Object detection I

Semester 2, 2021

Kris Ehinger

https://www.youtube.com/watch?v=WZmSMkK9VuA

Outline

- Object detection basics
- R-CNN
- Fast R-CNN
- Faster R-CNN

Learning outcomes

- Explain the standard approach to object detection (sliding window)
- Explain how object detection is implemented in region-based CNNs
- Explain design issues and trade-offs involved in building object detection methods

Object detection basics

Classification vs. Detection

Classification vs. Detection

- Object detection = locate objects in an image
    - Classification: “Is this a <class label>?”
    - Detection: “Where is the <class label>?”
- Object detection is usually modelled as a classification task performed within patches of an image
    - Detection: For every patch, “Is this a <class label>?”

Sliding window approach

Sliding window approach

Sliding window approach

Sliding window

- Free parameters:
    - Stride
    - Scale (size of window)
    - Shape (aspect ratio)
- Generally object dimensions are unknown, so a range of scales/shapes will be required

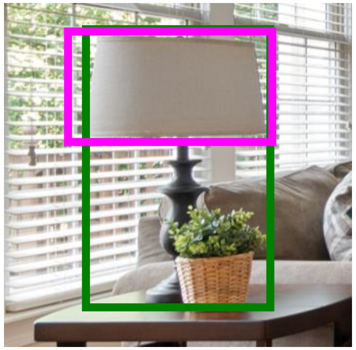
Sliding window evaluation

- Another parameter: threshold for detection
- Windows over the threshold will be considered “target”
- Note that this makes evaluation tricky

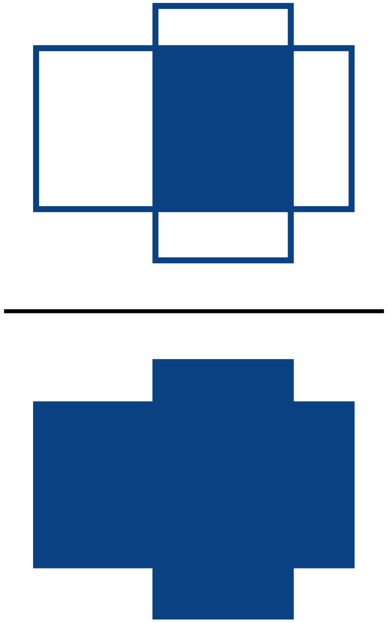
  ← Is this a good result?

Window evaluation: IoU

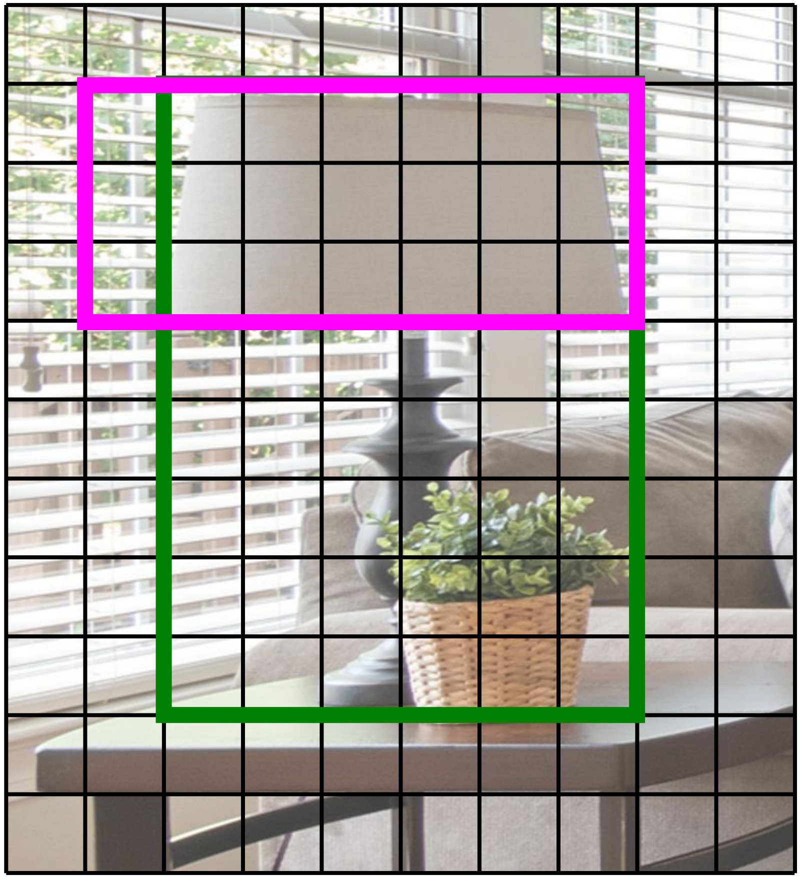
- Common method to evaluate a window result is Intersection over Union (IoU) between true bounding box and detection window



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$



Example: IoU



Summary

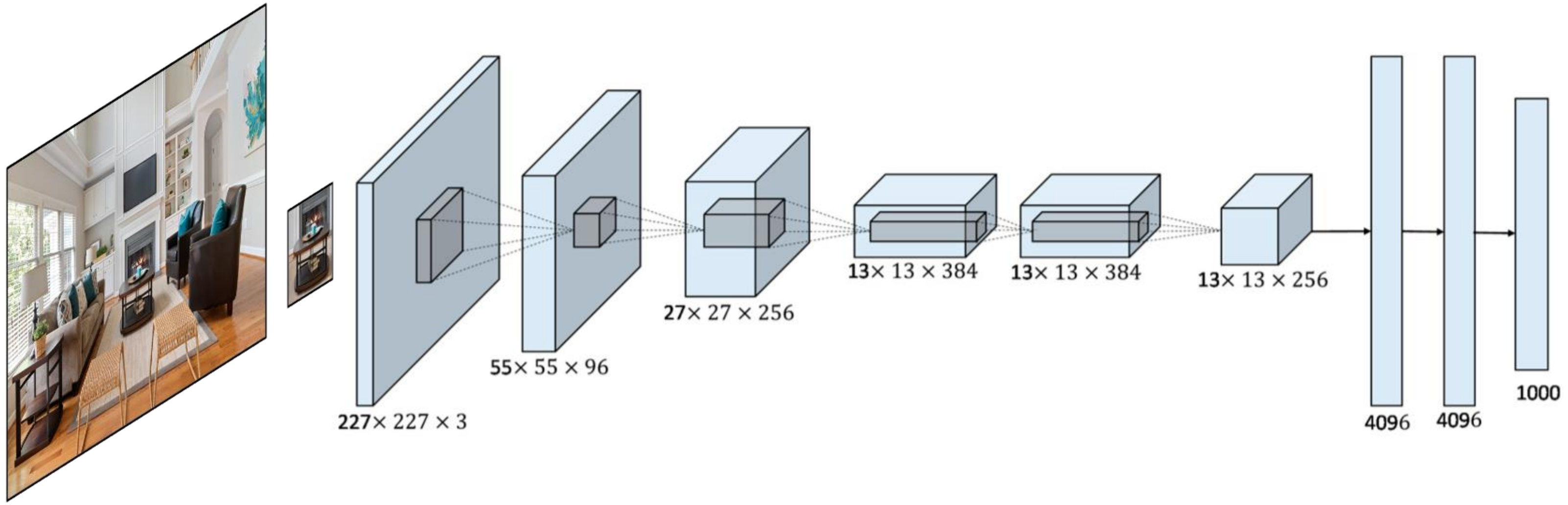
- Object detection is generally modelled as image classification within small regions of an image
- Windows over some threshold = "detections," can be evaluated using IoU with ground truth
- Problems:
  - Very large number of possible windows (slow, increases probability of false detections)
  - Overall evaluation of images with multiple targets can be complicated (multiple targets, multiple detection windows, different IoUs)

Figure: R. Girshick (2015)

R-CNN

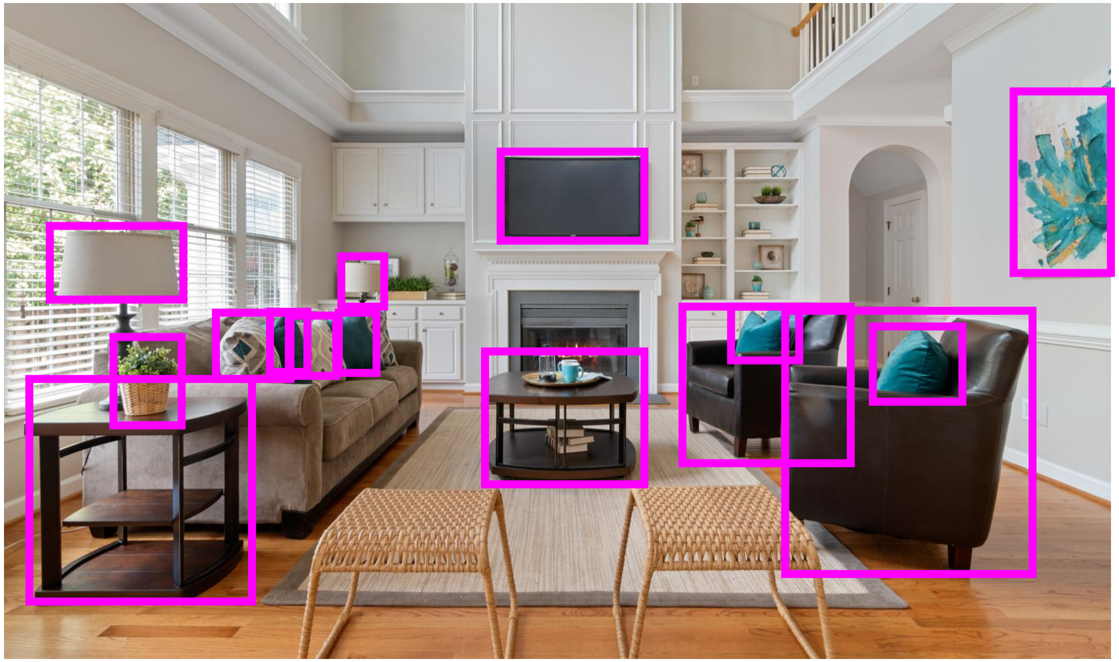
Sliding window classification

- Very large number of windows per image (C scales x S shapes x N locations)
- How to classify efficiently in parallel?



Sliding window classification

- Even in a neural network, classifying all possible boxes may be slow (may also increase false alarms)
- Solution? Focus on boxes most likely to be objects



R-CNN

- R-CNN = Region-based convolutional neural network
- Given an image, identify a small number of windows for object detection (“region proposals” or “regions of interest (ROIs)”)

Generating region proposals

- R-CNN uses Selective Search to generate ROIs
- Selective Search algorithm:
  - Oversegment image into superpixels
  - Iteratively combine adjacent superpixels based on similarity in colour + texture, size, and compactness

Image: Uijlings, van de Sande, Gevers, & Smeulders (2012)

R-CNN: Region-Based CNN

Input image

Figure: R. Girshick (2015)

R-CNN: Region-Based CNN

Input image

Regions of Interest (RoI) from Selective Search (~2k)

Figure: R. Girshick (2015)

R-CNN: Region-Based CNN

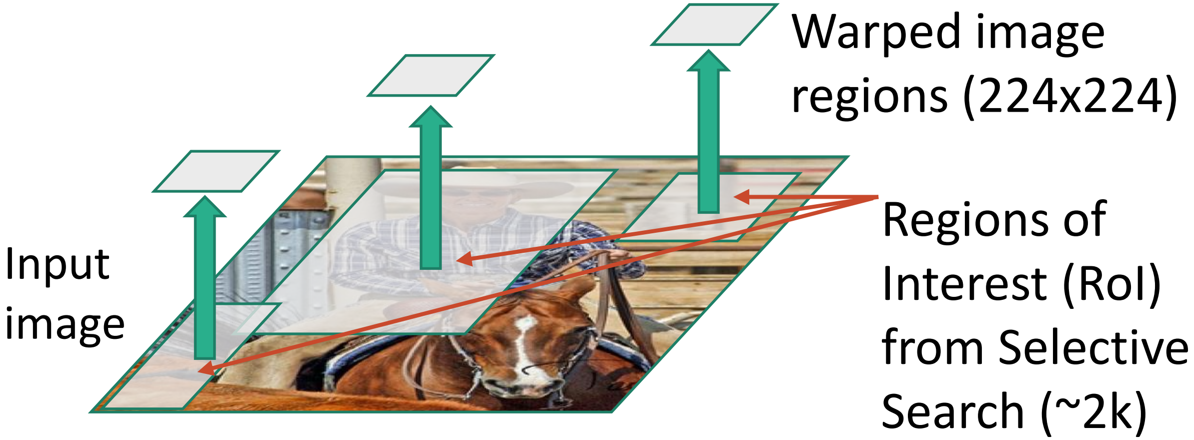
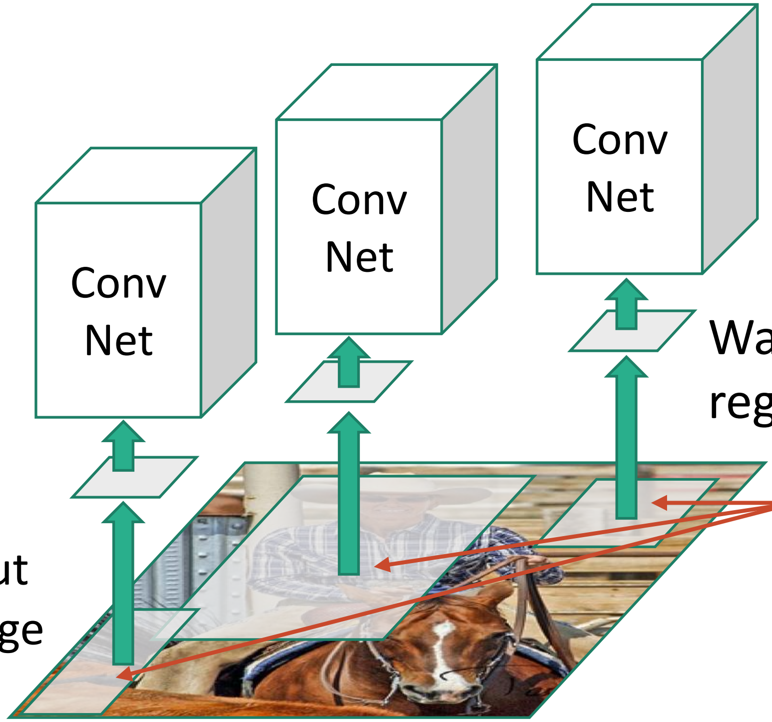


Figure: R. Girshick (2015)

R-CNN: Region-Based CNN



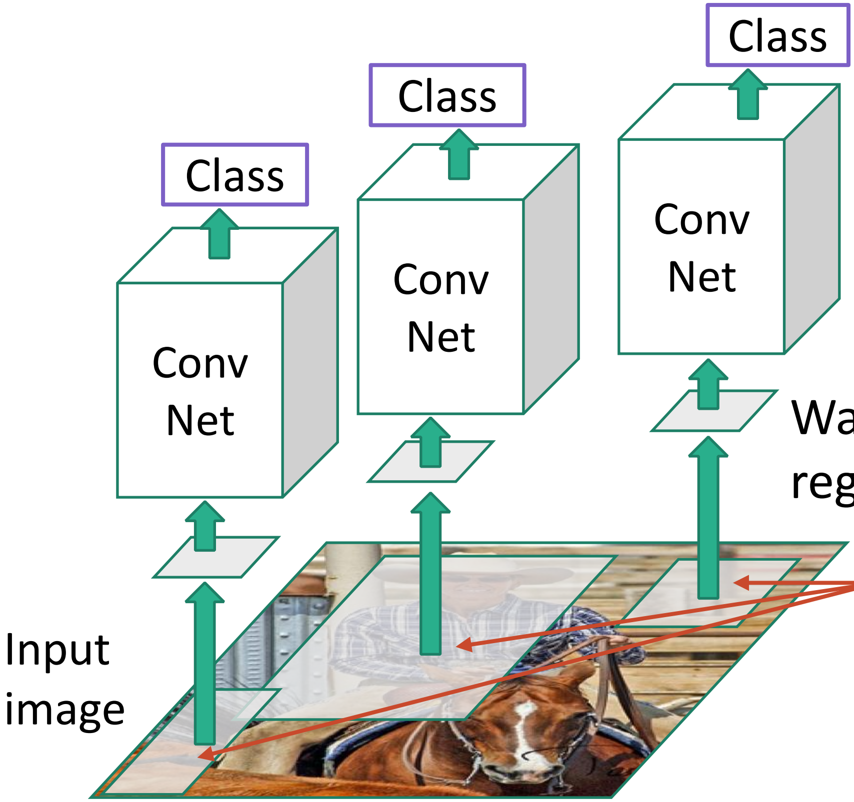
Run each region through a CNN

Warped image regions (224x224)

Regions of Interest (RoI) from Selective Search (~2k)

Figure: R. Girshick (2015)

R-CNN: Region-Based CNN



Classify each region...

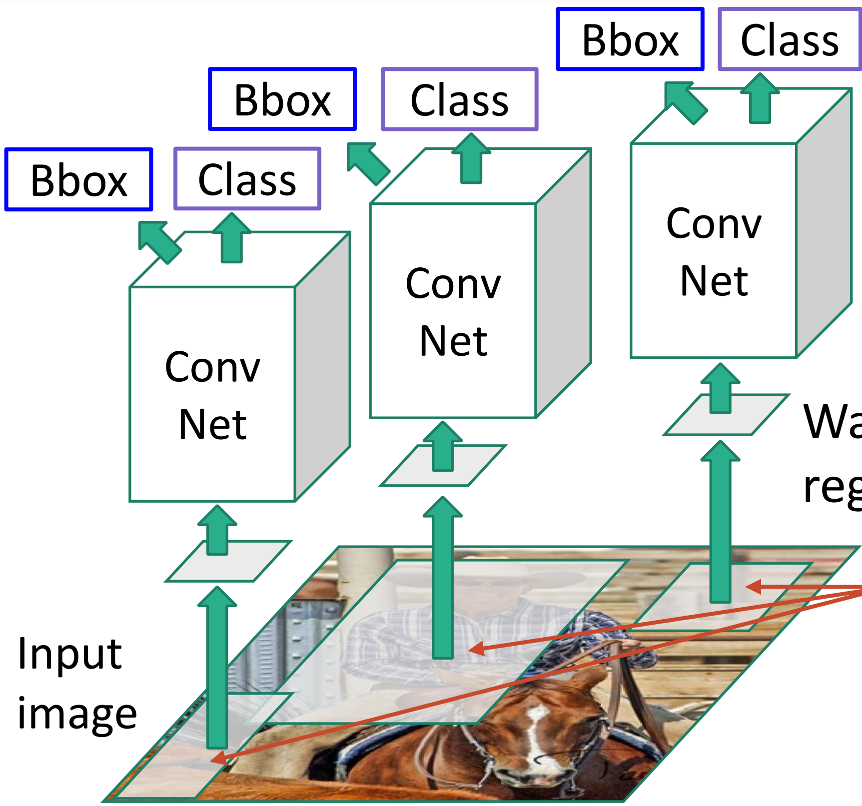
Run each region through a CNN

Warped image regions (224x224)

Regions of Interest (RoI) from Selective Search (~2k)

Figure: R. Girshick (2015)

R-CNN: Region-Based CNN



Classify each region...

Run each region through a CNN

and do bounding box regression: predict “transform” to correct the ROI (t_x, t_y, t_h, t_w)

Warped image regions (224x224)

Regions of Interest (RoI) from Selective Search (~2k)

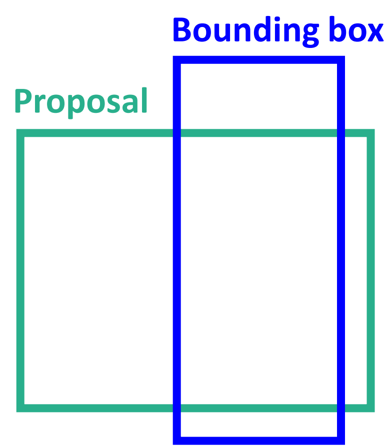


Figure: R. Girshick (2015)

Bounding box computation

- Original region proposal = (p_x, p_y, p_h, p_w)
- Transform = (t_x, t_y, t_h, t_w)
- Goal: compute bounding box = (b_x, b_y, b_h, b_w)
- Step 1. Translate

$$b_x = p_x + p_w t_x \qquad b_y = p_y + p_h t_y$$

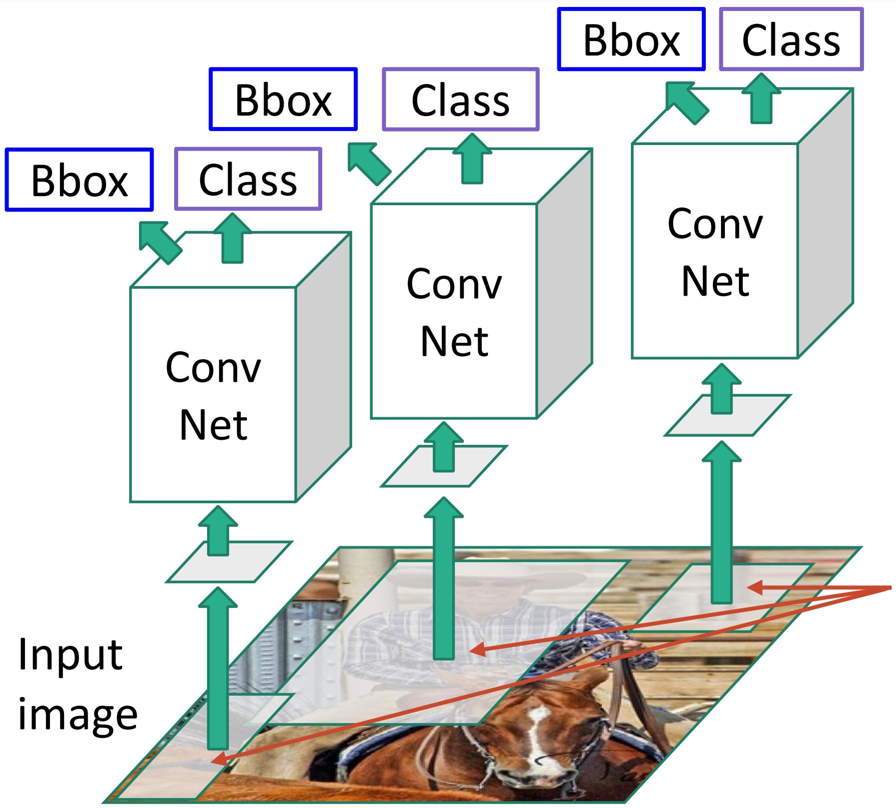
- Step 2. Scale

$$b_w = p_w \exp(t_w) \qquad b_h = p_h \exp(t_h)$$

R-CNN training

- CNN pretrained on ImageNet
- Last layer (1x1000) is replaced with a new classification layer of size 1x(N+1)
  - N+1 = N object classes + “background” class
  - CNN is retrained on (N+1)-way detection, using regions with IoU>=0.5 as ground truth “objects”
  - Sample regions so 75% of training set is “background”
- CNN features are used as input to:
  - Label classification model (1-vs-all linear SVM)
  - Bounding box model (class-specific linear regression)

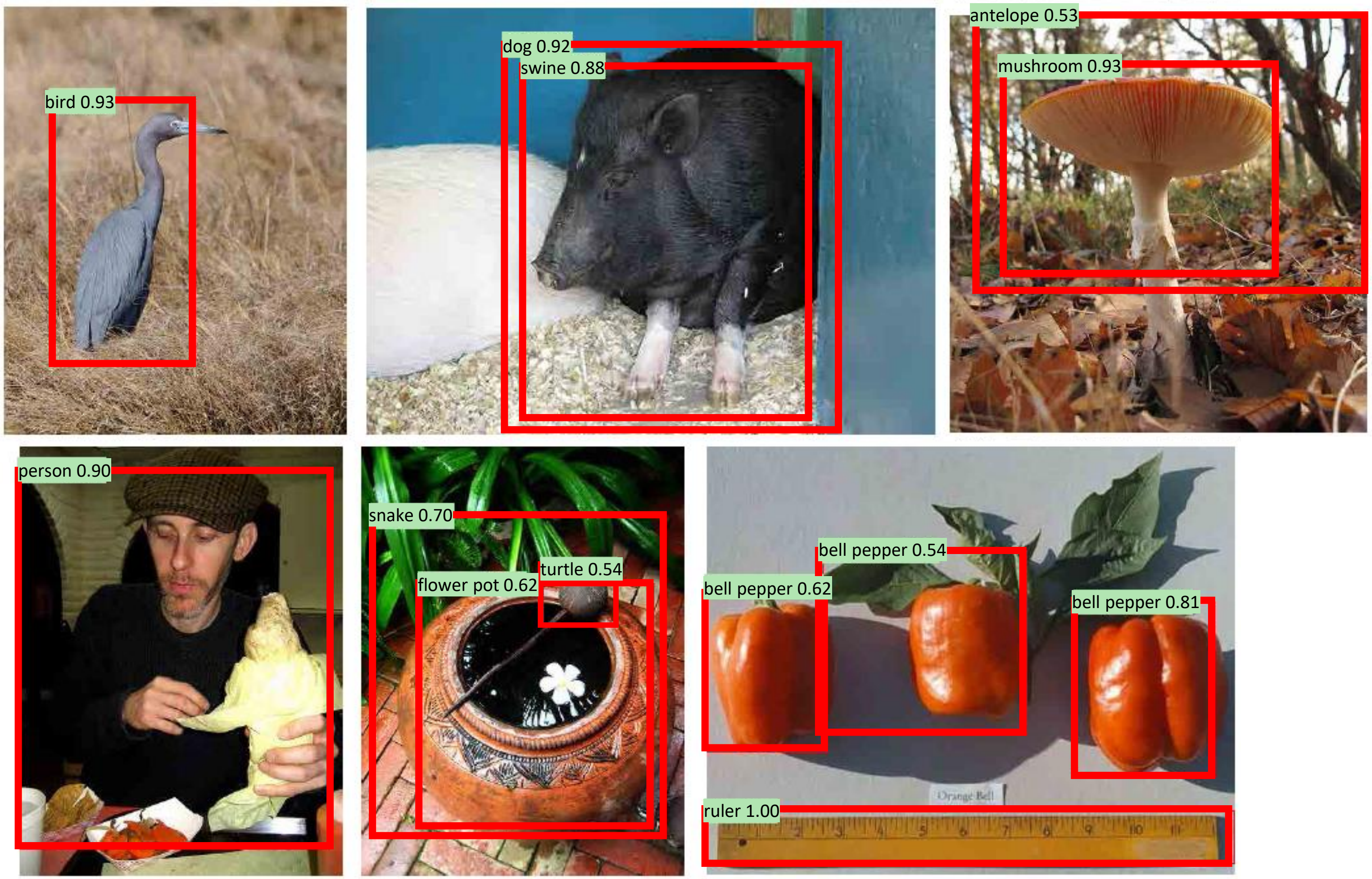
R-CNN testing



- Input test image
- Compute region proposals (Selective Search)
- Each region: run through CNN to predict class labels and bounding box transforms
- “Detections” = regions with highest class confidence scores
  - Based on a threshold, or top k?
  - Overall, or per-category?

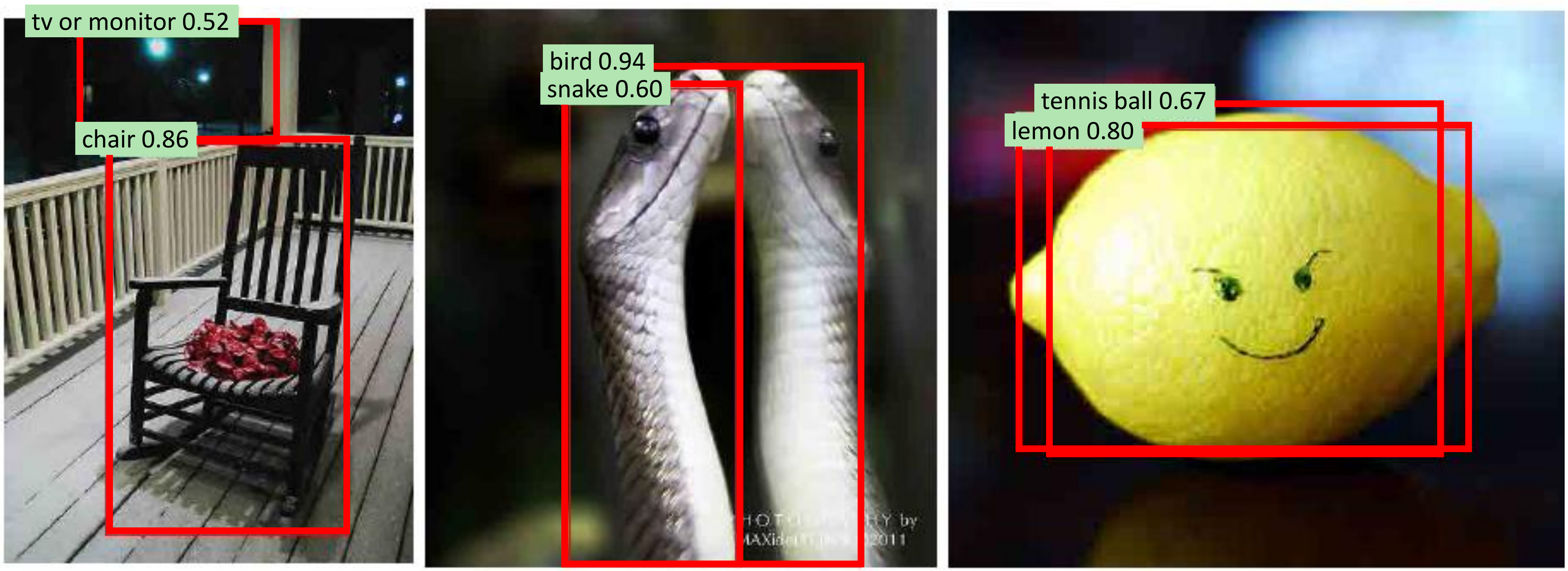
Figure: R. Girshick (2015)

R-CNN results (random sample)



Girshick, Donahue, Darrell, Malik (2014)

R-CNN results (authors' picks)



Girshick, Donahue, Darrell, Malik (2014)

R-CNN summary

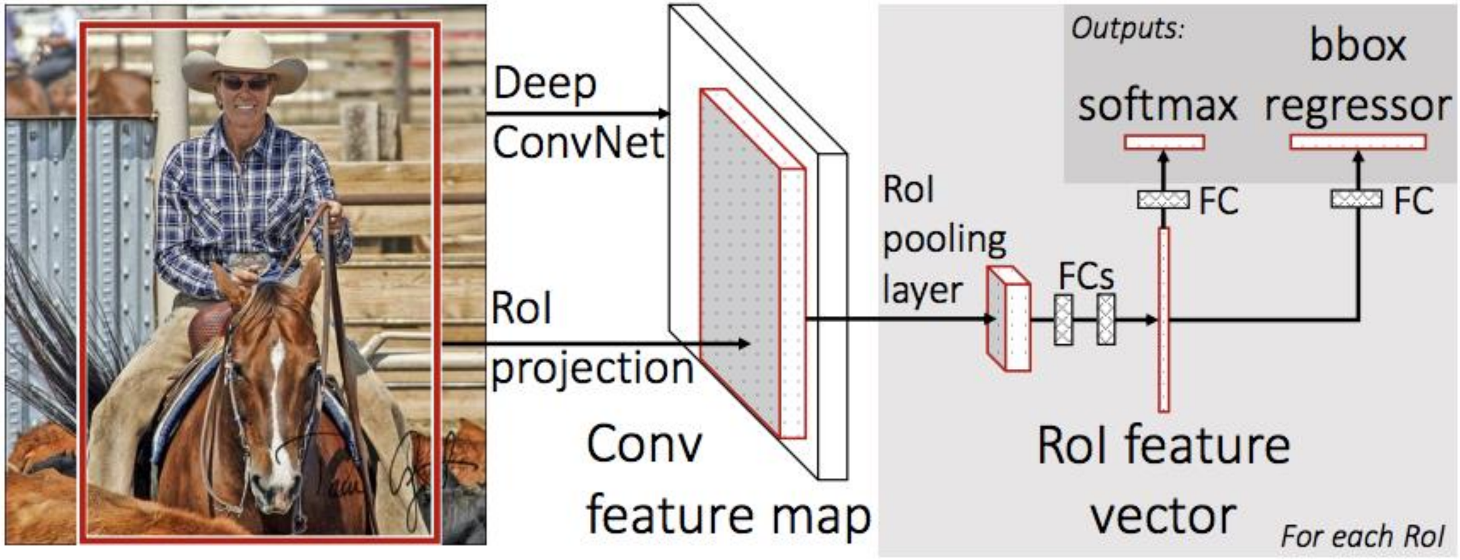
- R-CNN (Region-based convolutional neural network) does classification in parallel over a set of region proposals
- Output: class labels and bounding box transforms
- Advantages
  - Much more efficient than classifying every window
- Disadvantages
  - Still requires classifying many windows (e.g., 2000)
  - Region proposal step could miss some objects

Figure: R. Girshick (2015)

Fast R-CNN

Fast R-CNN

- Major change: run the whole image through a fully-convolutional neural network
- Take region proposals from last convolutional layer



Girshick (2015)

Fast R-CNN

Input image

Figure: R. Girshick (2015)

Fast R-CNN

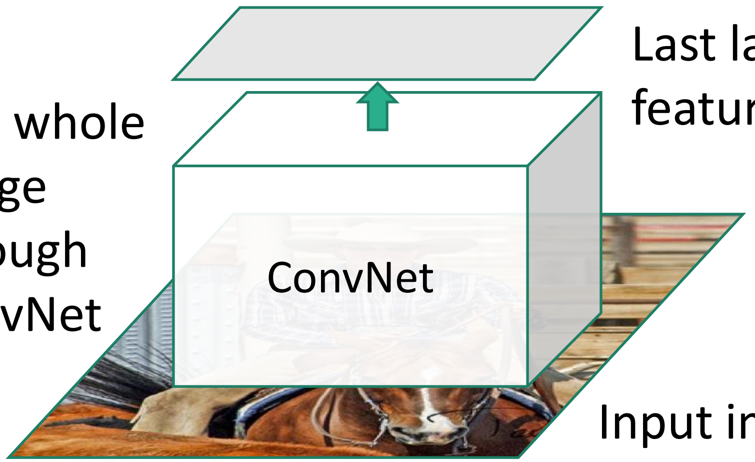
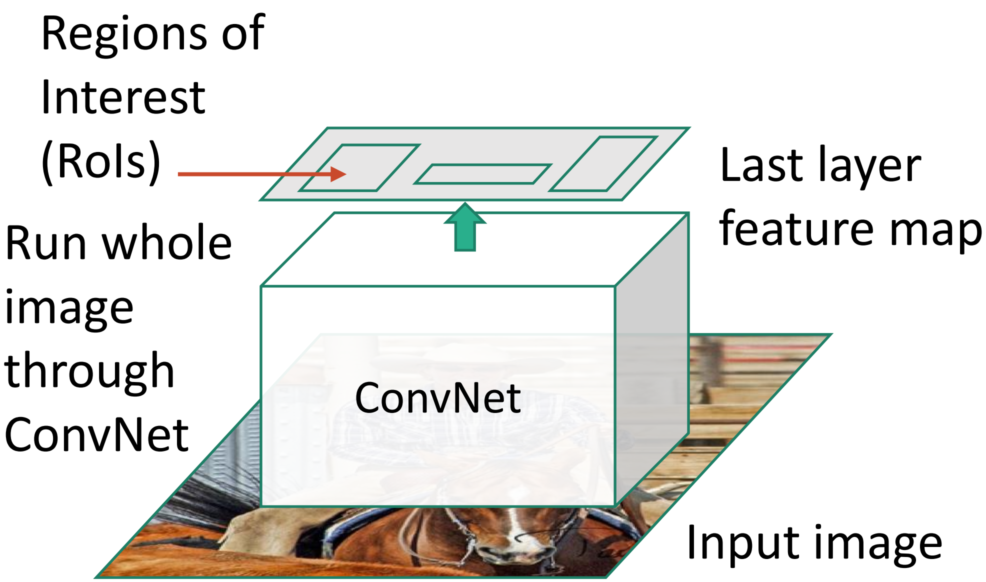


Figure: R. Girshick (2015)

Fast R-CNN



As in R-CNN, Selective Search is used to generate region proposals

Figure: R. Girshick (2015)

Fast R-CNN

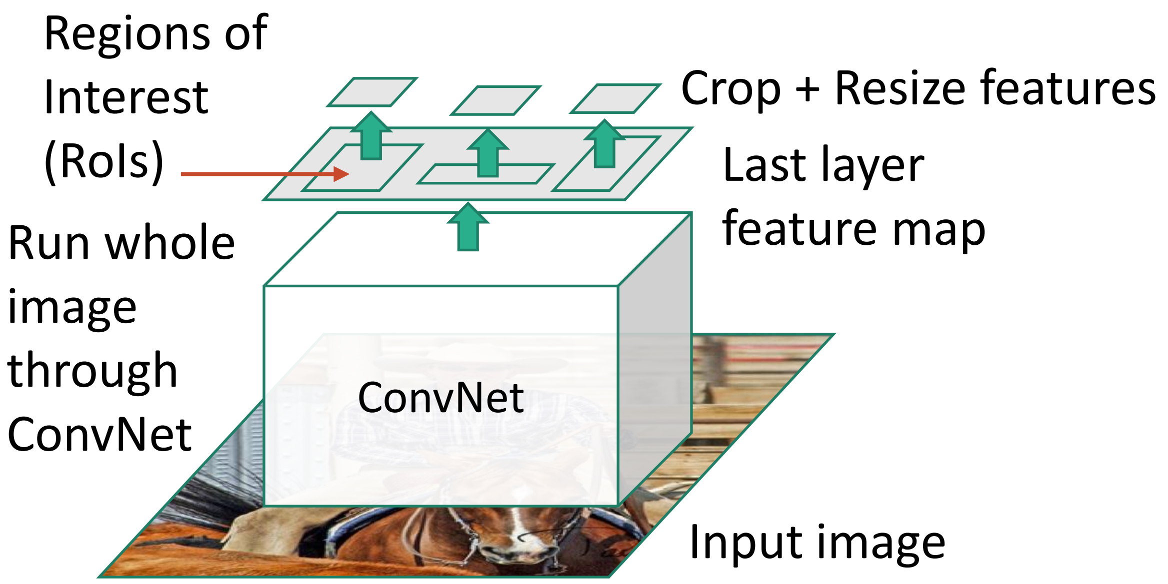
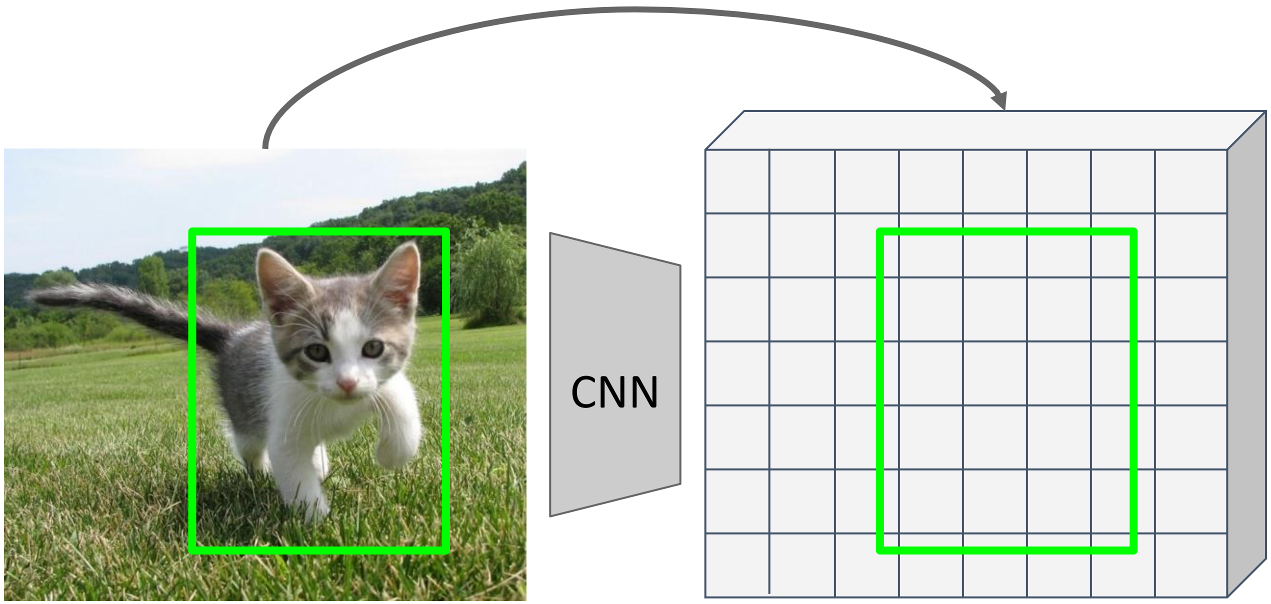


Figure: R. Girshick (2015)

How to crop/resize features?

Map bounding box (pixels) to last layer columns



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

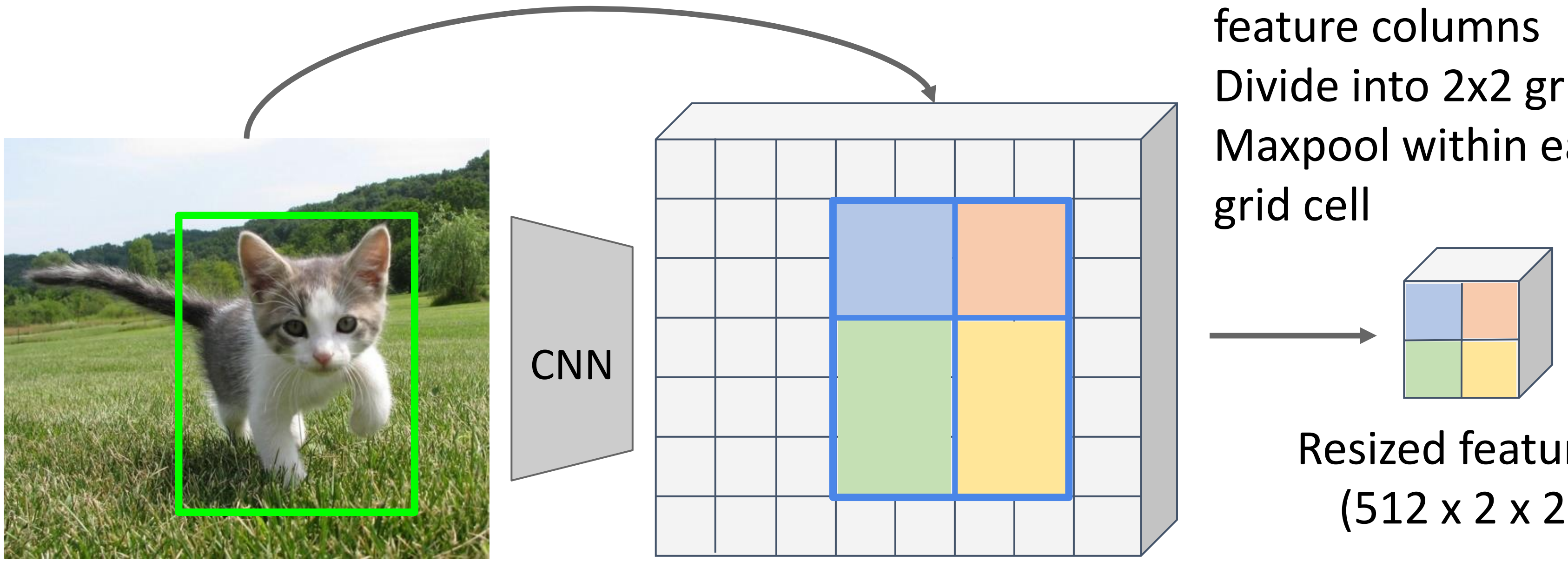
Conv. layer features
(e.g. 512 x 20 x 15)

Slide: J. Johnson

How to crop/resize features?

Map bounding box (pixels) to last layer columns

“Snap” to closest feature columns
Divide into 2x2 grid
Maxpool within each grid cell



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

Conv. layer features
(e.g. 512 x 20 x 15)

Slide: J. Johnson

Fast R-CNN

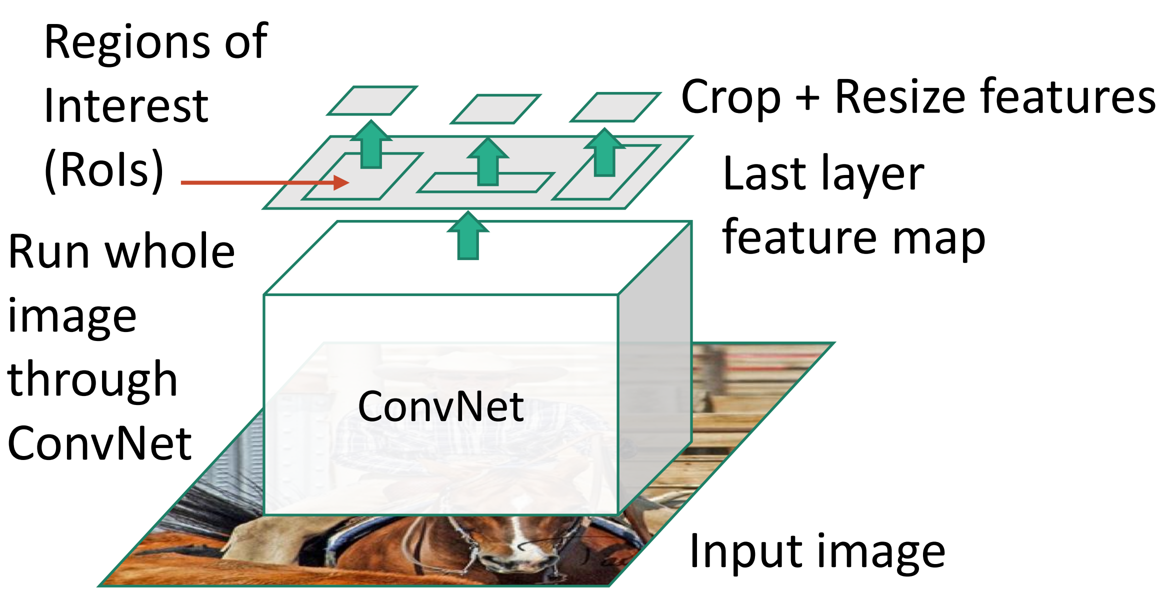


Figure: R. Girshick (2015)

Fast R-CNN

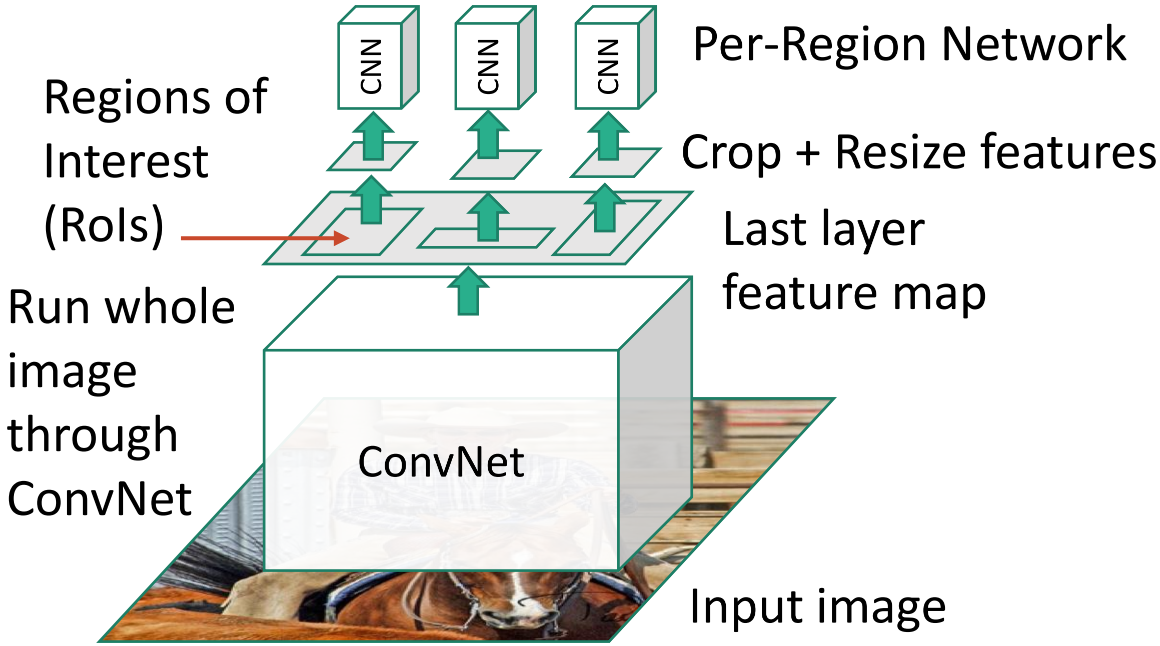


Figure: R. Girshick (2015)

Fast R-CNN

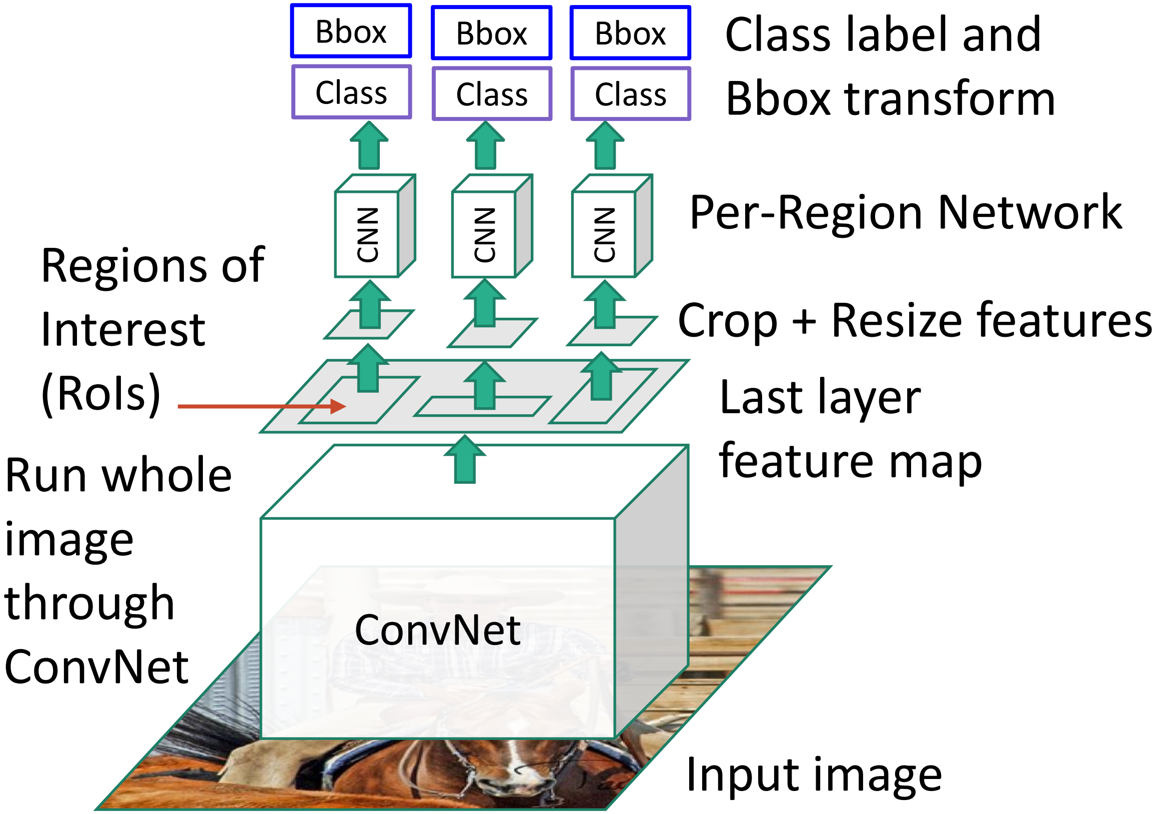
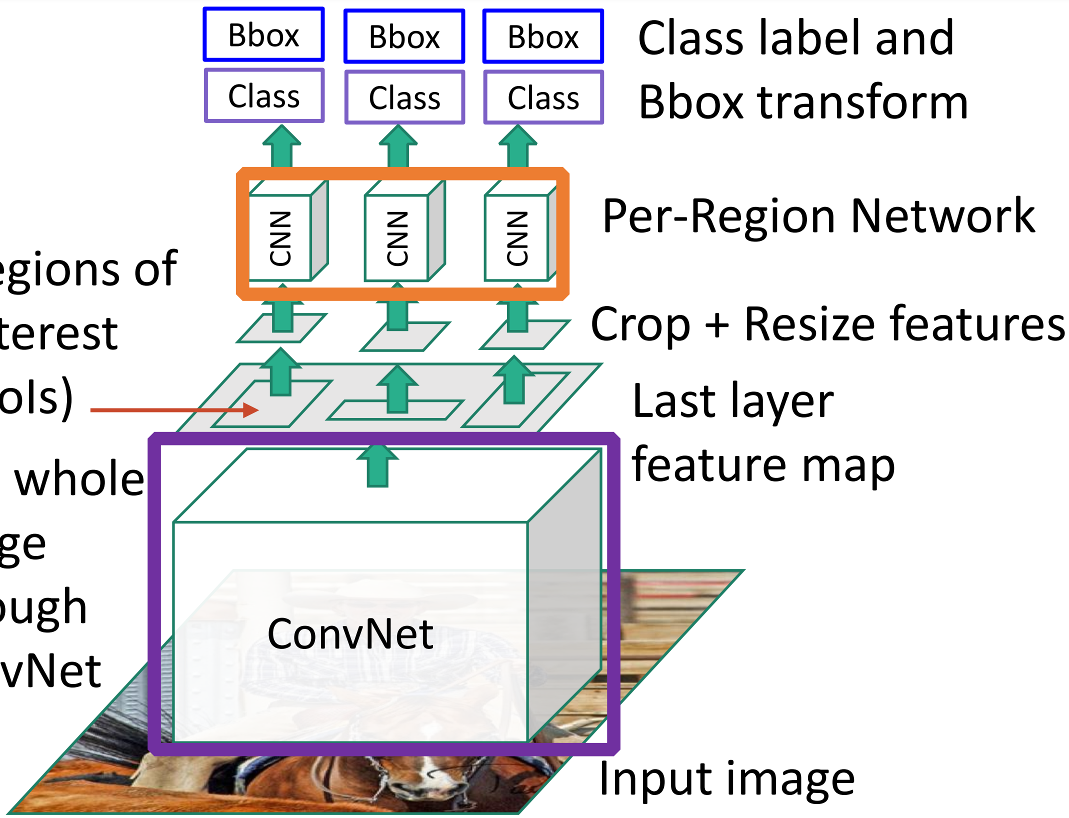


Figure: R. Girshick (2015)

Fast R-CNN



Mapping features to label/bbox is probably not too complex

Backbone network extracts features from image ("feature embedding")

What architecture to use for "backbone" FCN and per-region networks?

Figure: R. Girshick (2015)

Fast R-CNN

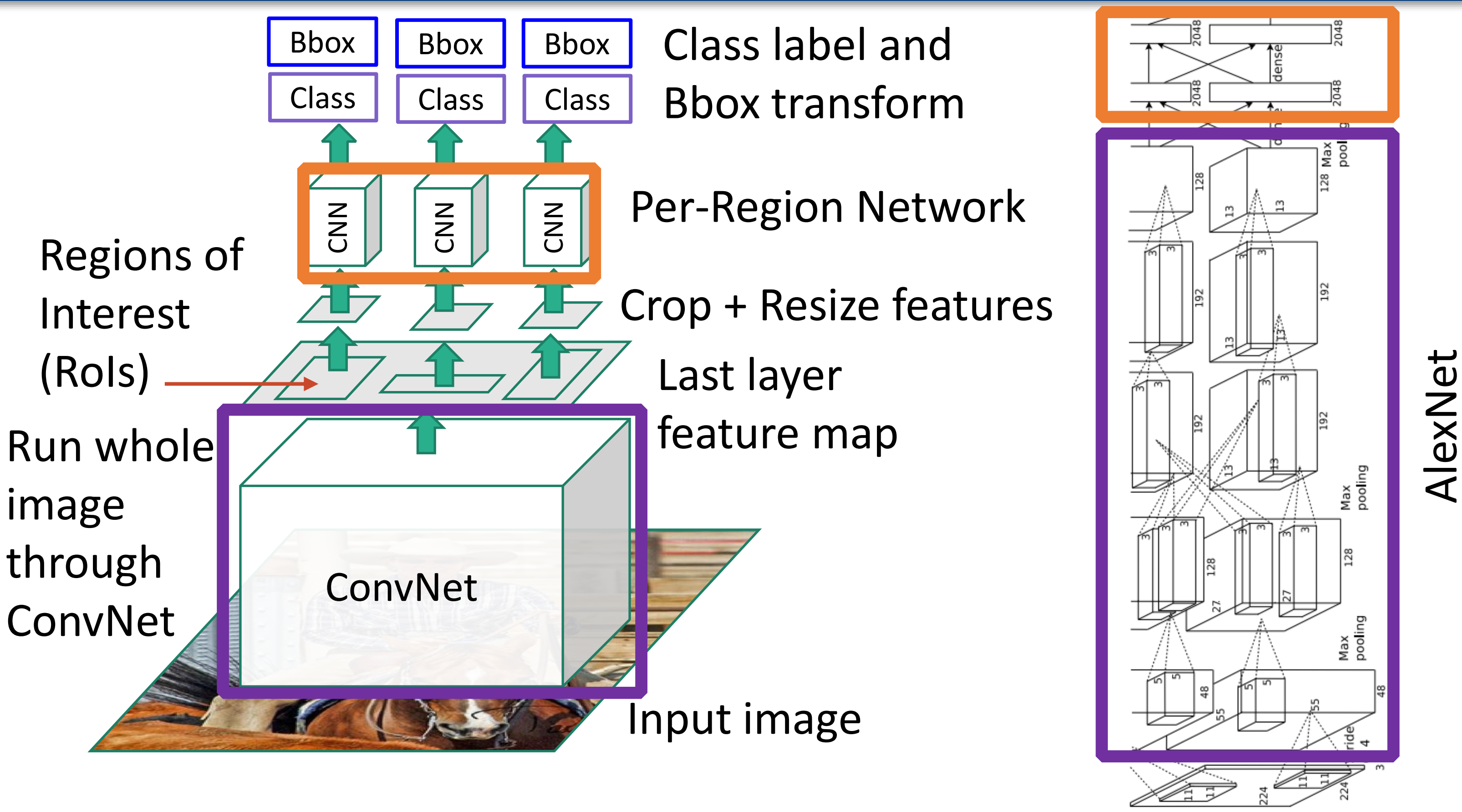


Figure: R. Girshick (2015)

Fast R-CNN

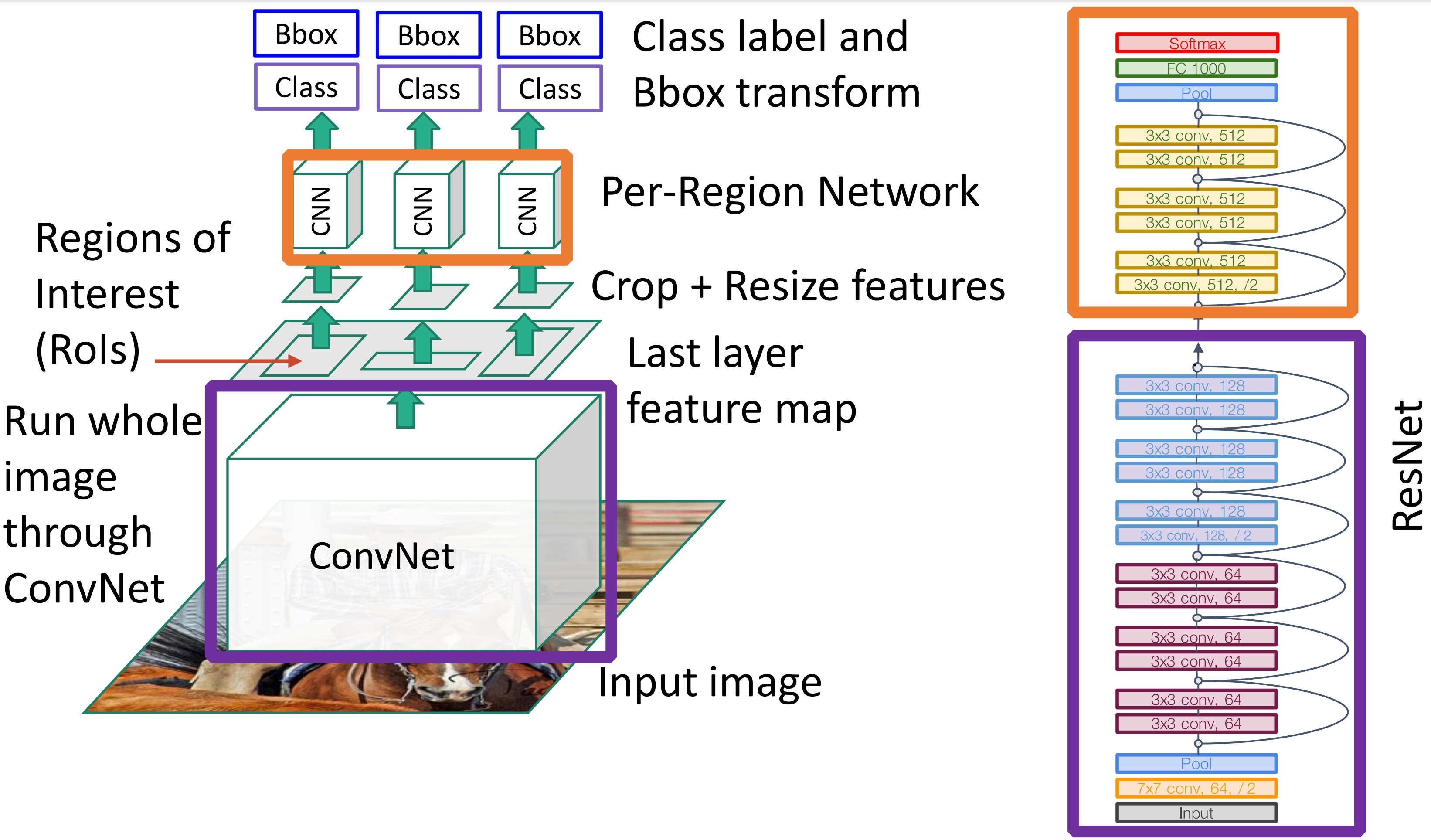
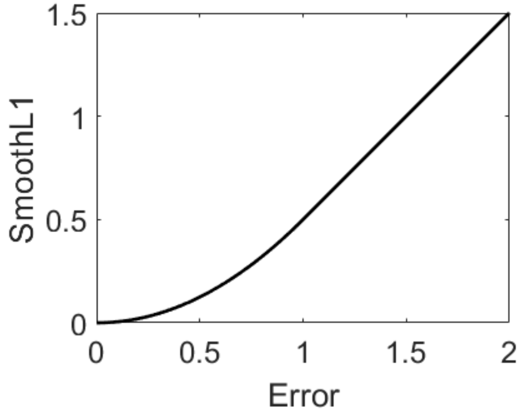


Figure: R. Girshick (2015)

Fast R-CNN training

- Train on R regions sampled from N images
  - For efficiency, N is small (1-2) and R is large (e.g., 64)
  - Sample regions so 75% of training set is “background”
- Train with a multi-task loss: $L = L_{cls} + L_{loc}$
  - L_{cls} = cross-entropy loss over labels
  - L_{loc} = SmoothL1 of abs(true-predicted bbox parameter)
- L_{loc} computed for object classes only



Fast R-CNN summary

- End-to-end region-based convolutional neural network
- Advantages:
  - Faster than R-CNN (~9x faster training, ~140x faster test)
  - Slightly more accurate than R-CNN
- Disadvantage:
  - ROIs aren't learned; region proposal step could miss some objects

Faster R-CNN

Faster R-CNN

- Major change: network learns region proposals, instead of using Selective Search

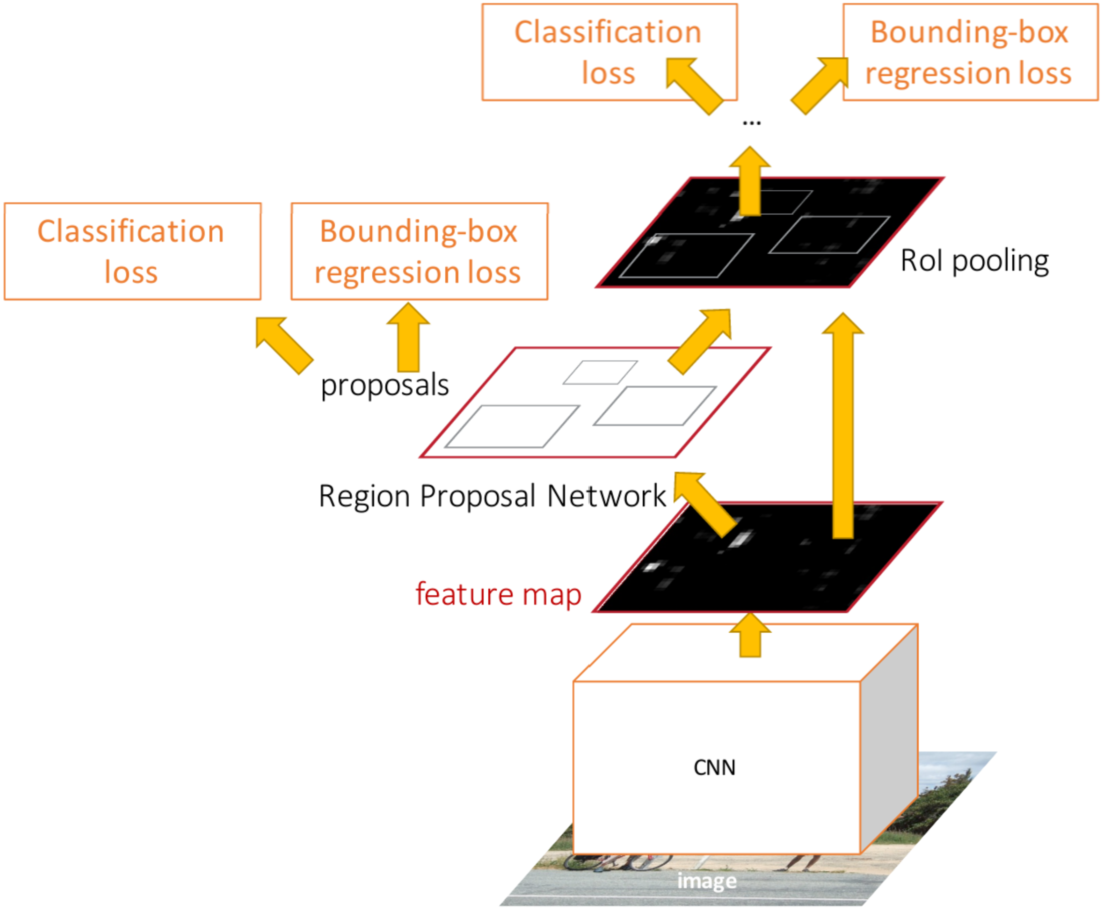
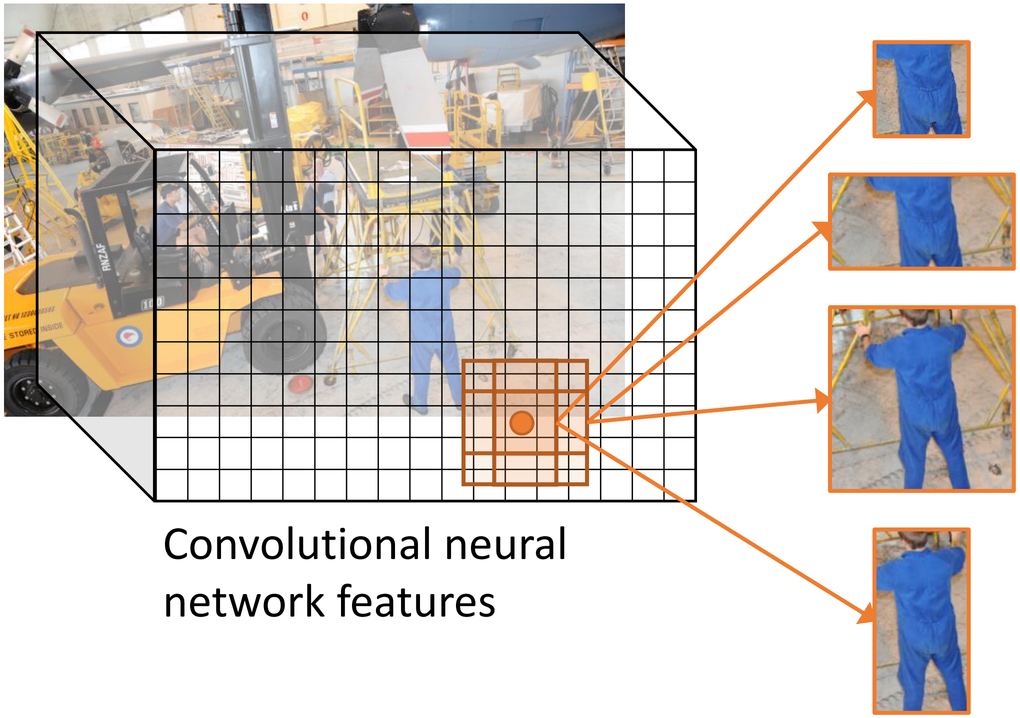


Figure: Ren, He, Girshick, & Sun (2015)

Region proposal network (RPN)



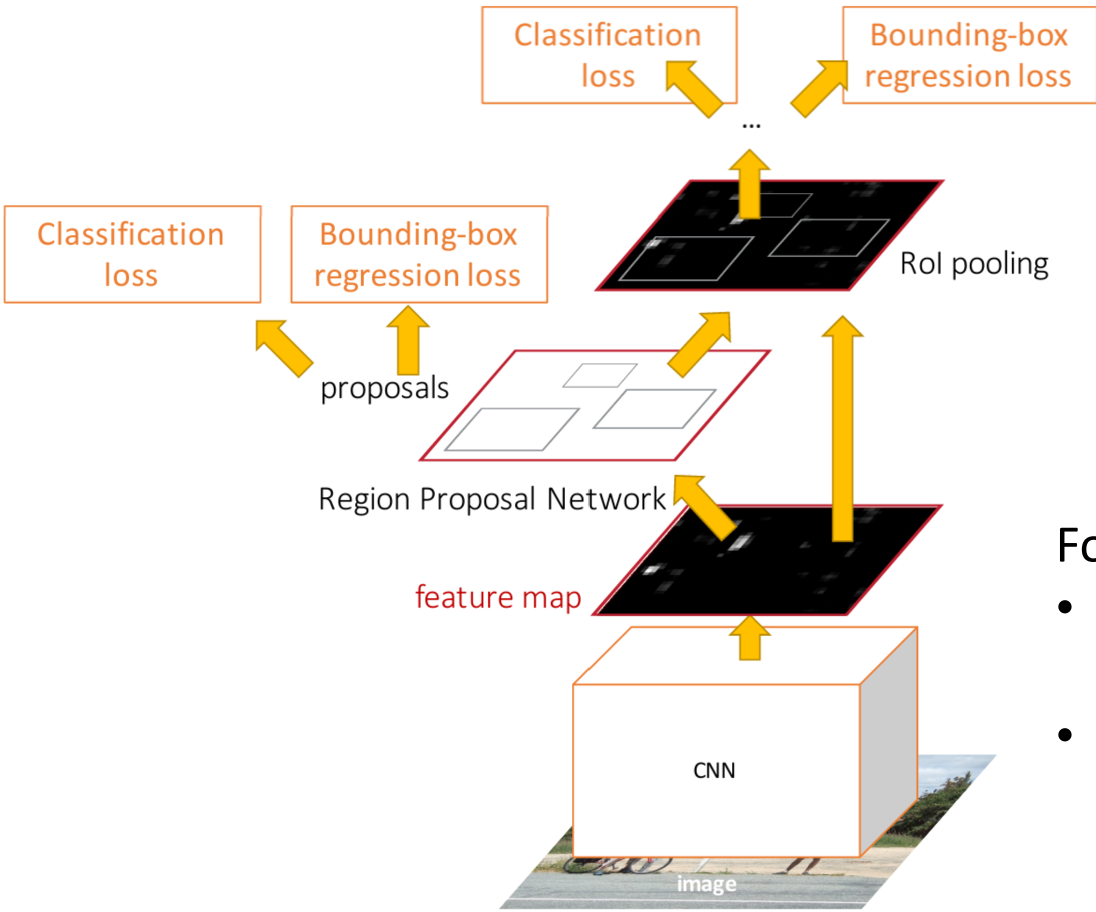
Convolutional neural network features

An "anchor point" is placed at each column in the feature map

Each anchor point generates k regions of fixed size and aspect ratio

In each region, predict object class and bounding box transform

Faster R-CNN



For each region:

- Crop & resize features
- Predict object class and bbox transform

For each image:

- Run backbone CNN to get feature map
- Compute region proposals from RPN

Figure: Ren, He, Girshick, & Sun (2015)

Faster R-CNN training

- RPN loss is weighted sum of:
  - Classification loss: binary cross entropy loss (any object vs. background)
  - Regression loss: SmoothL1 between true and predicted bbox parameters
- As in Fast R-CNN, training samples R regions (anchors) from N images (R = 256, N = 1)
  - Anchors are sampled so up to 50% are objects
- Full network is RPN + Fast R-CNN (sharing a backbone)
  - Various ways to train this, but original method alternates between training RPN and Fast R-CNN

Faster R-CNN Summary

- Faster R-CNN is similar to Fast R-CNN but learns the region proposals with a region proposal network (RPN)
- Even faster than fast R-CNN (~10x faster test)
- Modular approach – variations on Faster R-CNN with deep backbone tend to be quite accurate
  - Speed-accuracy trade-off: deeper networks are also slower

Summary

- Object detection = classification of image regions
- Exhaustive search is slow; most methods use only a subset of regions (“region proposals” or “regions of interest (ROIs)”)
- Many parameters to consider:
  - What counts as a true detection / true rejection (IoU threshold)?
  - How to select region proposals?
  - How to deal with class imbalance? (“background” is most common class)