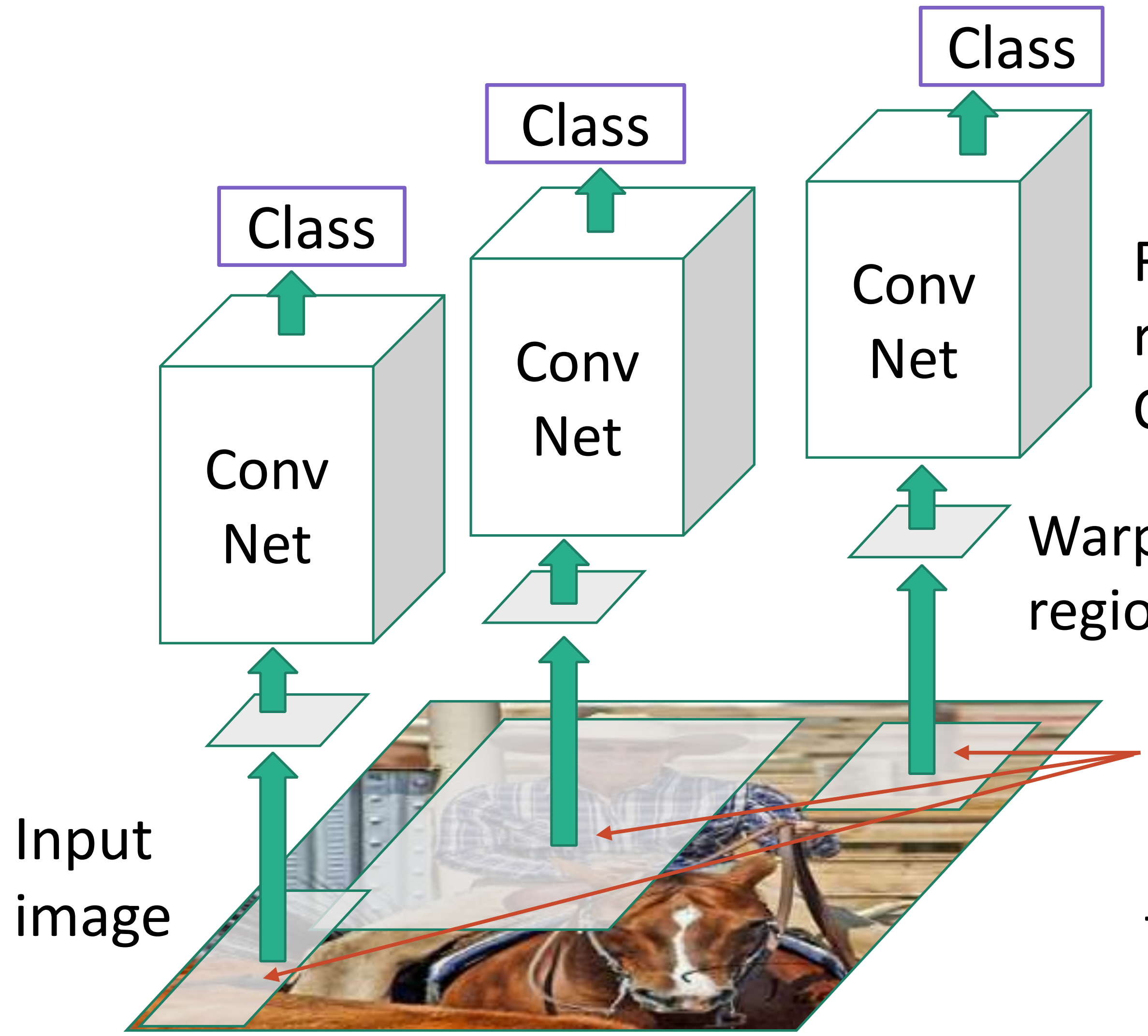
R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN: Region-Based CNN

Classify each region



Forward each region through ConvNet

Warped image regions (224x224)

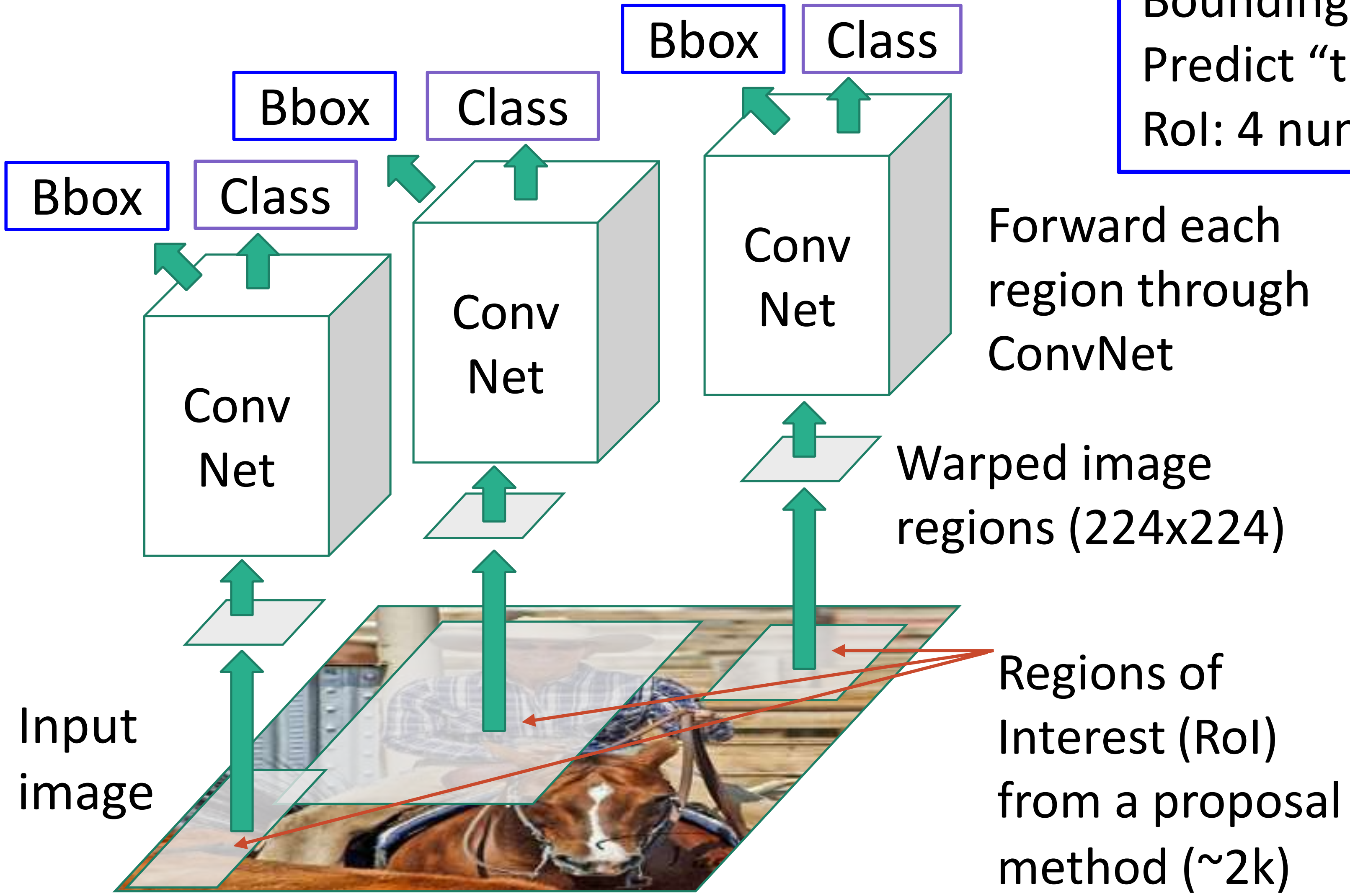
Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN: Region-Based CNN

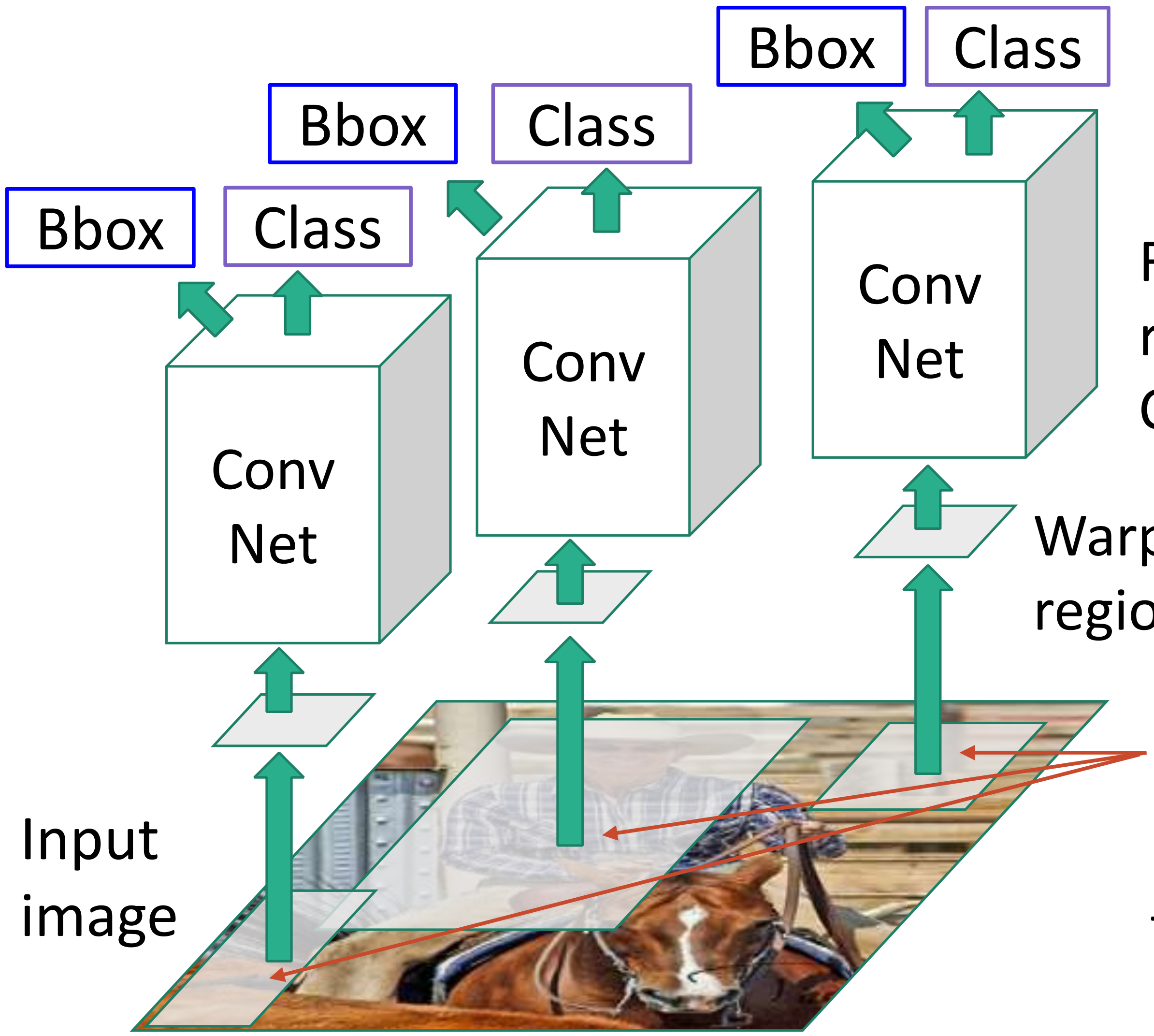
Classify each region

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



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)

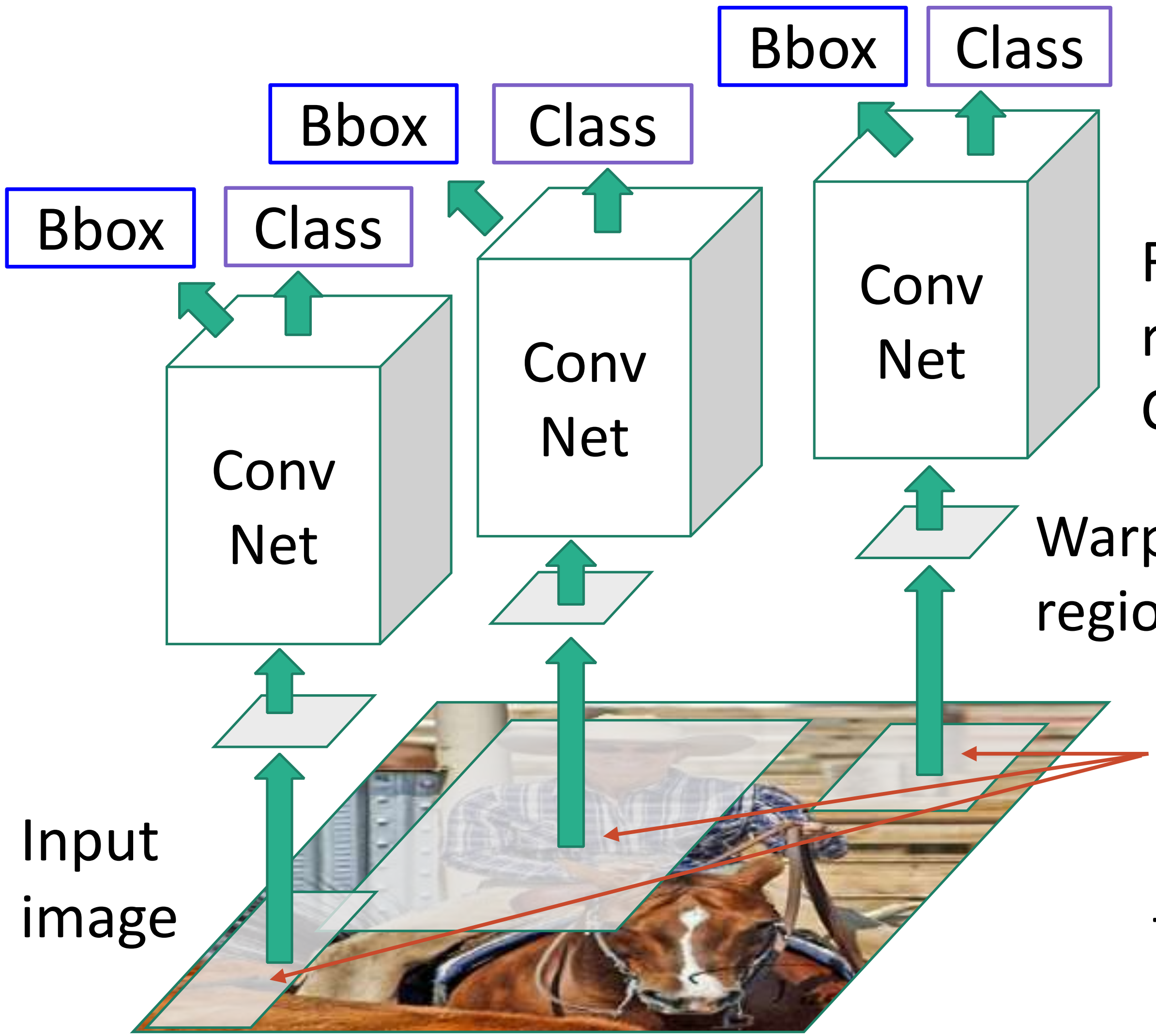
Region proposal: (p_x, p_y, p_h, p_w)
Transform: (t_x, t_y, t_h, t_w)
Output box: (b_x, b_y, b_h, b_w)

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.

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)

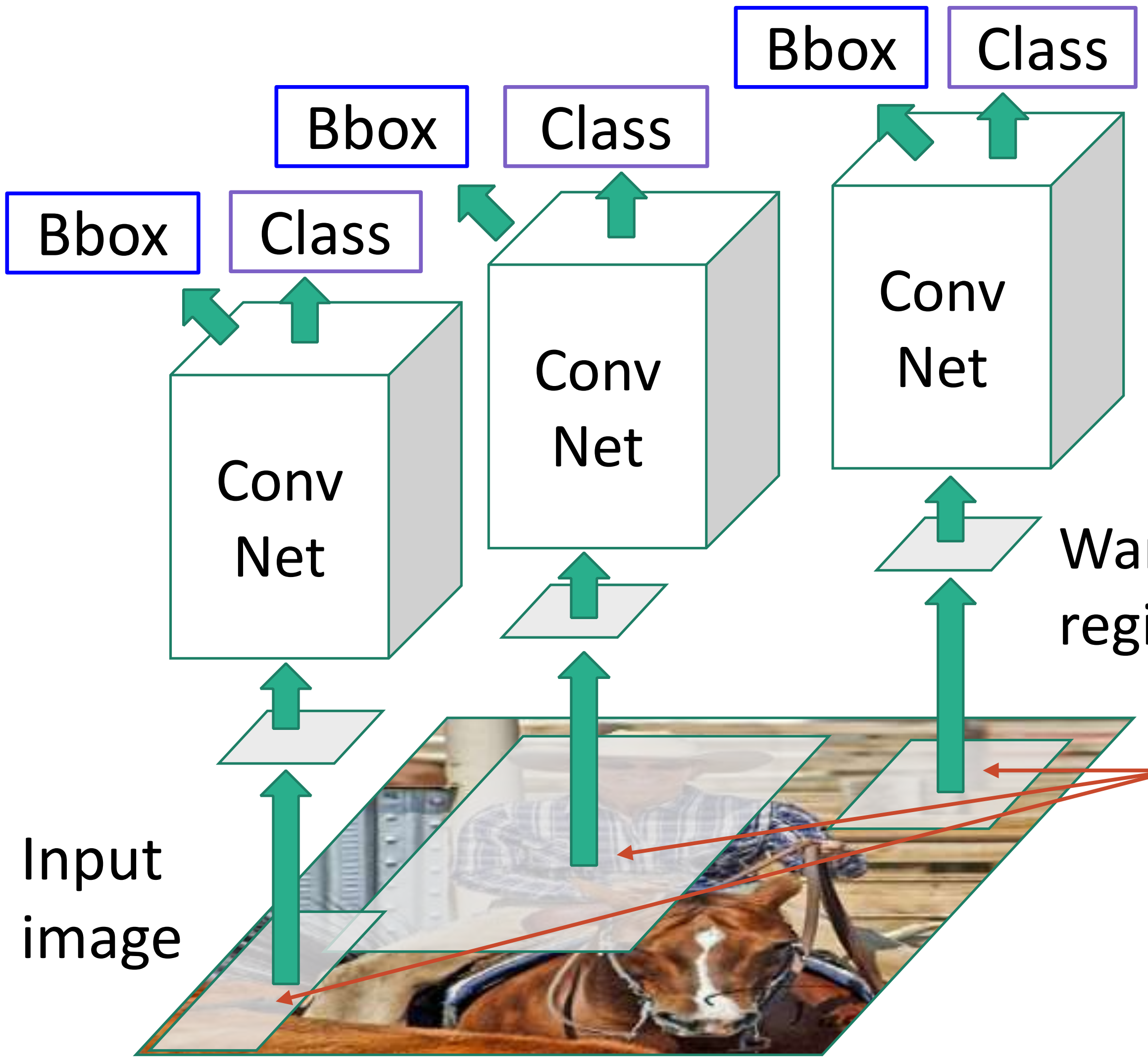
Forward each region through ConvNet

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

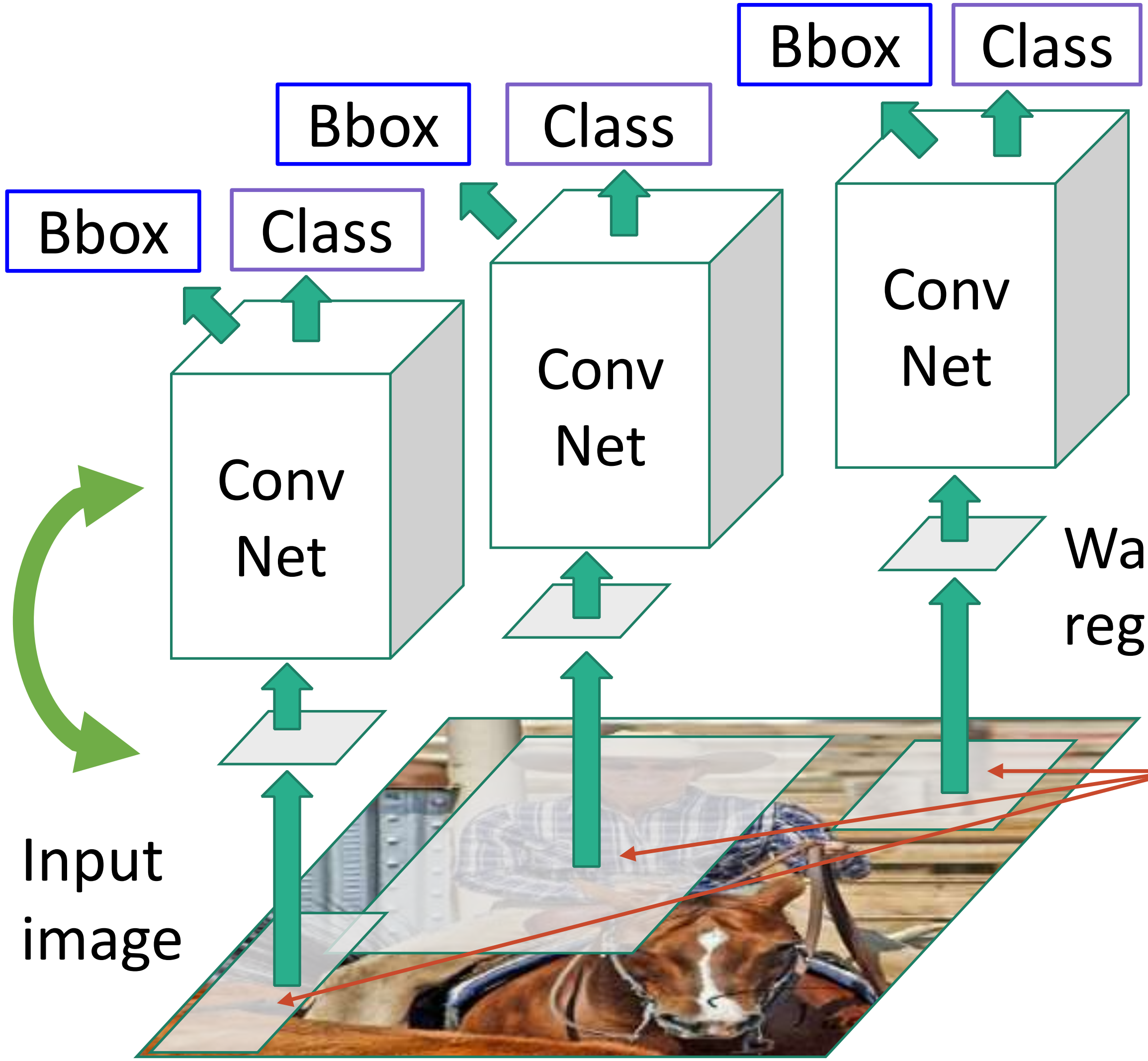
Warped image regions (224x224)

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.

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)

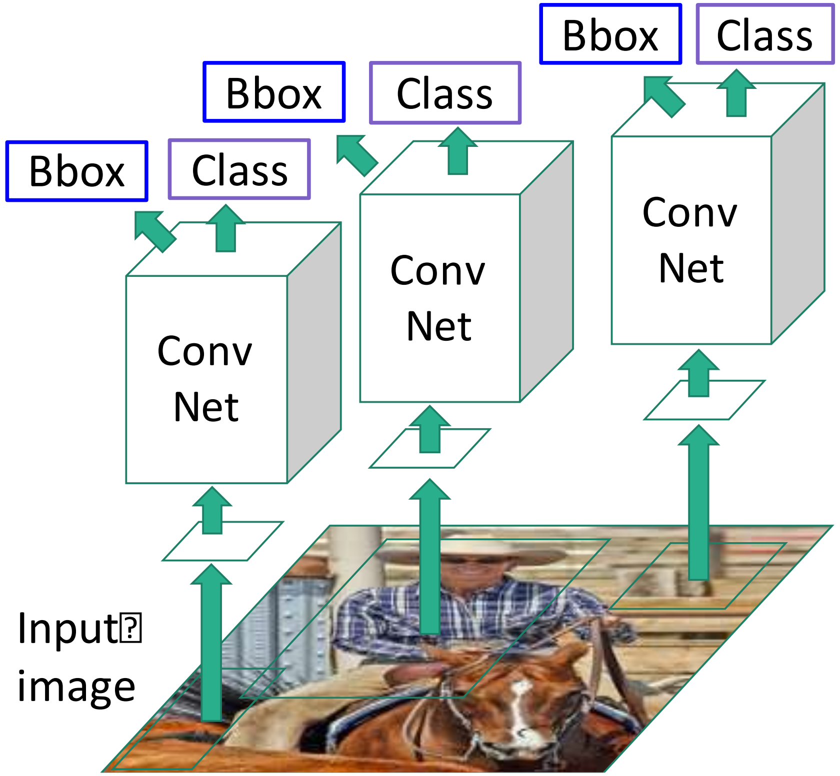
Problem: Very slow! Need to do ~2k forward passes for each image!

Solution: Run CNN *before* warping!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

"Slow" R-CNN

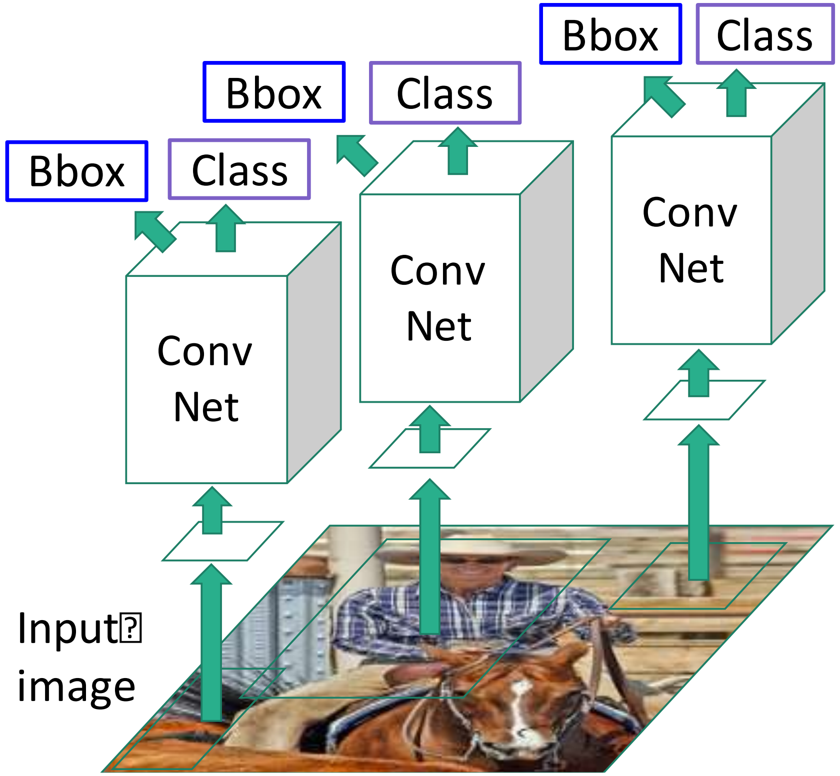
Process each region independently



Fast R-CNN

“Slow” R-CNN

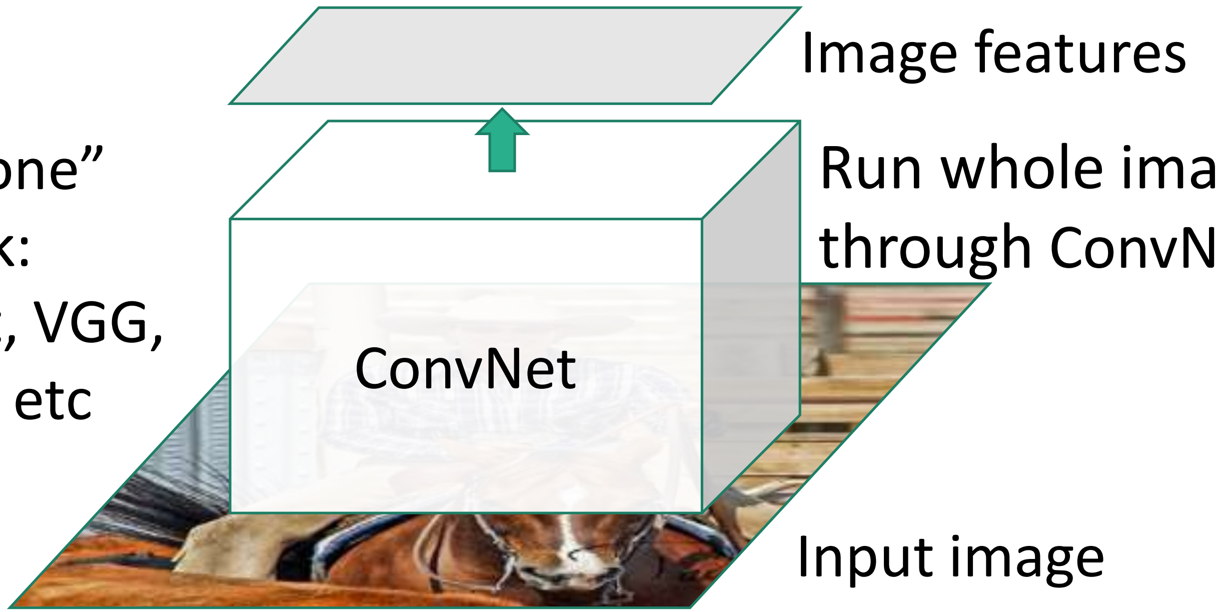
Process each region independently



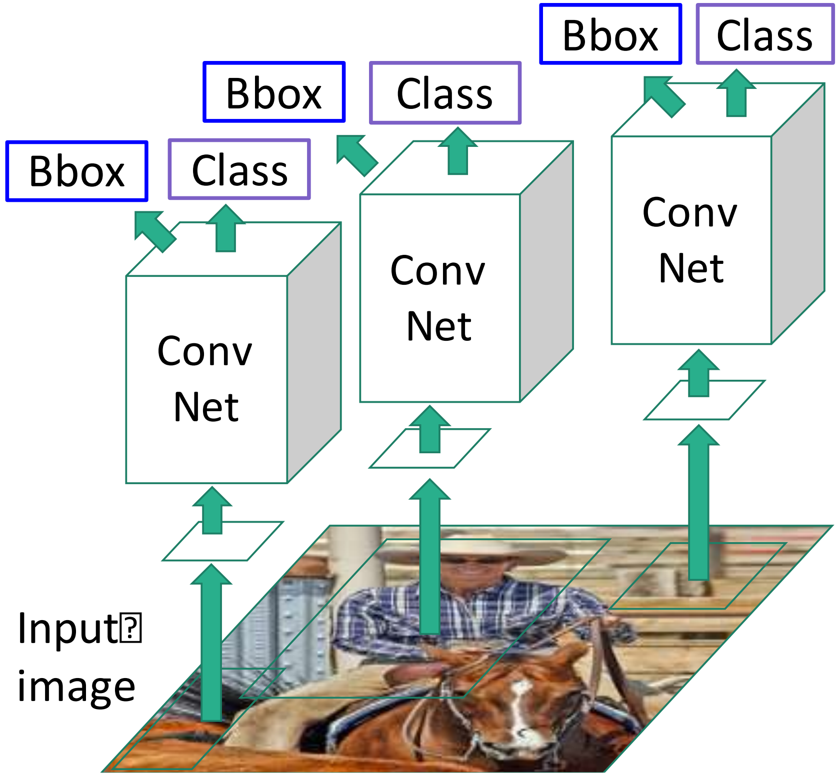
Input image

Fast R-CNN

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

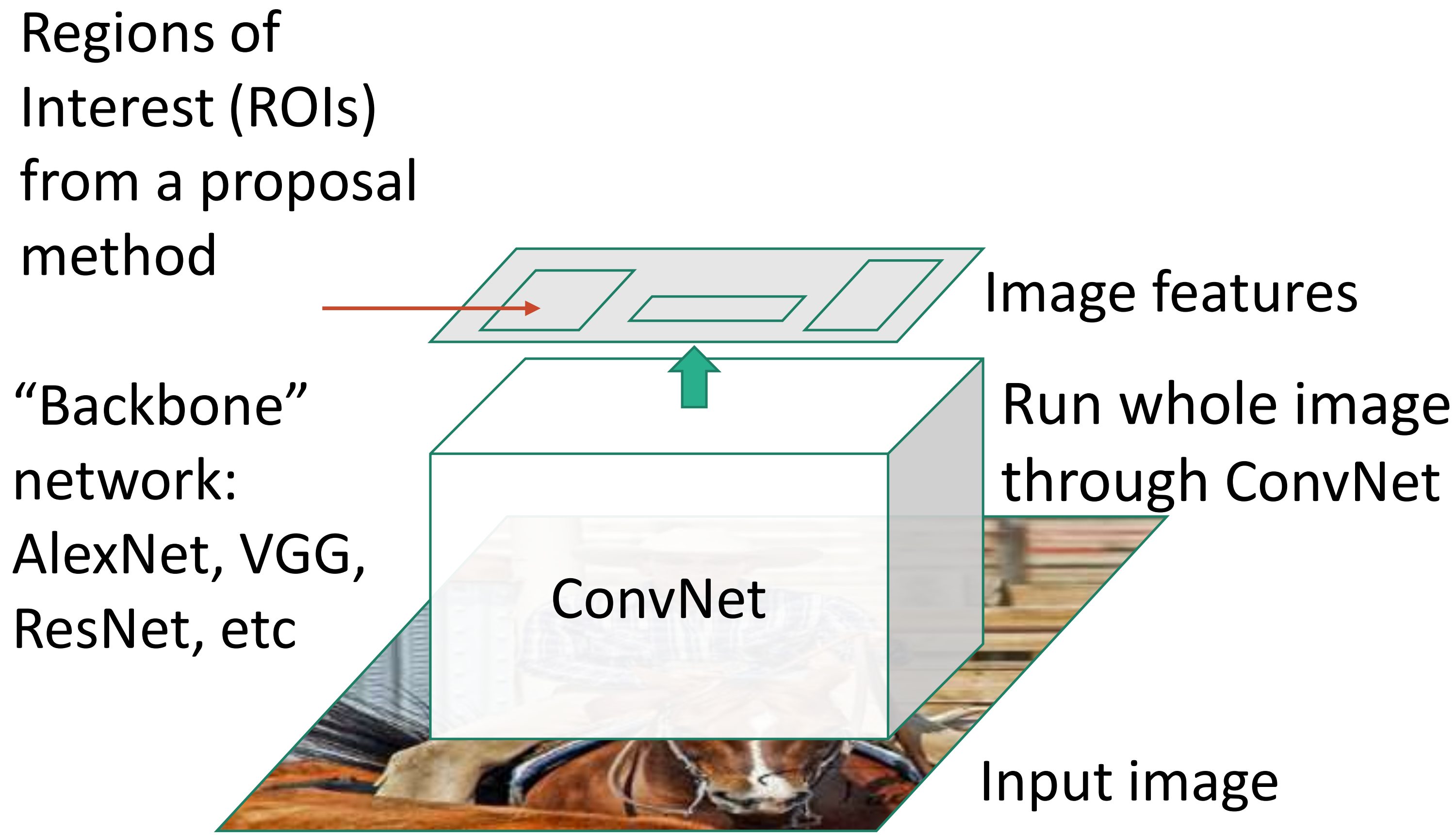


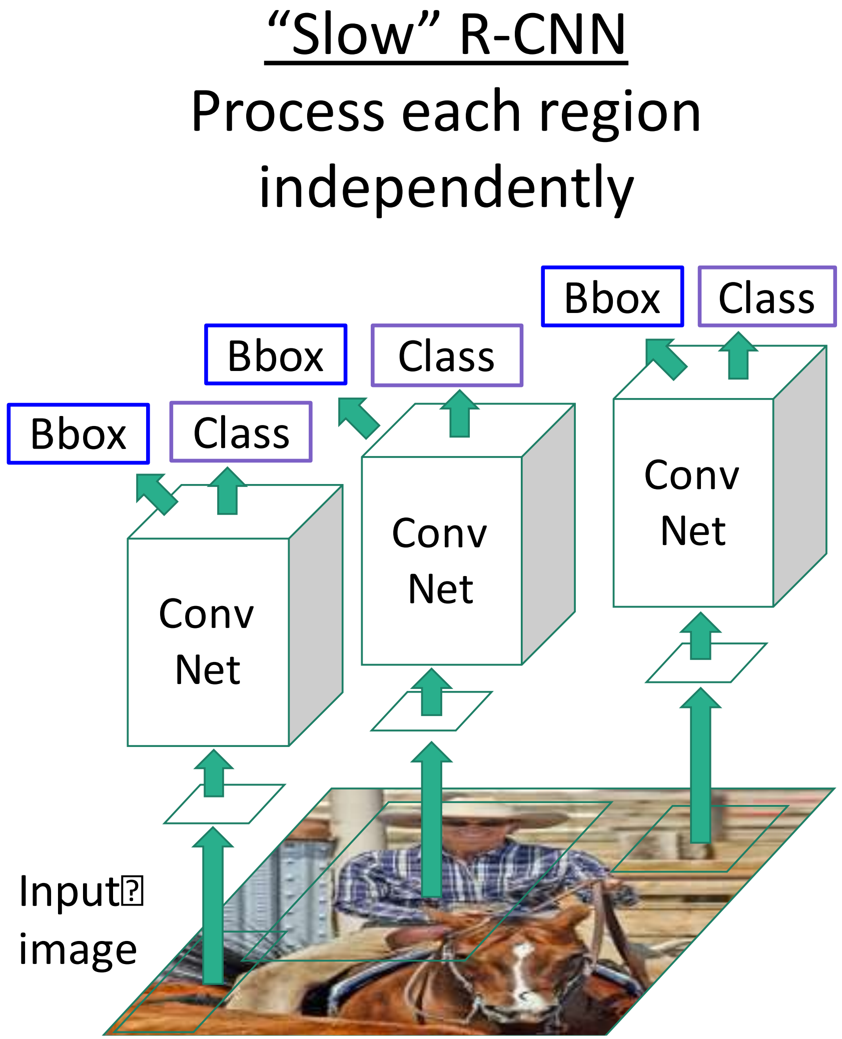
"Slow" R-CNN
Process each region independently



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

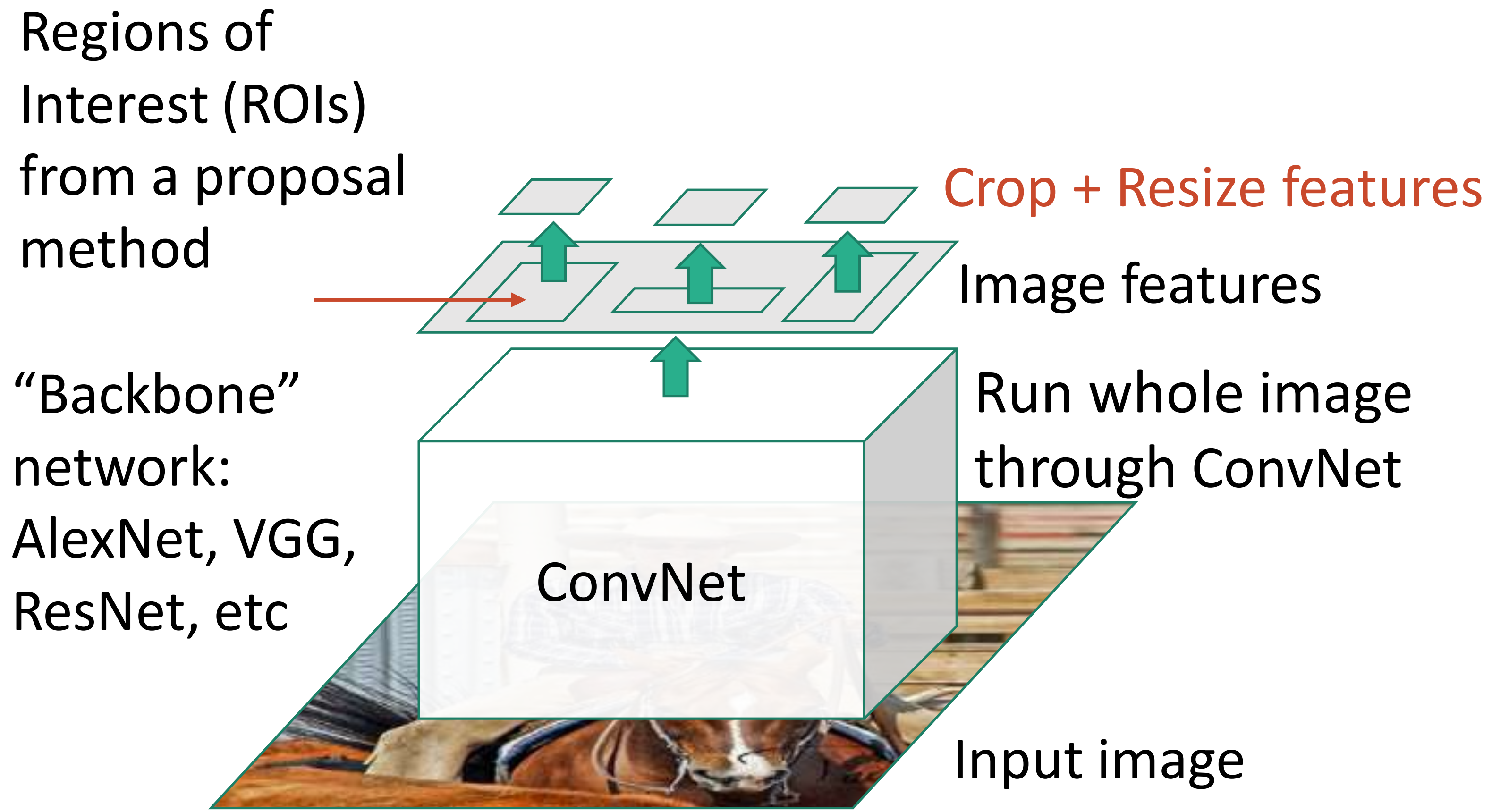
Fast R-CNN

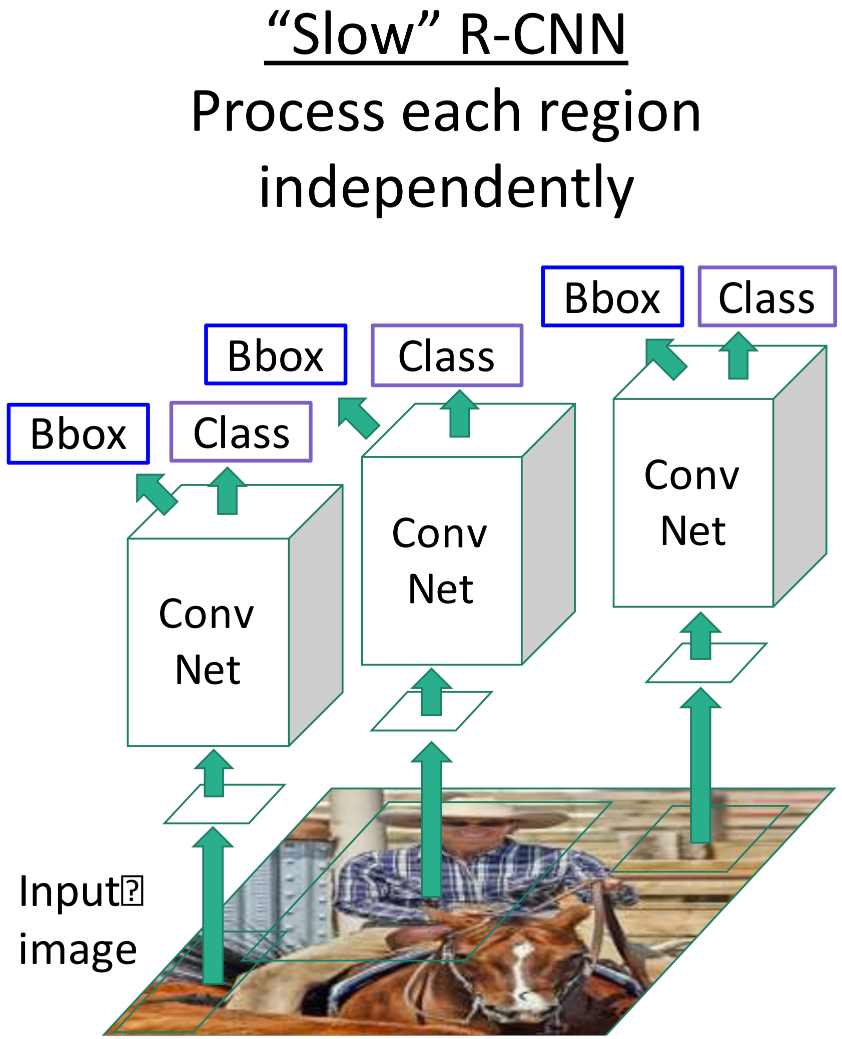




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



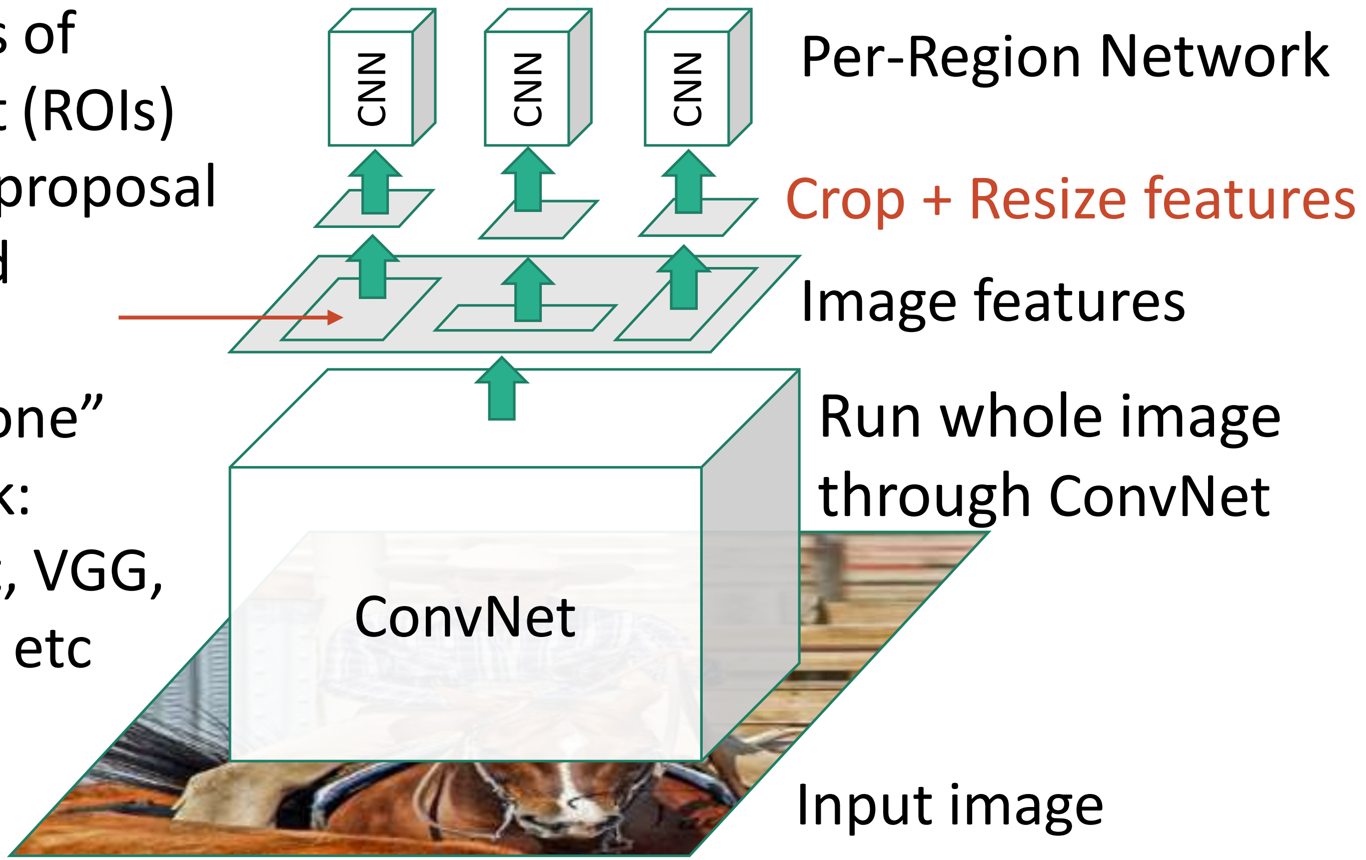


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

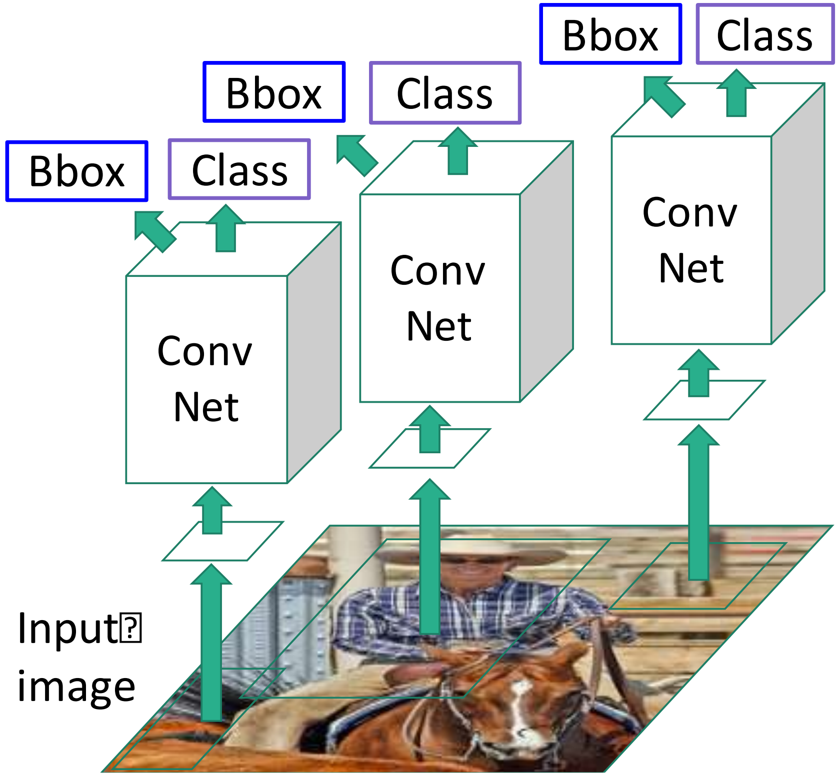
Fast R-CNN

Regions of Interest (ROIs) from a proposal method

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

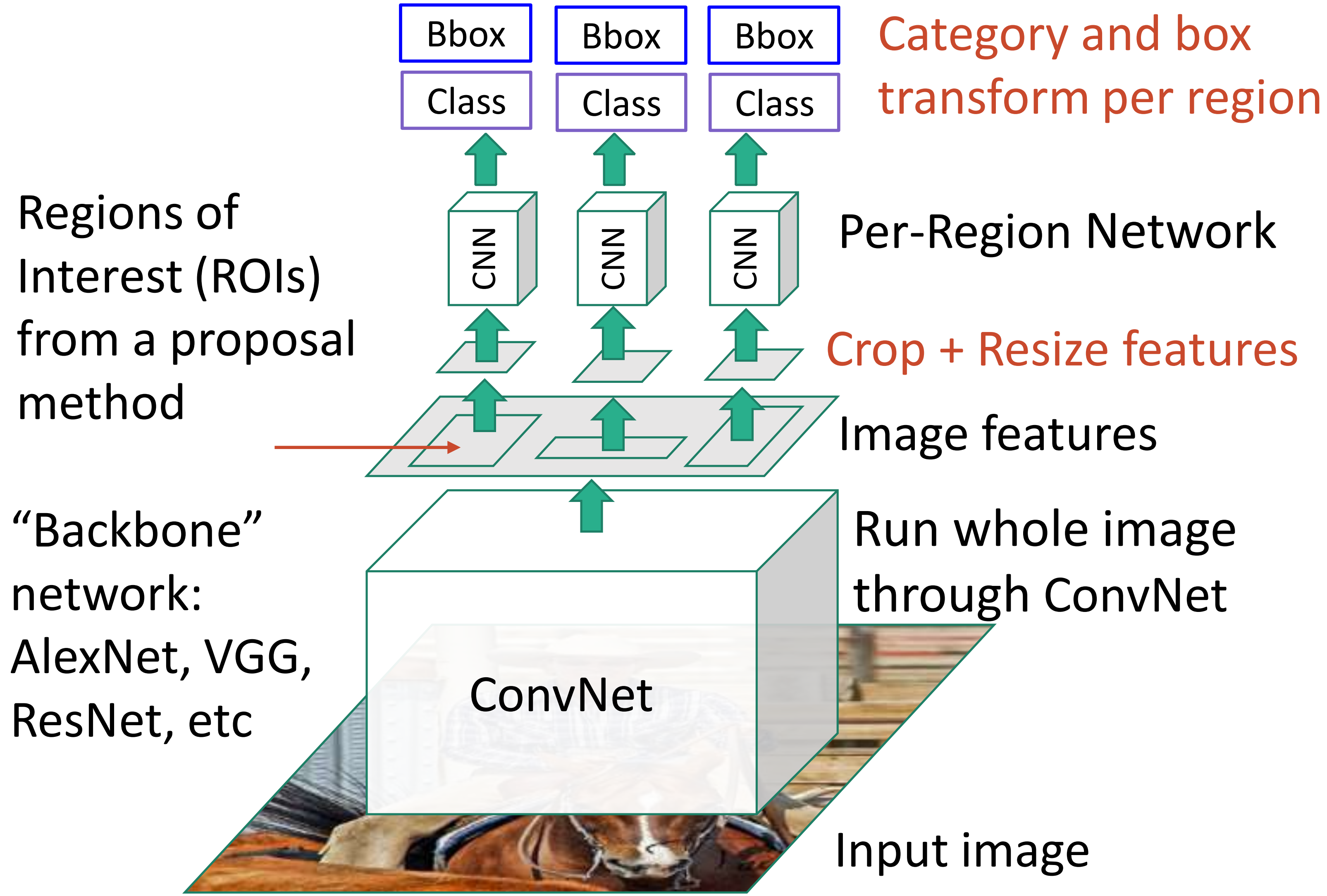


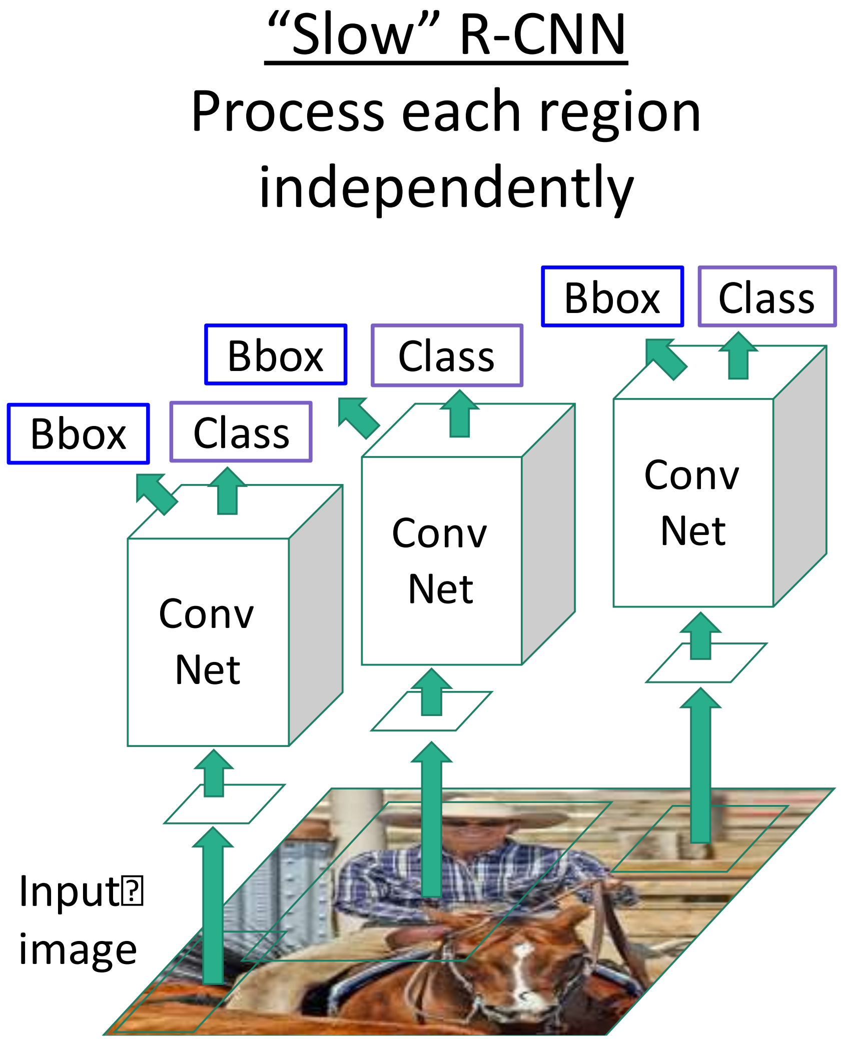
"Slow" R-CNN
Process each region independently



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

Fast R-CNN





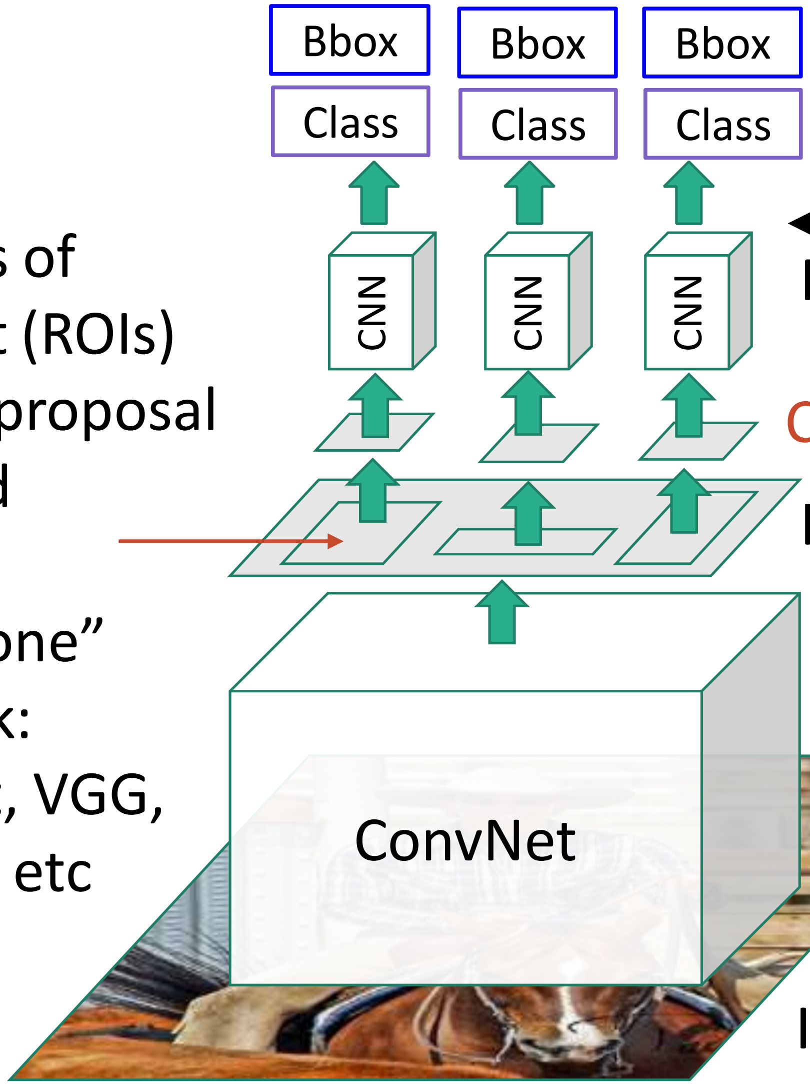
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; [source](). Reproduced with permission.

Fast R-CNN

Category and box transform per region

Per-Region network is relatively lightweight

Regions of Interest (ROIs) from a proposal method



Per-Region Network

Crop + Resize features

Image features

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

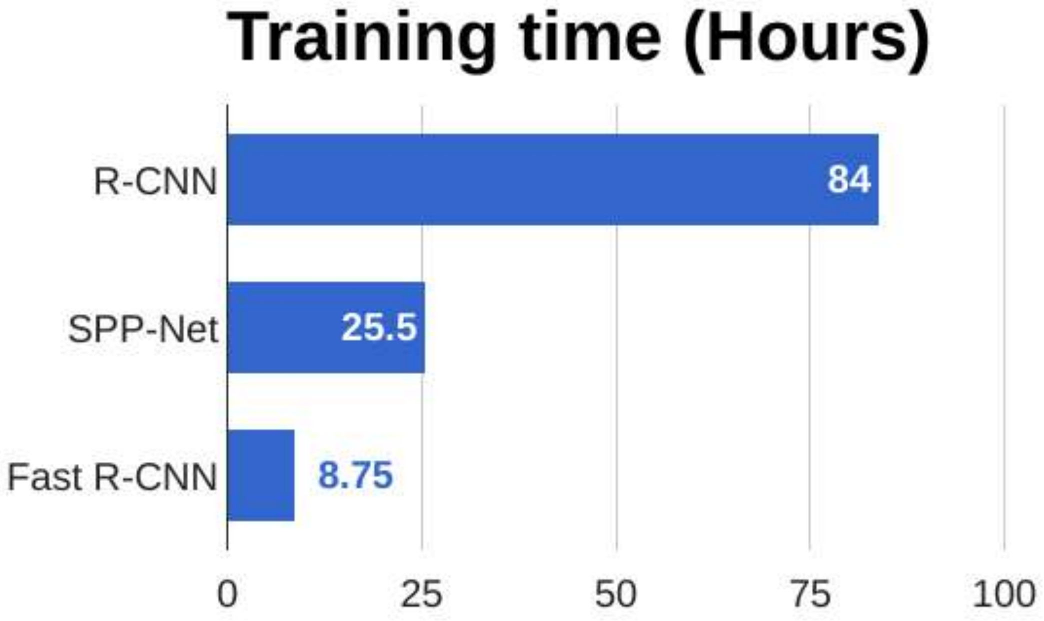
Run whole image through ConvNet

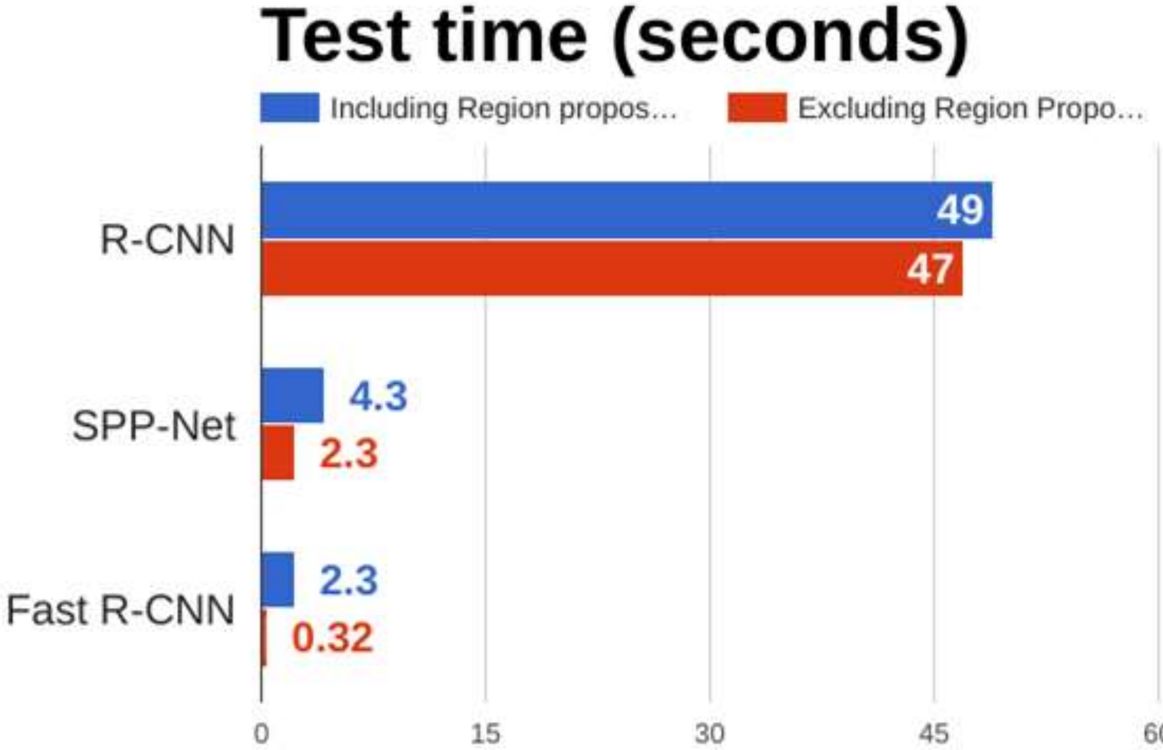
Most of the computation happens in backbone network; this saves work for overlapping region proposals

Input image

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

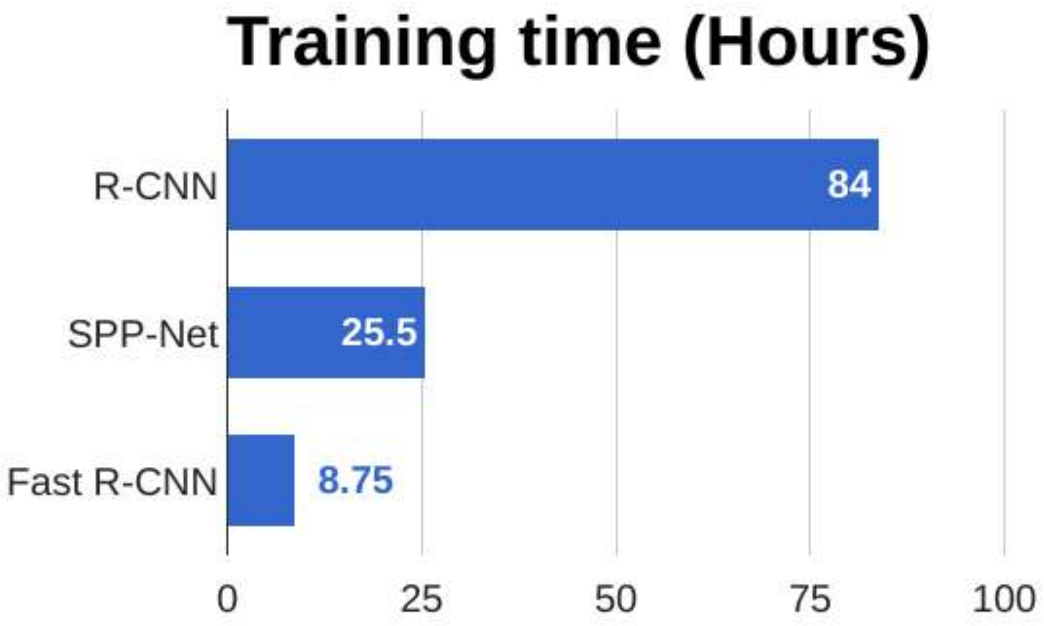
Fast R-CNN vs “Slow” R-CNN

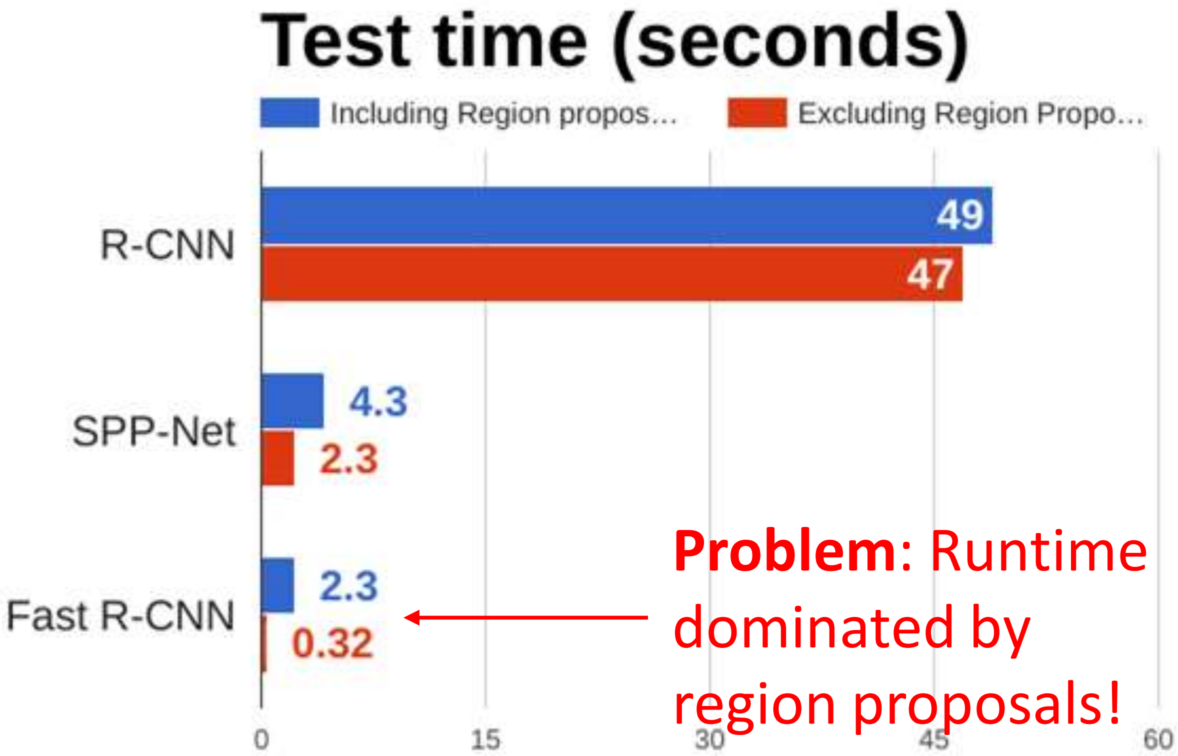




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs “Slow” R-CNN

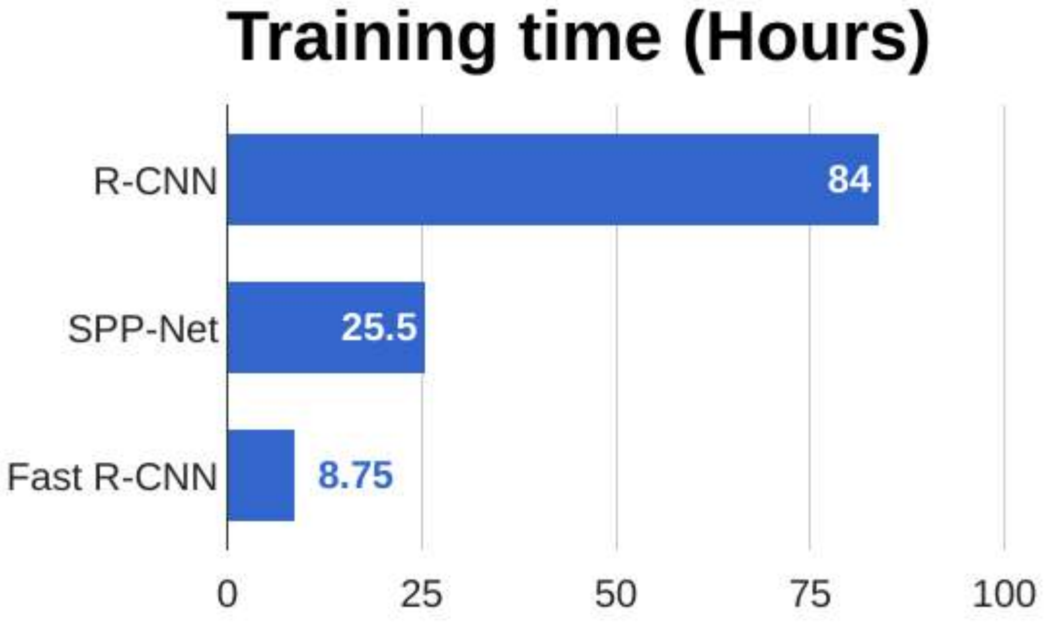


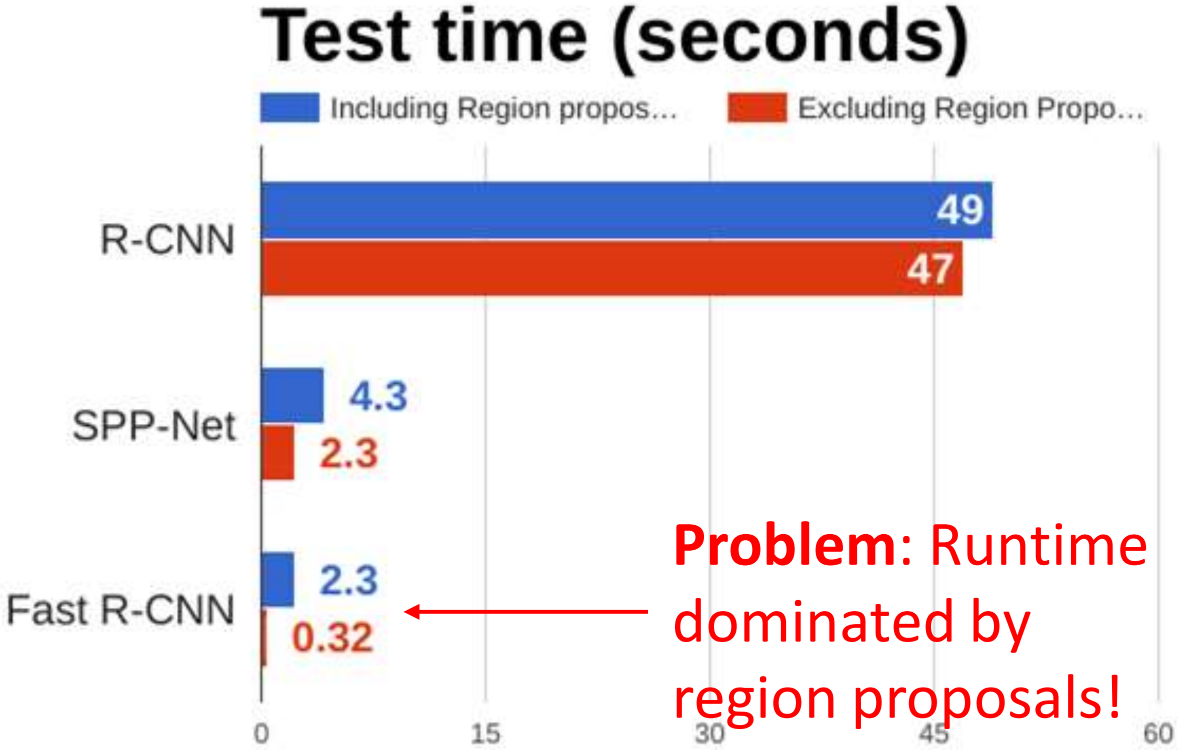


Problem: Runtime dominated by region proposals!

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!

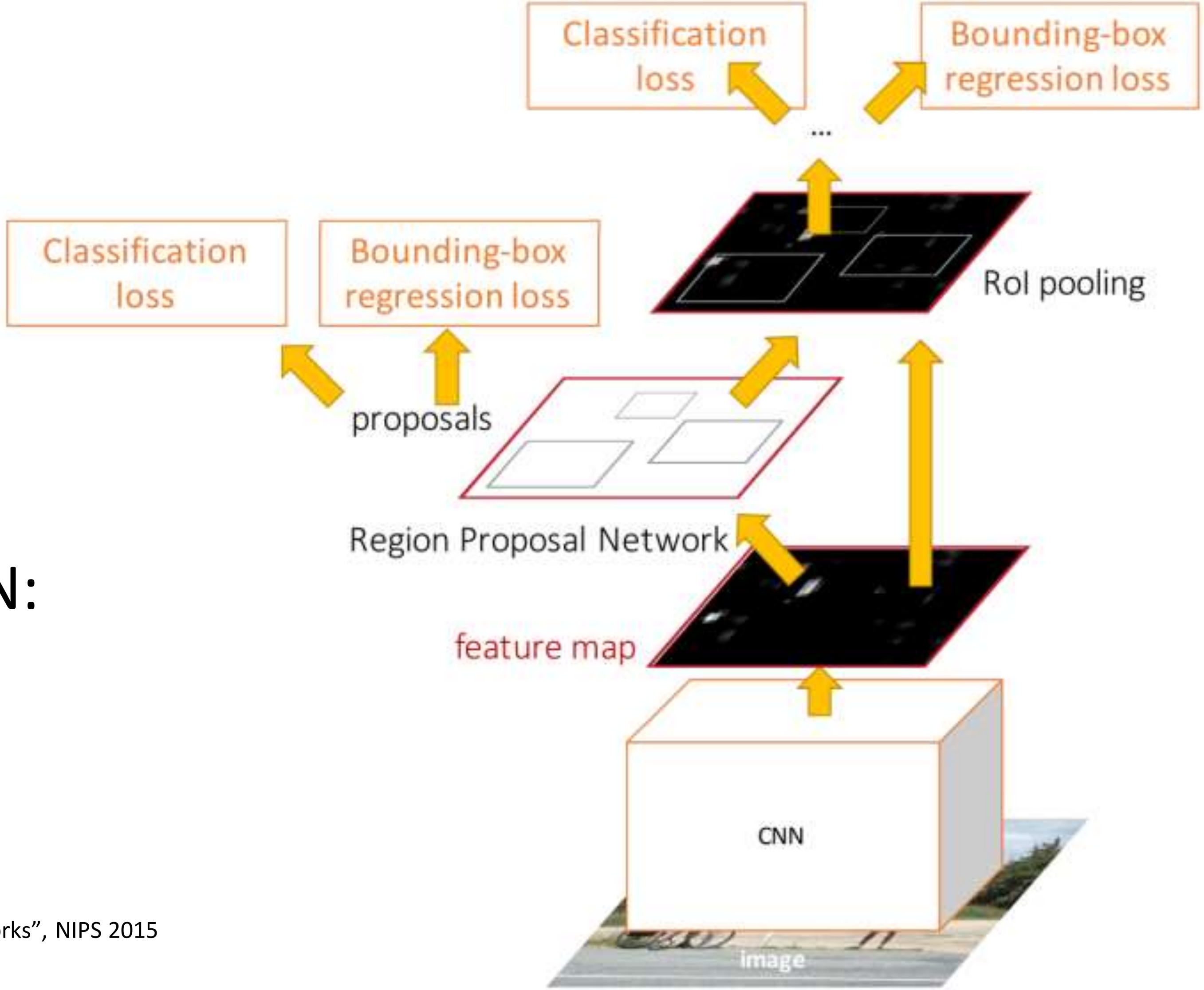
Recall: Region proposals computed by heuristic "Selective Search" algorithm on CPU -- let's learn them with a CNN instead!

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

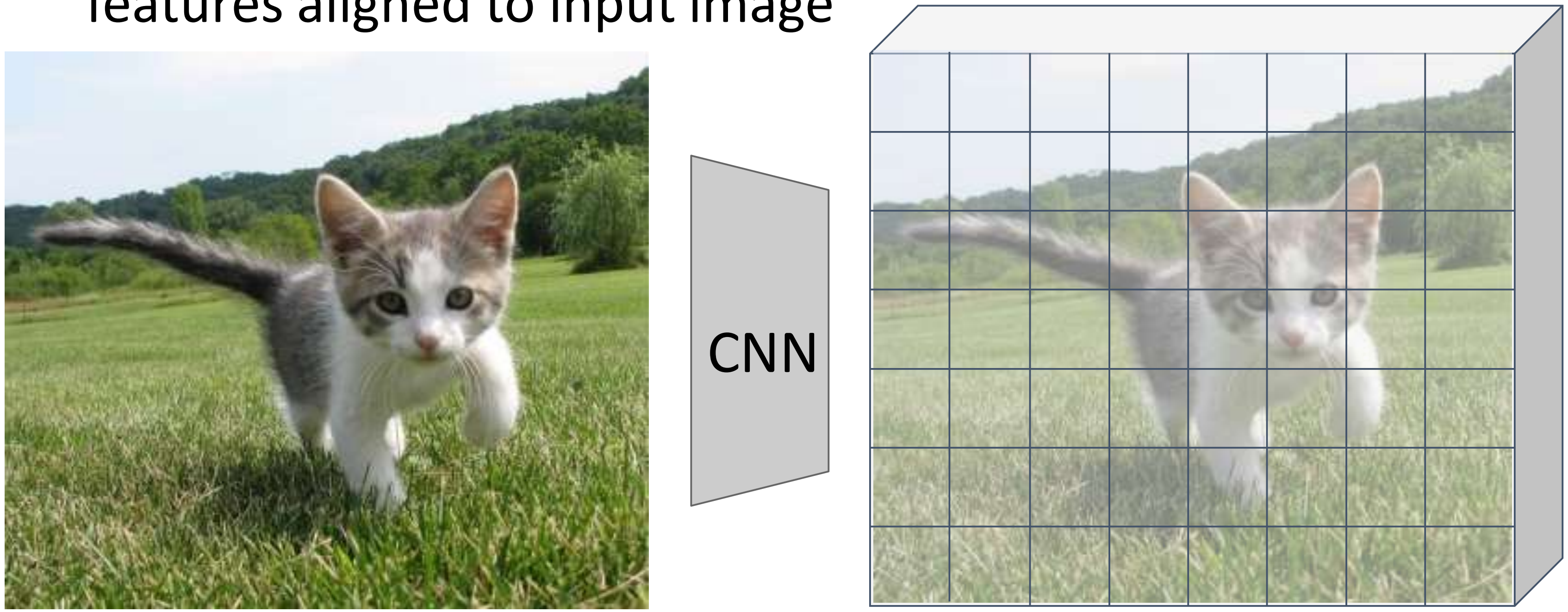
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

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)

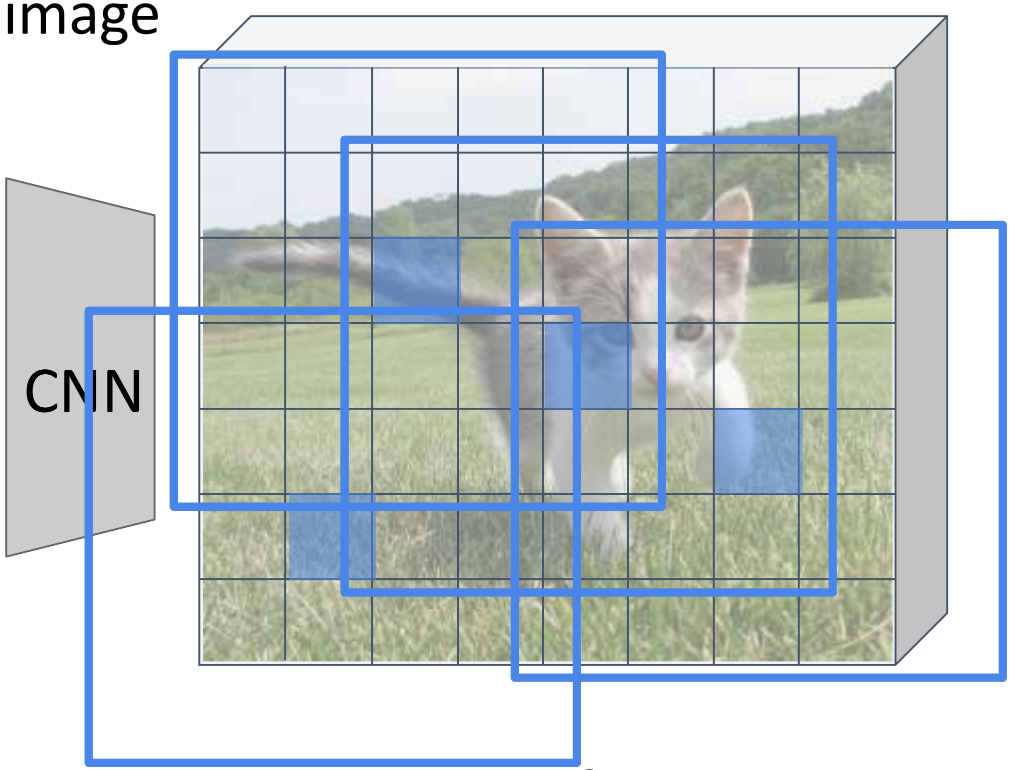


Image features
(e.g. 512 x 20 x 15)

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)

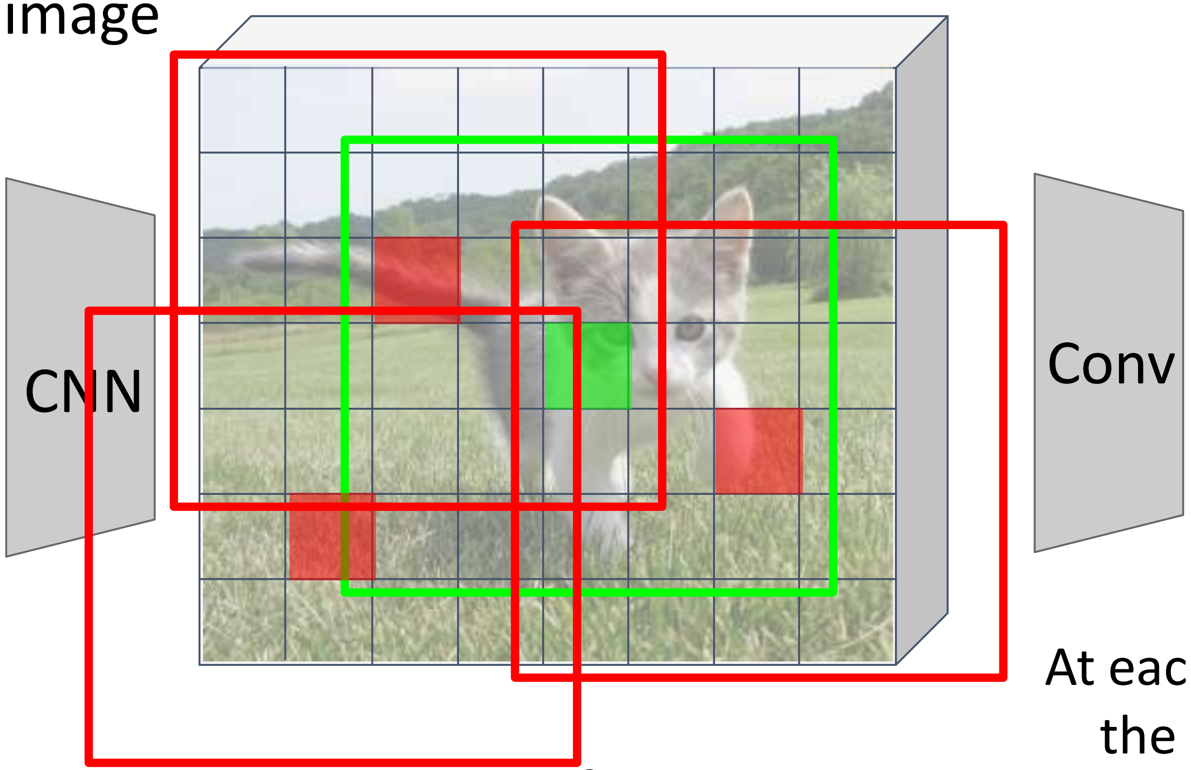


Image features
(e.g. 512 x 20 x 15)

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)

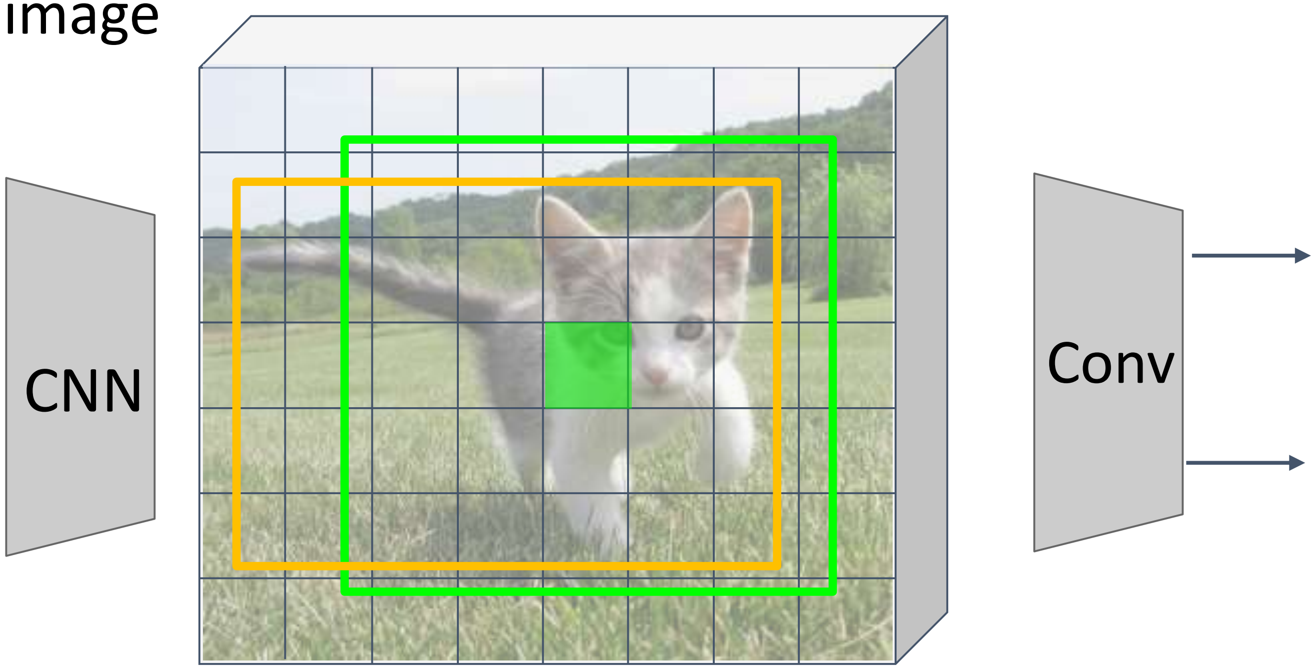


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

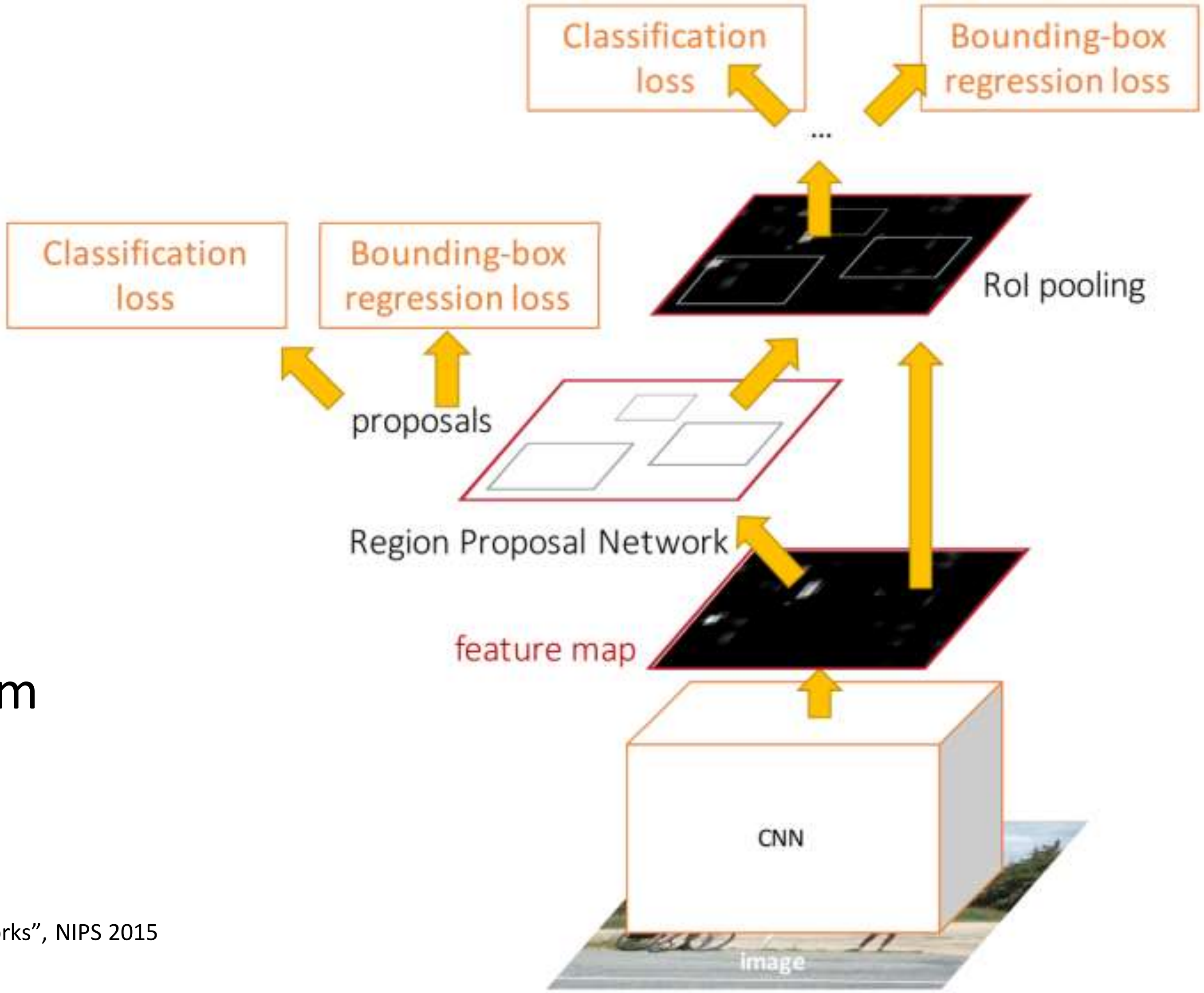
Box transforms
4 x 20 x 15

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

Faster R-CNN: Learnable Region Proposals

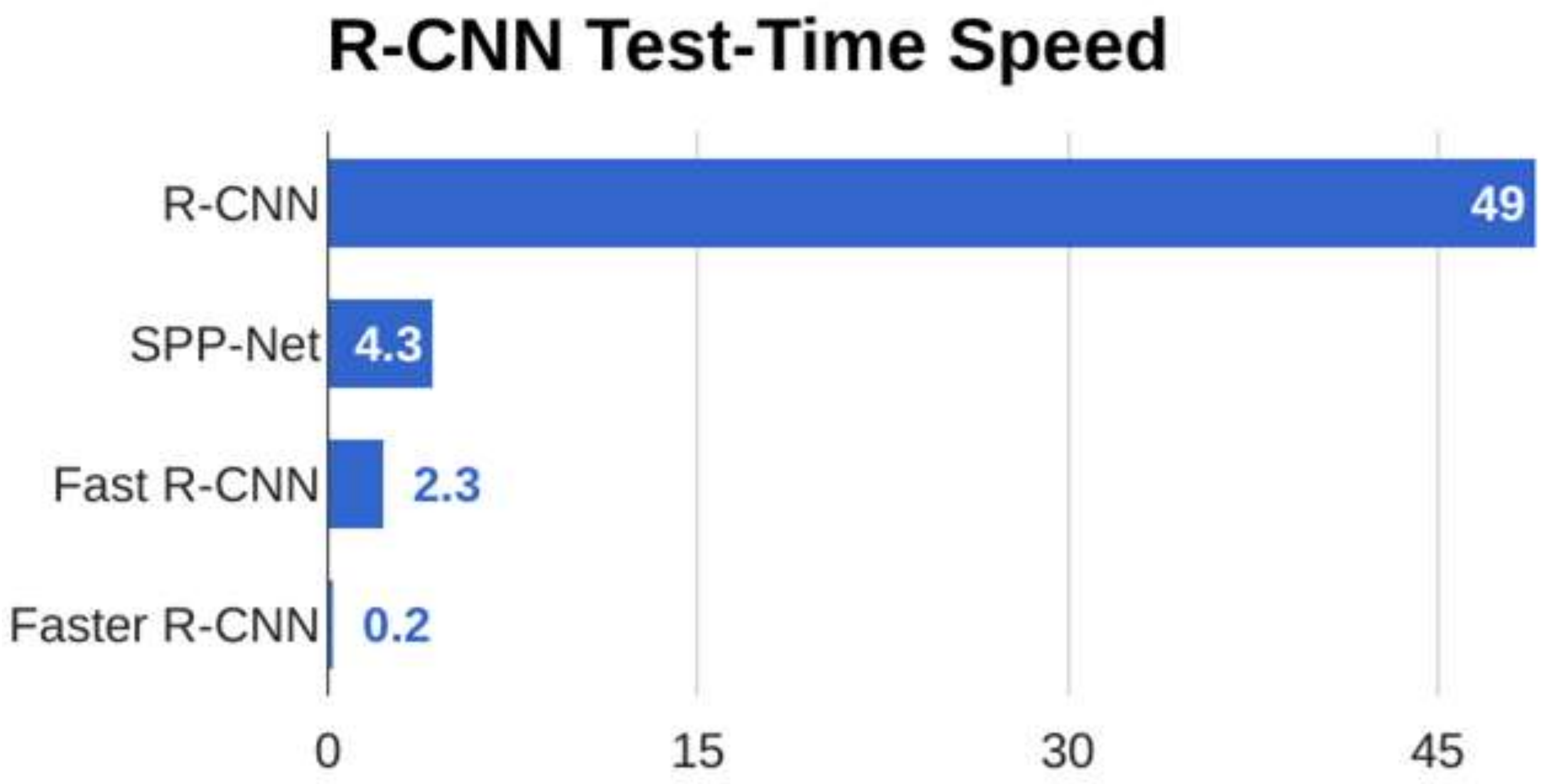
Jointly train with 4 losses:

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

Fast**er** R-CNN: Learnable Region Proposals

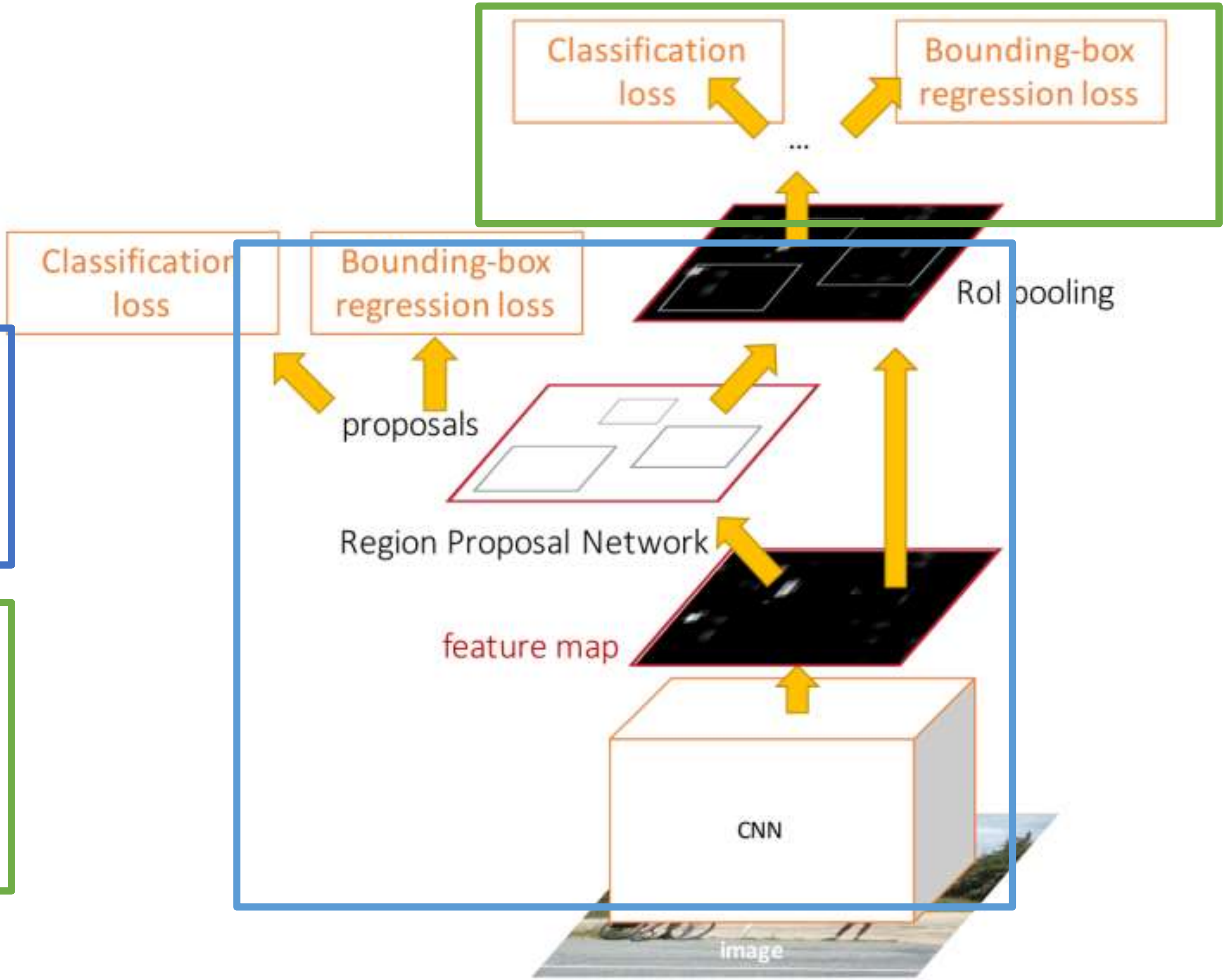


Faster R-CNN: Learnable Region Proposals

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



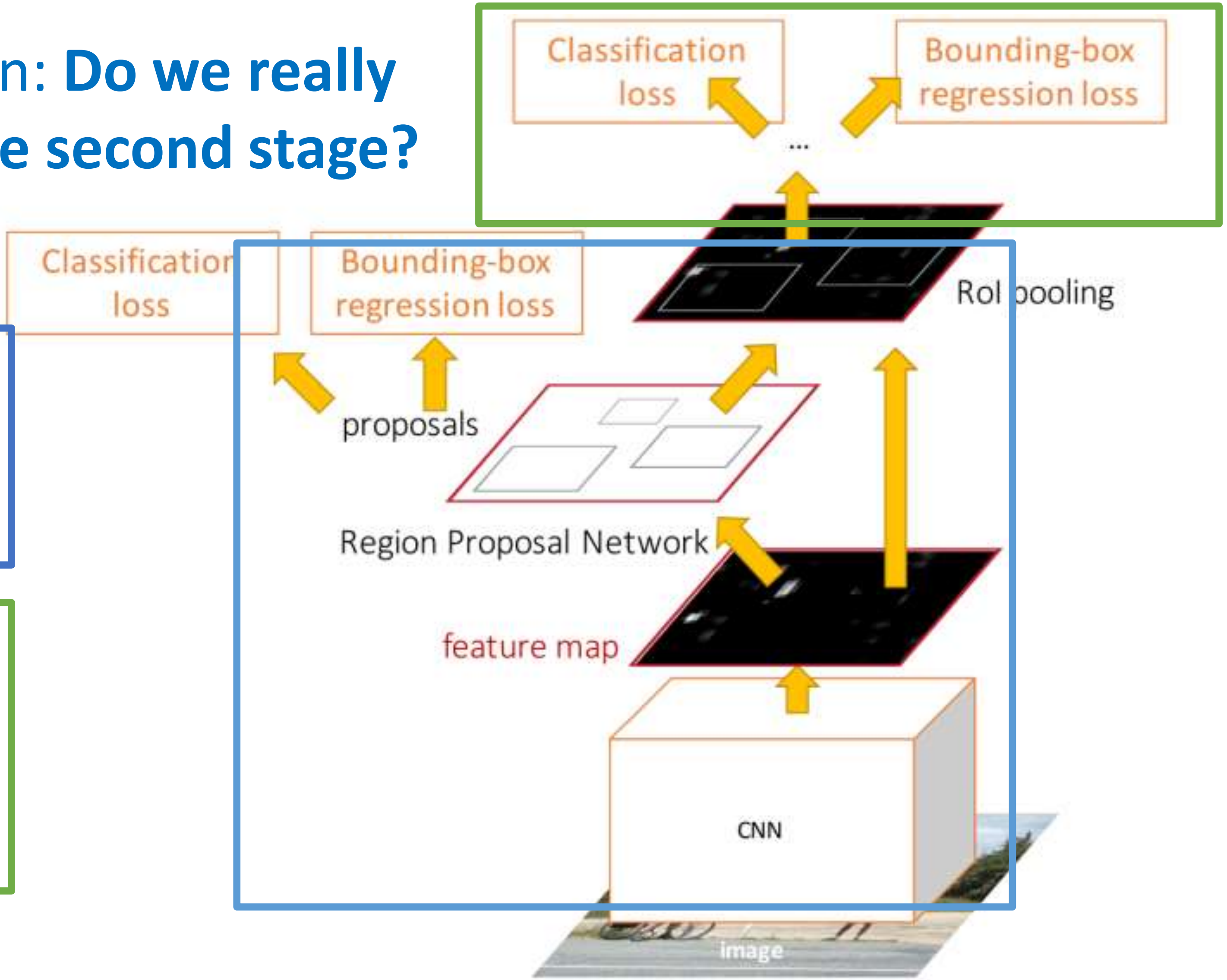
Faster R-CNN: Learnable Region Proposals

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

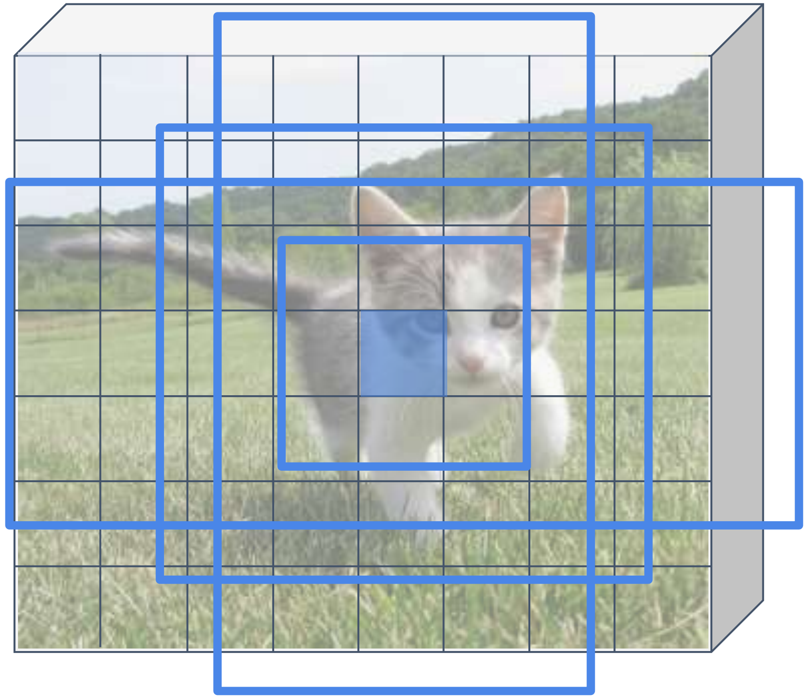


Image features
(e.g. 512 x 20 x 15)

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)

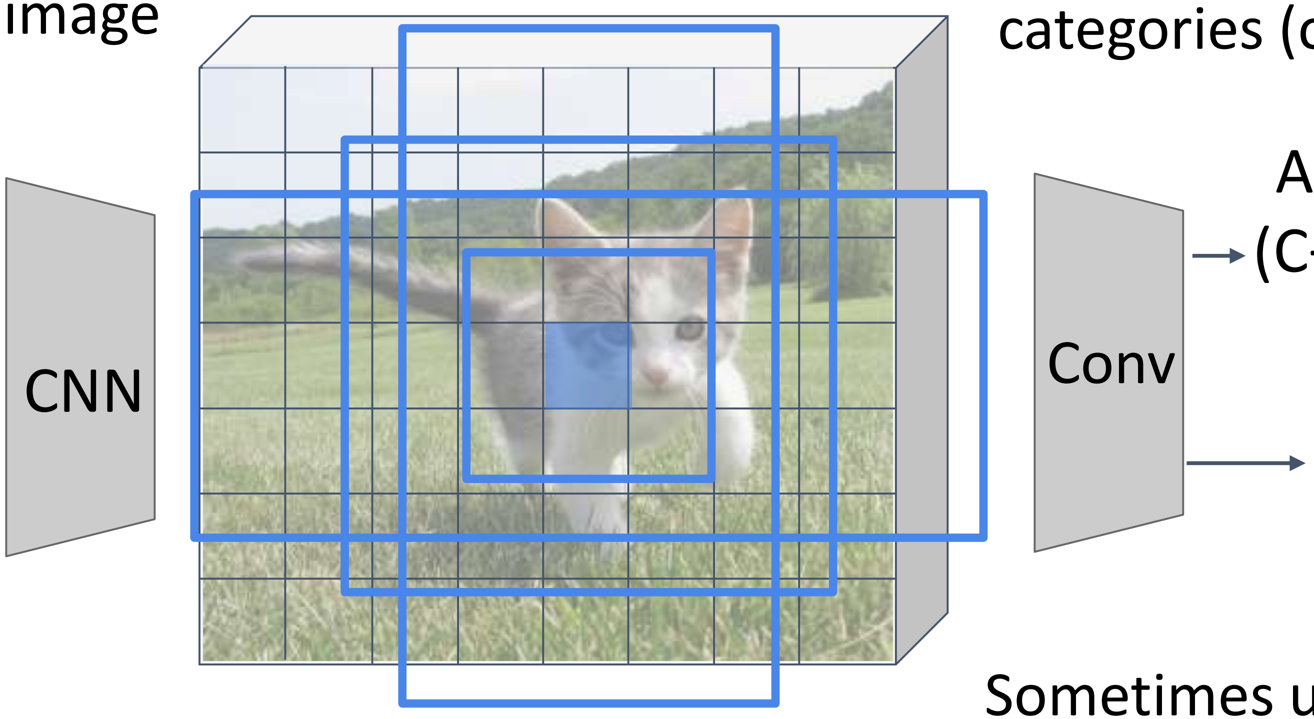


Image features
(e.g. 512 x 20 x 15)

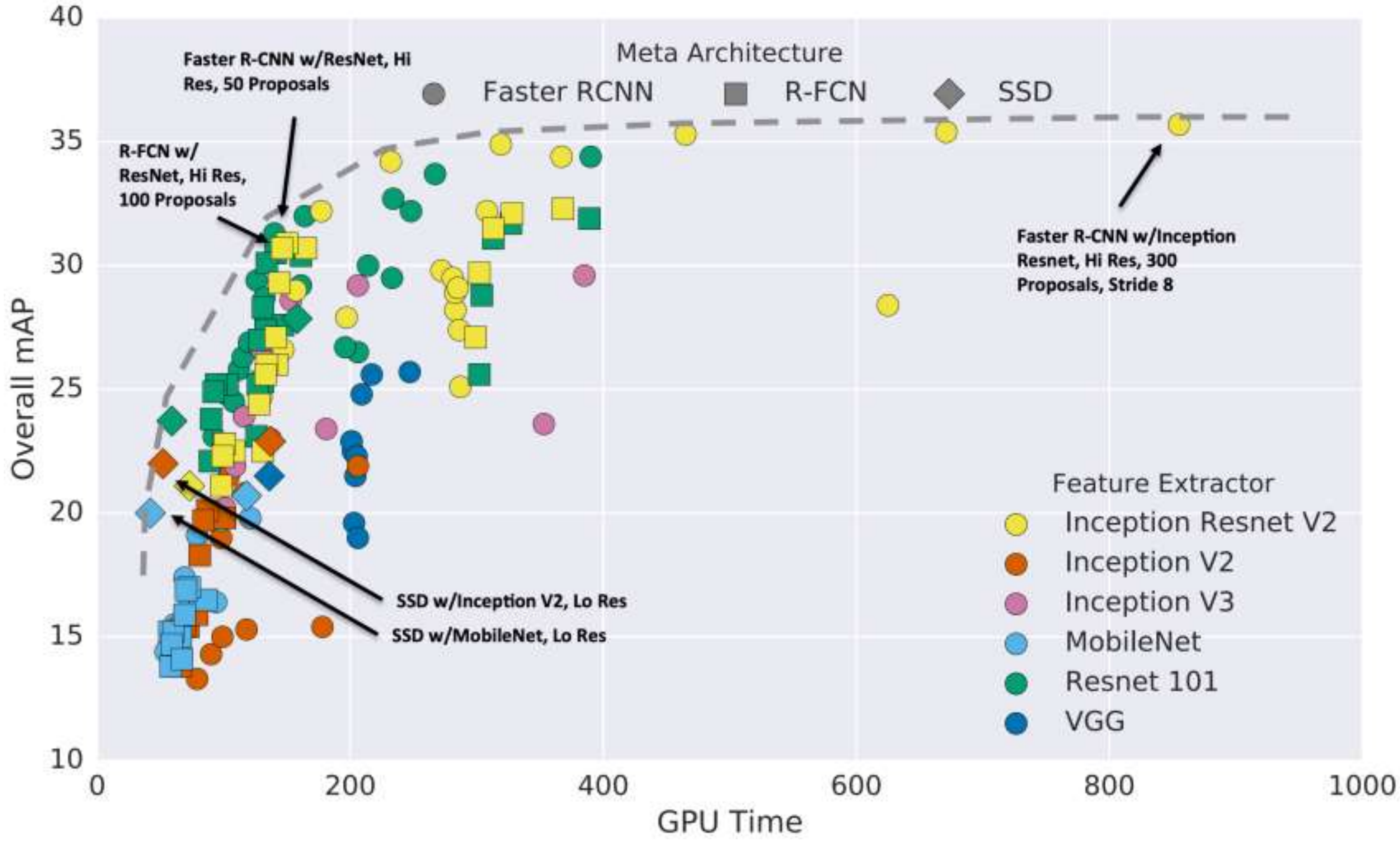
Anchor category
→ (C+1) x K x 20 x 15

→ Box transforms
C x 4K x 20 x 15

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

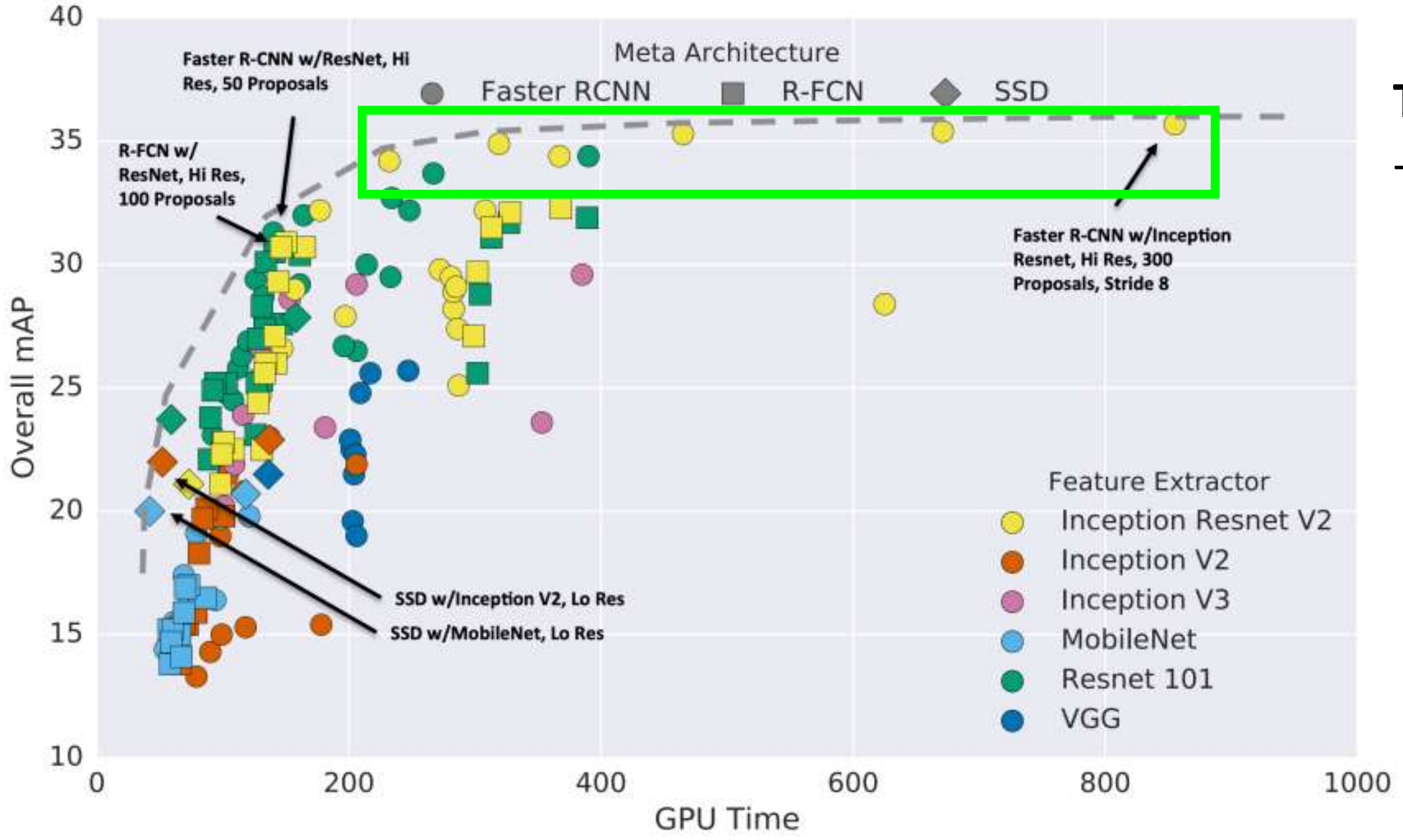
Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016
Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

Object Detection: Lots of variables!



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

Object Detection: Lots of variables!

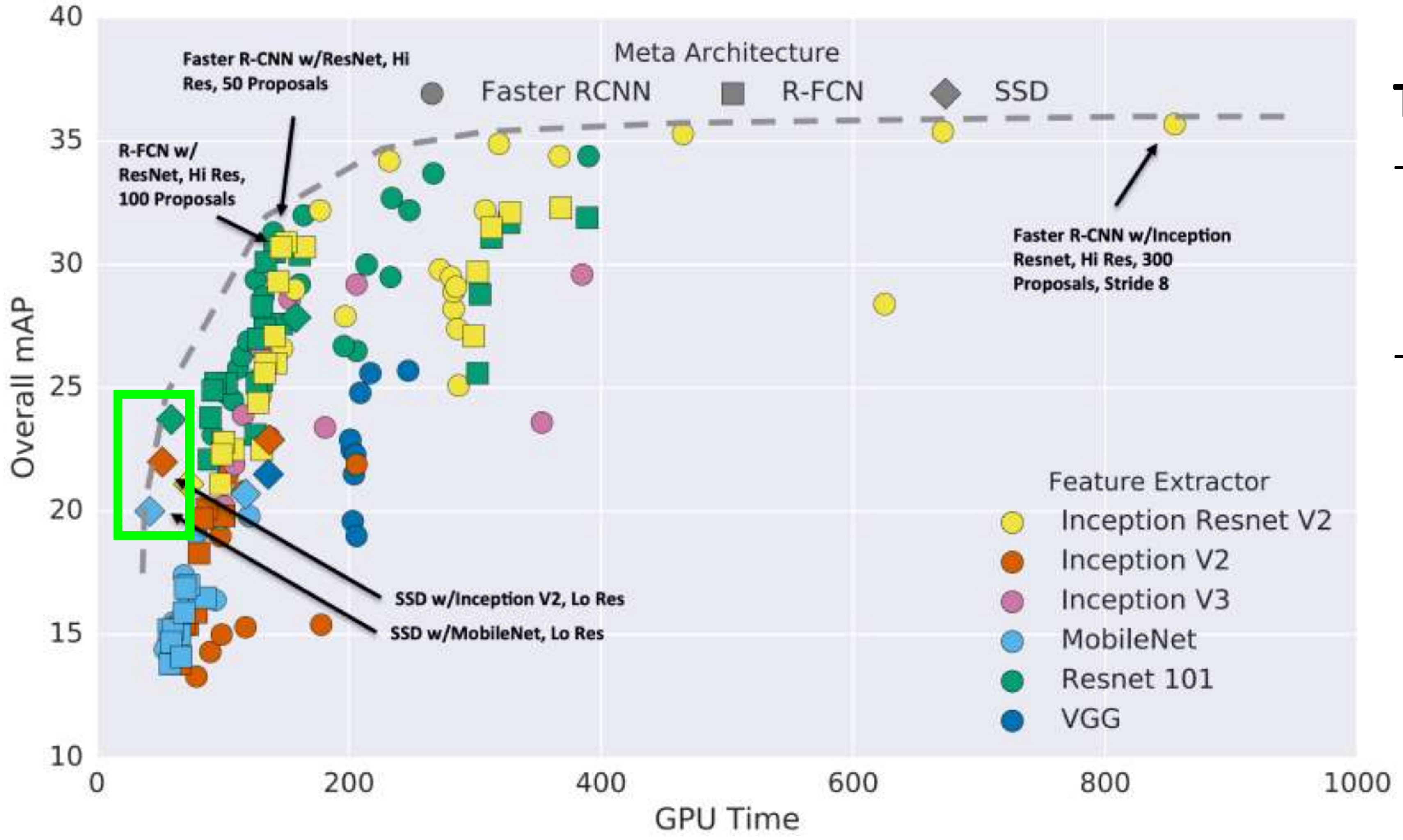


Takeaways:

- 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!

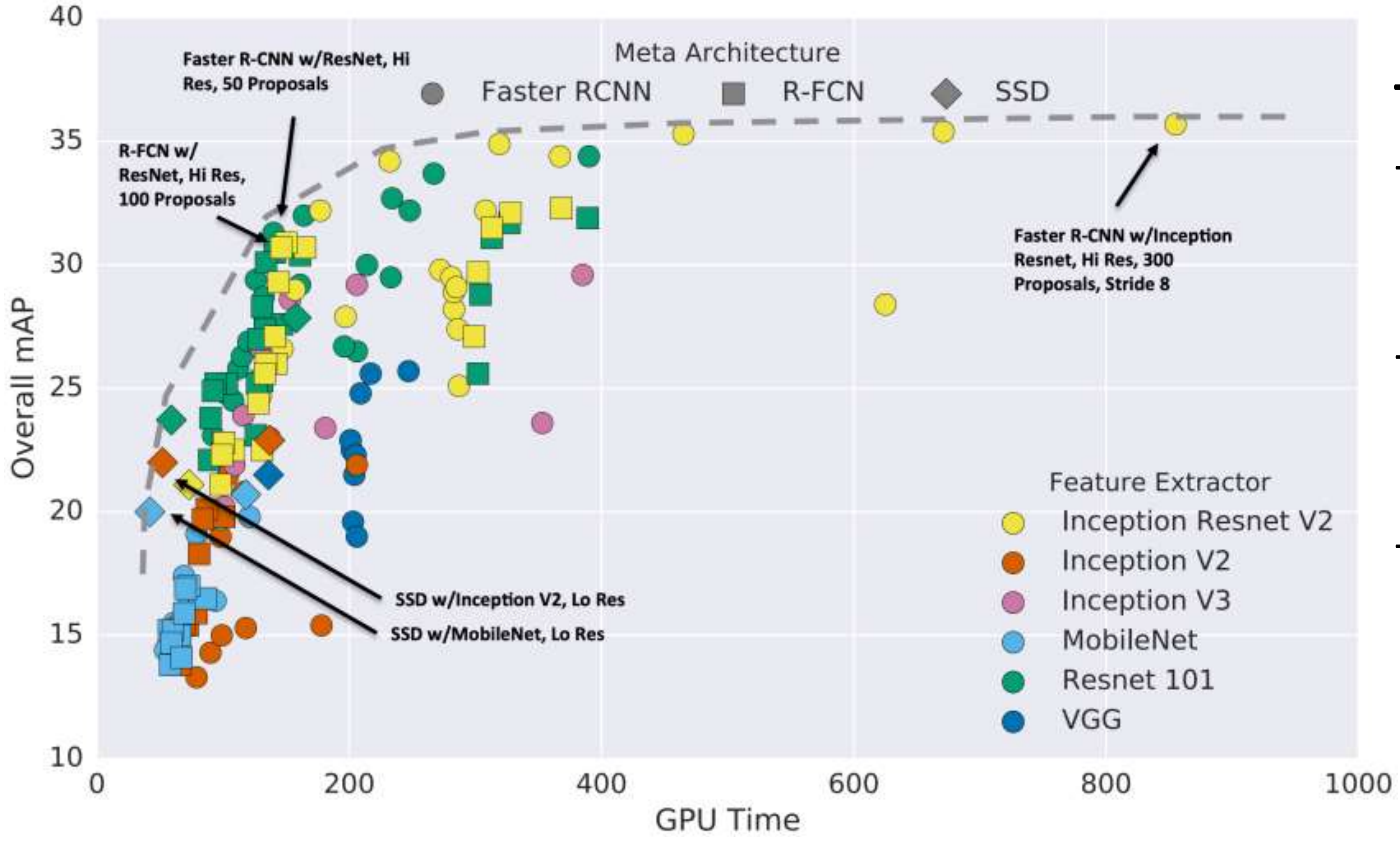


Takeaways:

- 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!

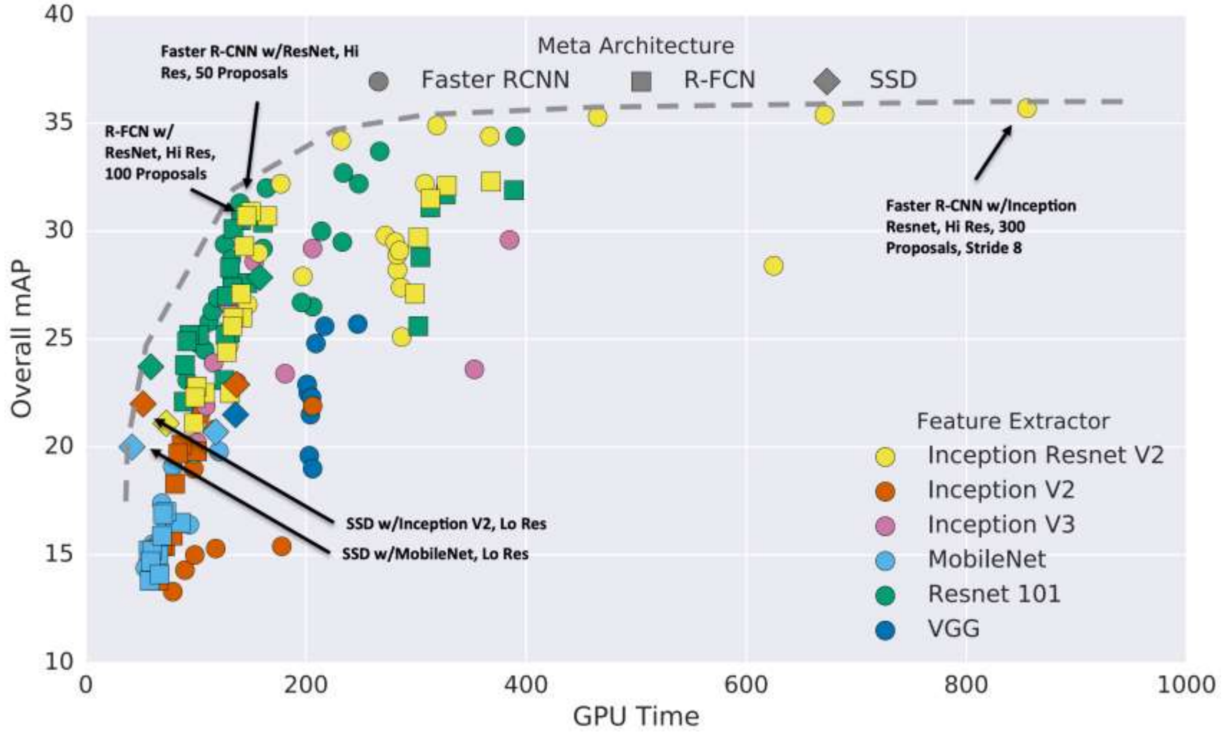


Takeaways:
- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well
- Bigger backbones improve performance, but are slower

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

Object Detection: Lots of variables!

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

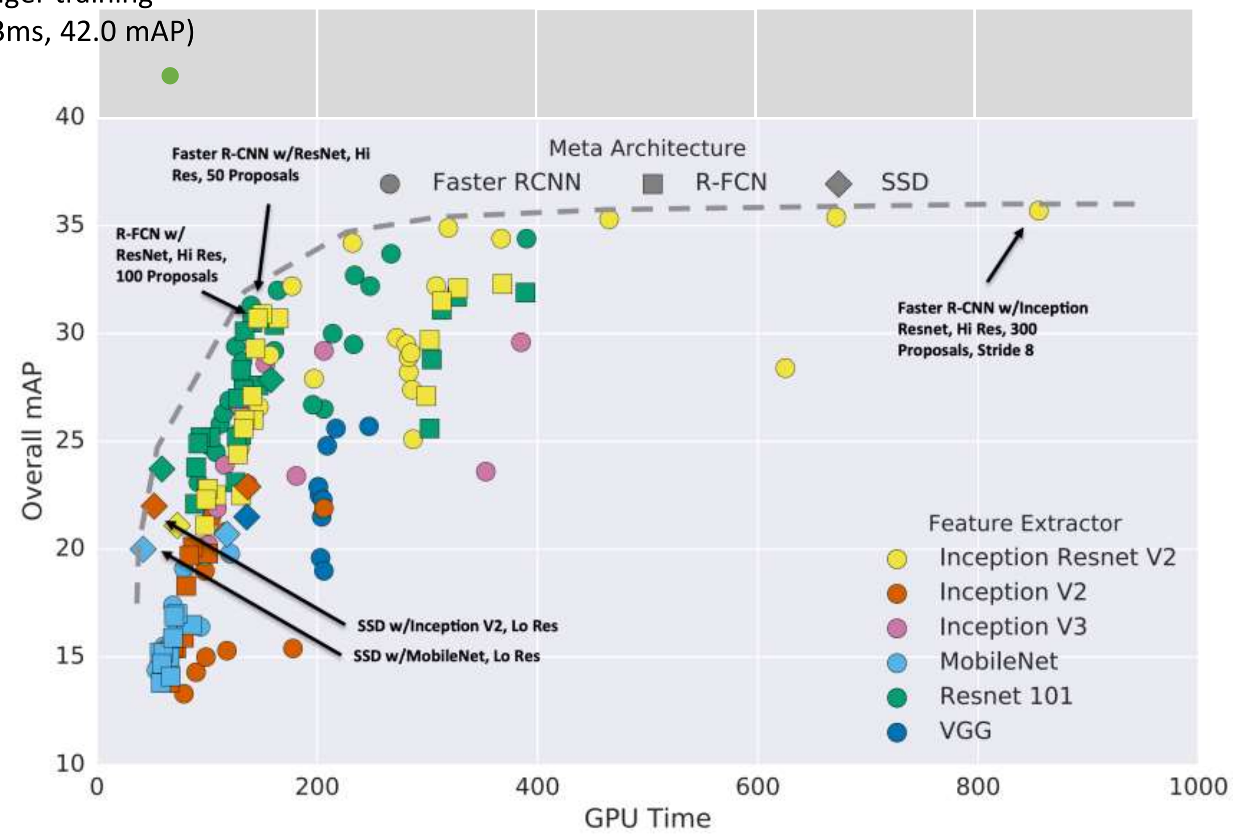


Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!

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



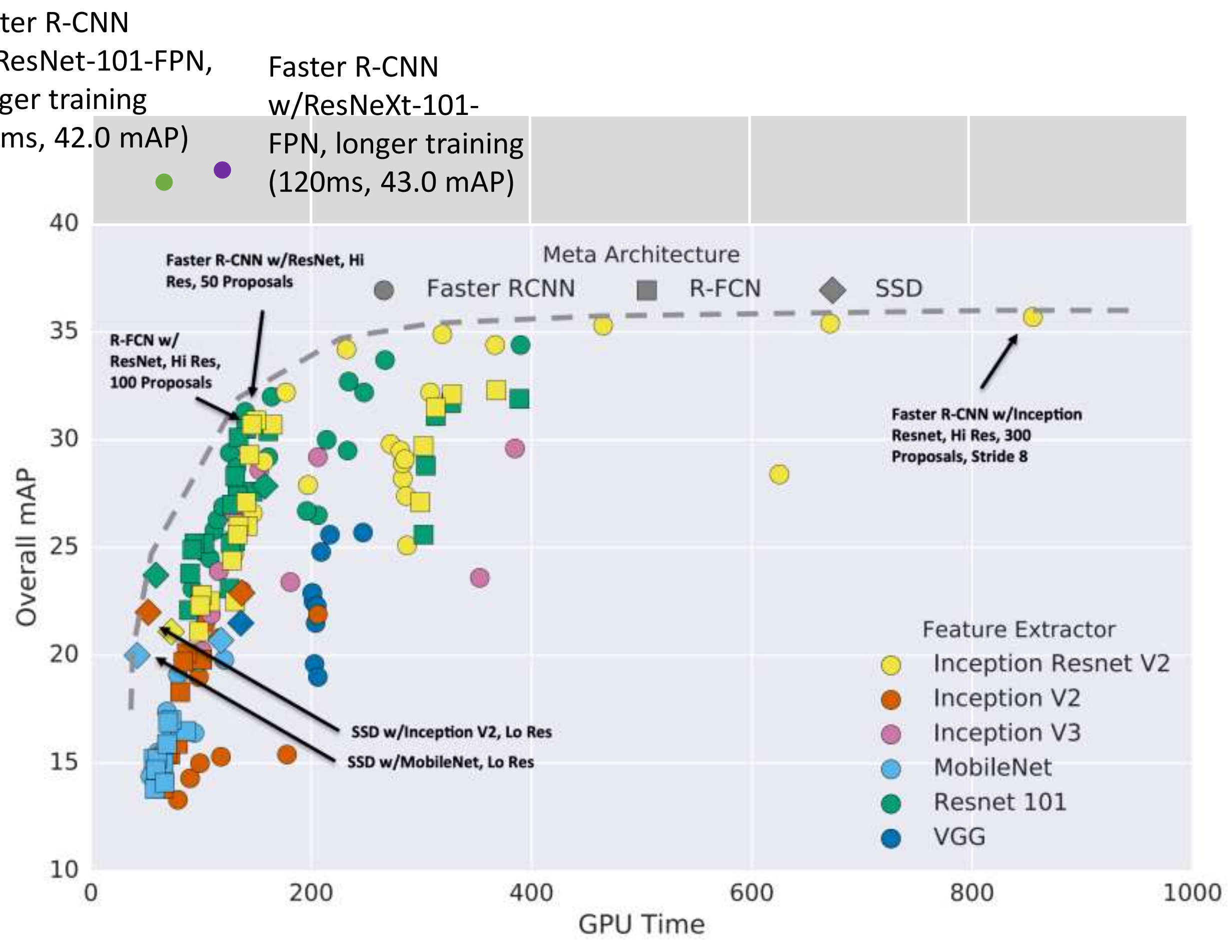
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- 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!



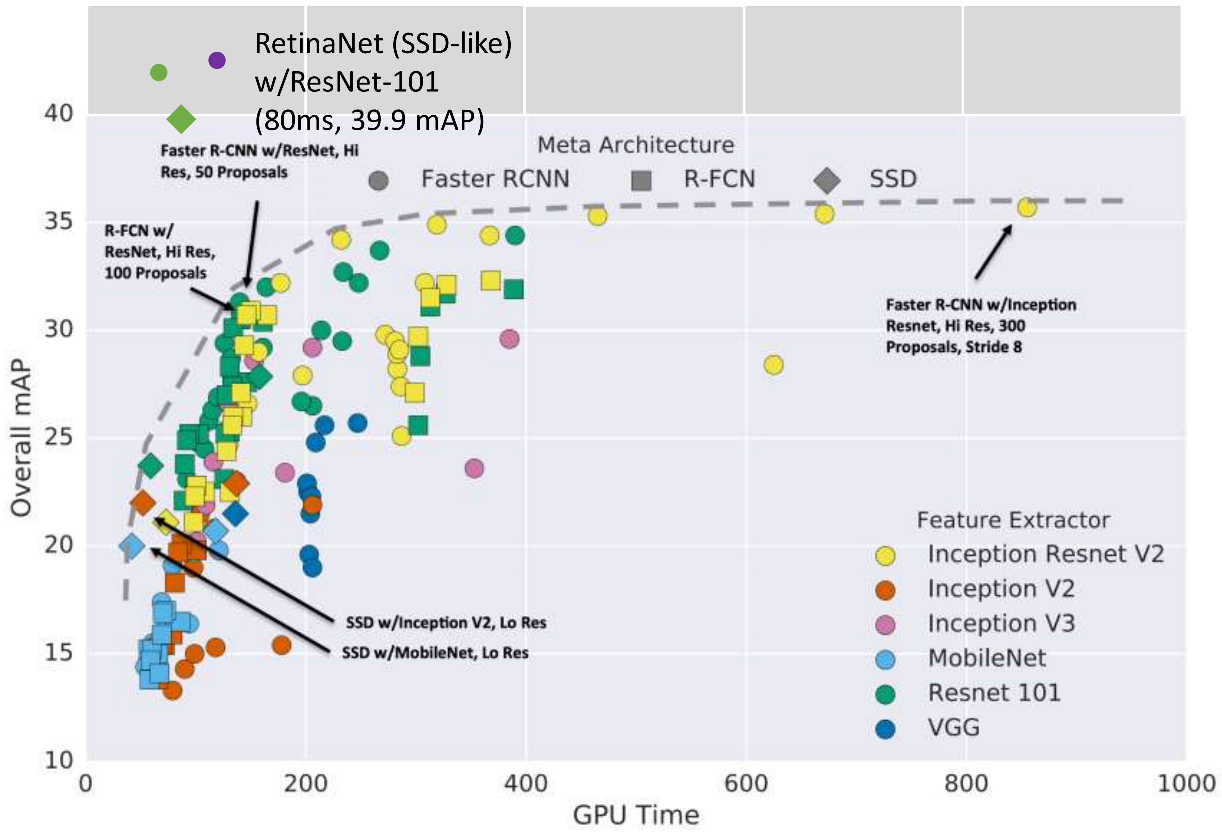
These results are a few years old ... since then GPUs have gotten faster, and we've improved performance with many tricks:

- 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!



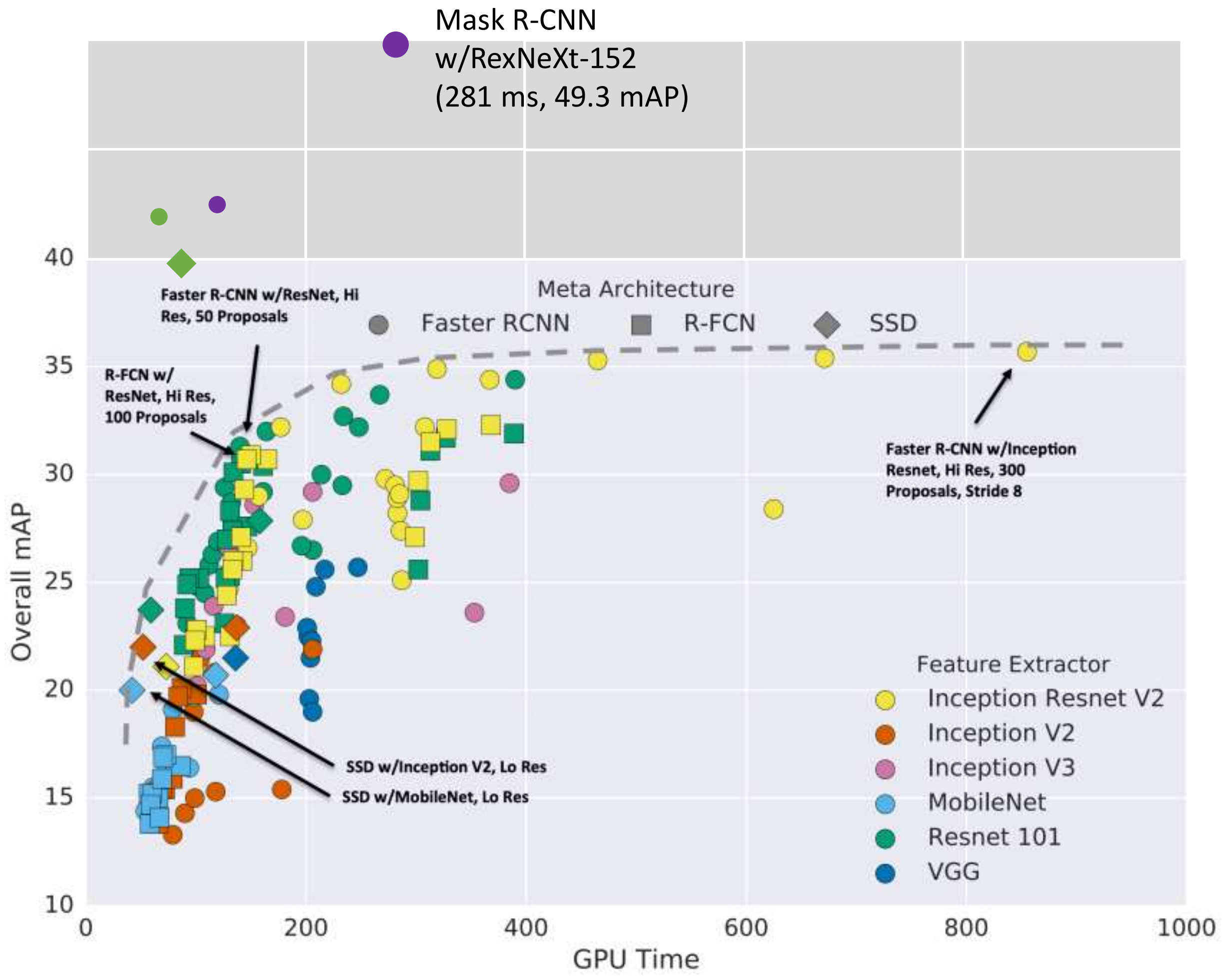
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

- 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!



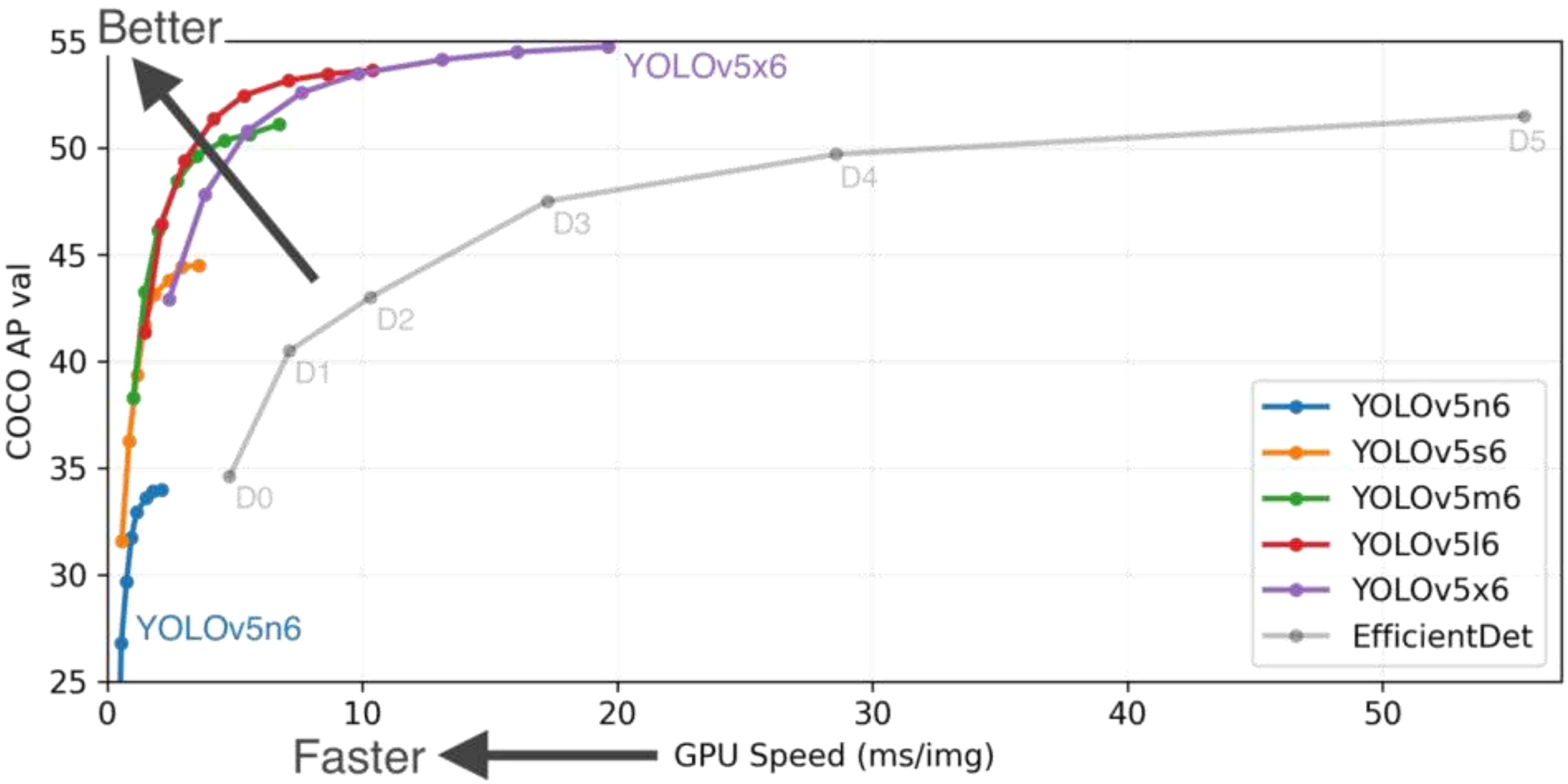
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

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

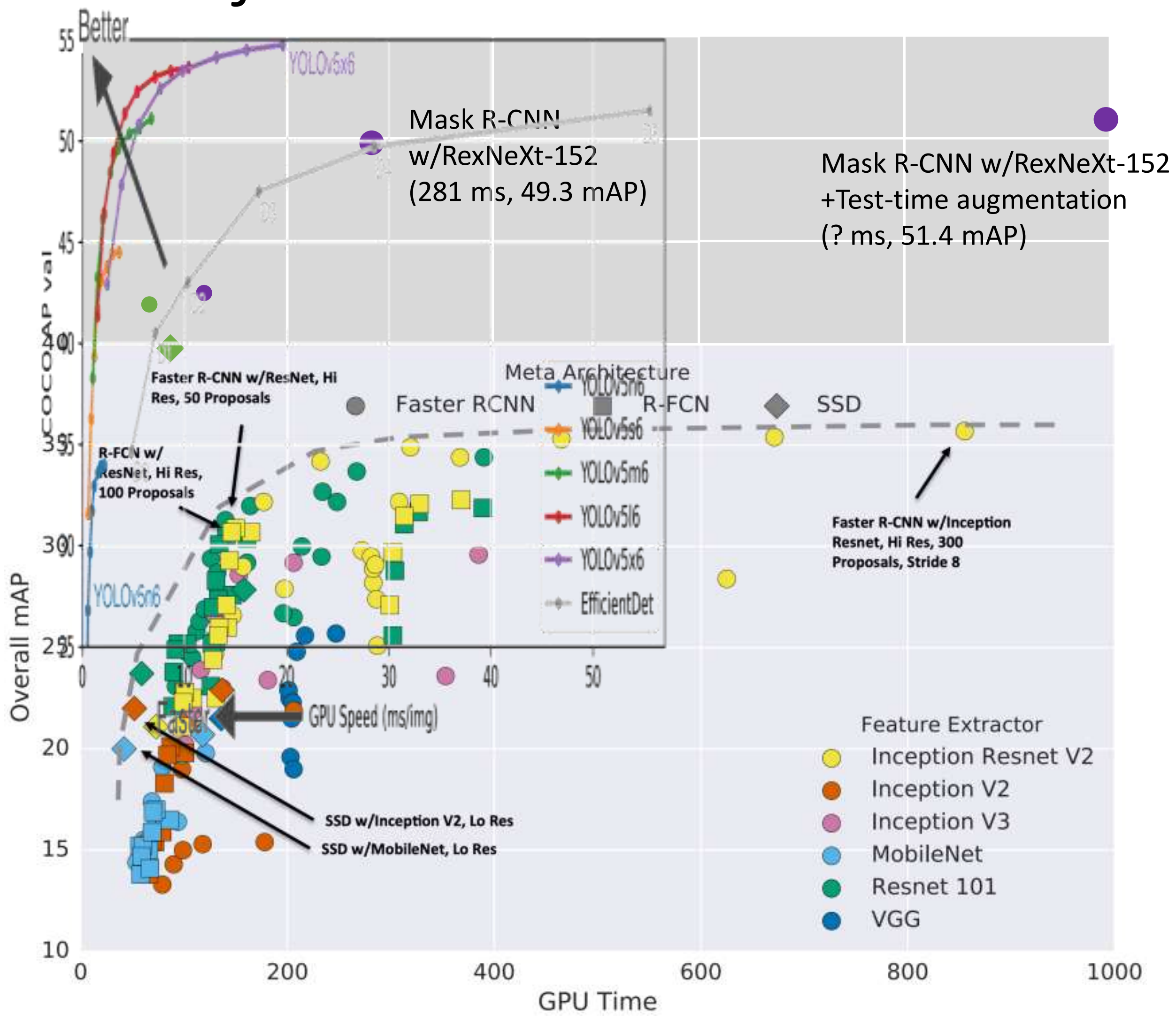
Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Wu et al, Detectron2, GitHub 2019

Object Detection: Lots of variables!



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: Open-Source Code

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

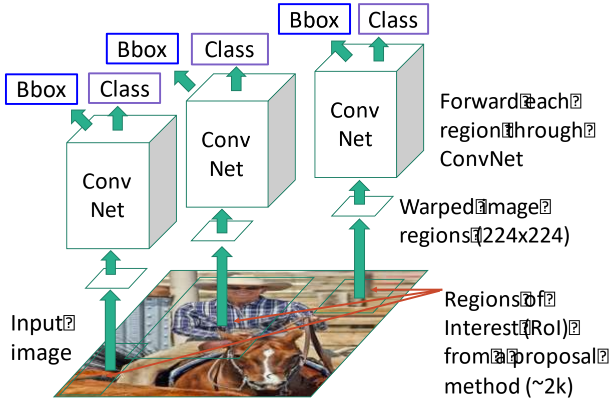
YOLOv5 (Ultralytics – PyTorch):

https://github.com/ultralytics/yolov5

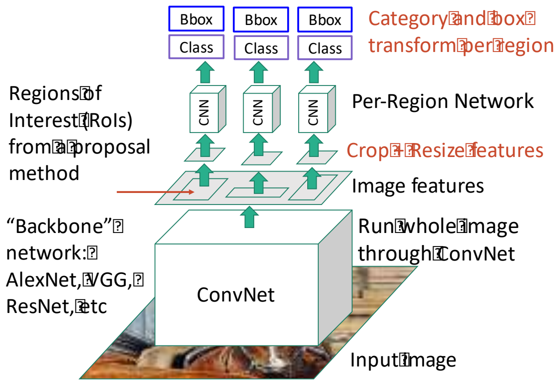
Single-stage

Summary

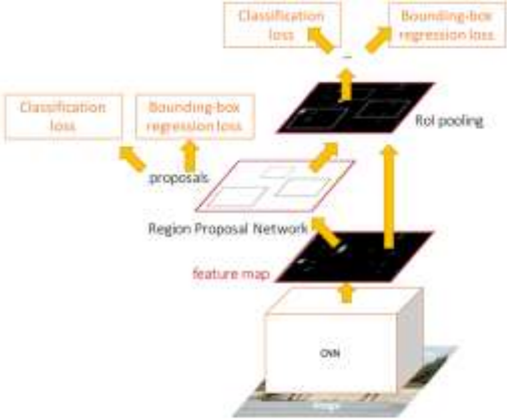
"Slow" R-CNN: Run CNN independently for each region



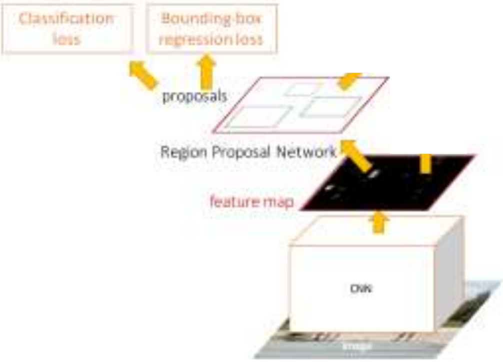
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



Test 2

20 Dec, Monday

900-1100 am

Online via Teams