Trustworthy AI Systems

-- Image Segmentation

Instructor: Guangjing Wang

guangjingwang@usf.edu

Group Member

- Two to three students will form a group
- Midterm project: any machine learning application projects
- Final project: evaluating midterm project based on (ONE or TWO) trustworthy AI principles
- 09/08 Group Checkpoints: providing names of your teammates

Quizzes and Slides

- Each open-book quiz will contain 25 single choice questions in 50 minutes with pen and paper.
  - You are not required to memorize or recite everything in the lecture
  - You need to understand points in the lecture: what, why, how
  - You are expected to spend more time beyond the lectures e.g., reading papers, checking the open-source code, API documentation...
- Be a graduate student
  - The learning style changes compare to your undergraduate study
  - There is no required homework or exercise...
  - You need to learn how to learn, how to practice...
- Slides are shared on Canvas

Paper Review (Not a Homework)

- Paper review is a basic task for a researcher
  - Paper Summary
  - Strengths
  - Weaknesses
  - Questions
  - Future Opportunities

When you read a paper, thinking:

- What are the research problem and motivation?
- What are the challenges and technical contributions?
- How is the experimental evaluation?
- How are the related work and overall writing?

Last Lecture

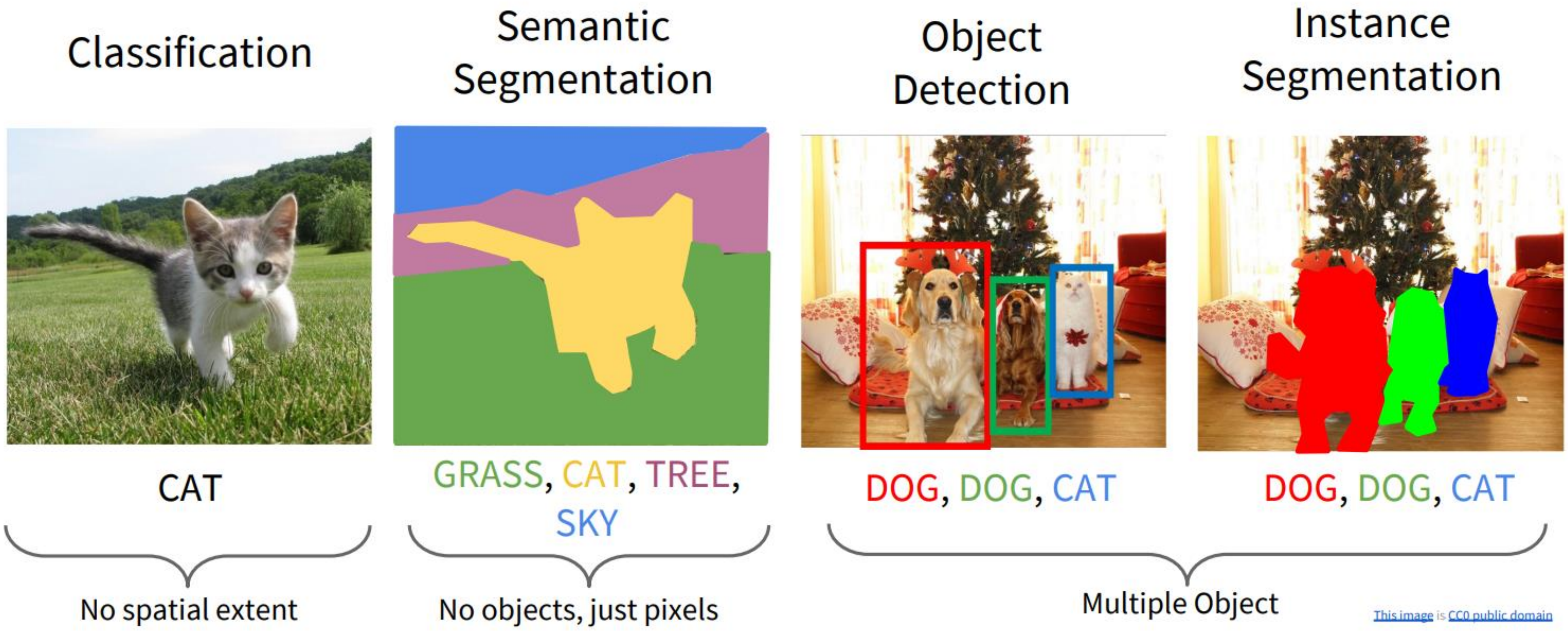
- An image of dimensions $W_{in} \times H_{in}$.
- A filter of dimensions $K \times K$.
- Stride S and padding P.

Shape of output activation map

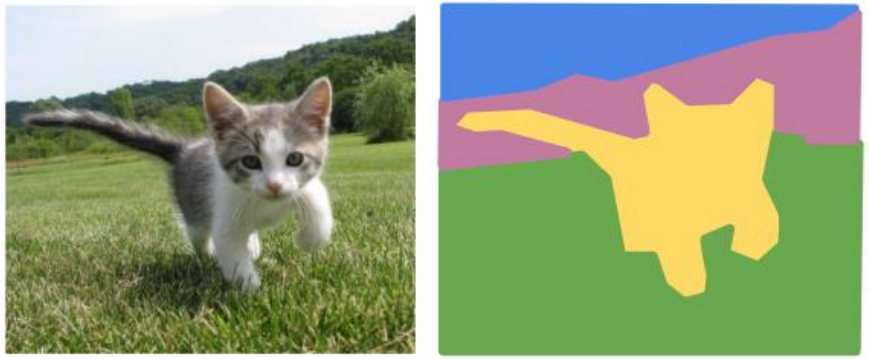
$$\mathbf{W_{out}} = \frac{\mathbf{W_{in} - K + 2P}}{\mathbf{S}} + \mathbf{1}$$
$$\mathbf{H_{out}} = \frac{\mathbf{H_{in} - K + 2P}}{\mathbf{S}} + \mathbf{1}$$

- Image classification
  - Can be extend to any classification problems
- Convolutional neural network
  - The key components: convolution, pool, activation, normalization
  - The general structure design of CNN, e.g., ResNet
- Some practices for project
  - Data preprocessing
  - Transfer learning
  - Regularization
  - Hyperparameter tunning during training

Computer Vision Tasks

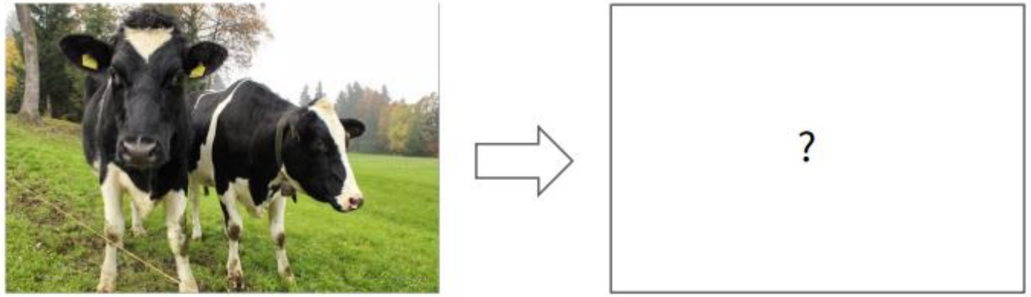


Semantic Segmentation: Problem



GRASS, CAT, TREE,
SKY, ...

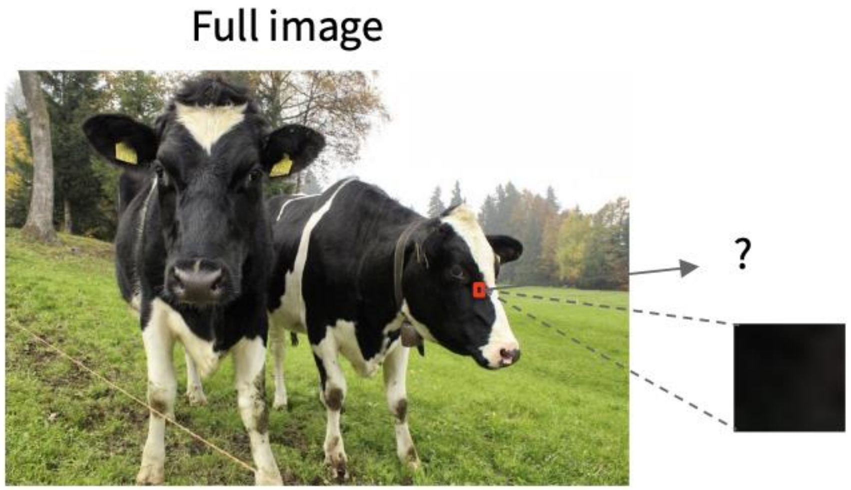
Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Label each pixel in the image with a category label.

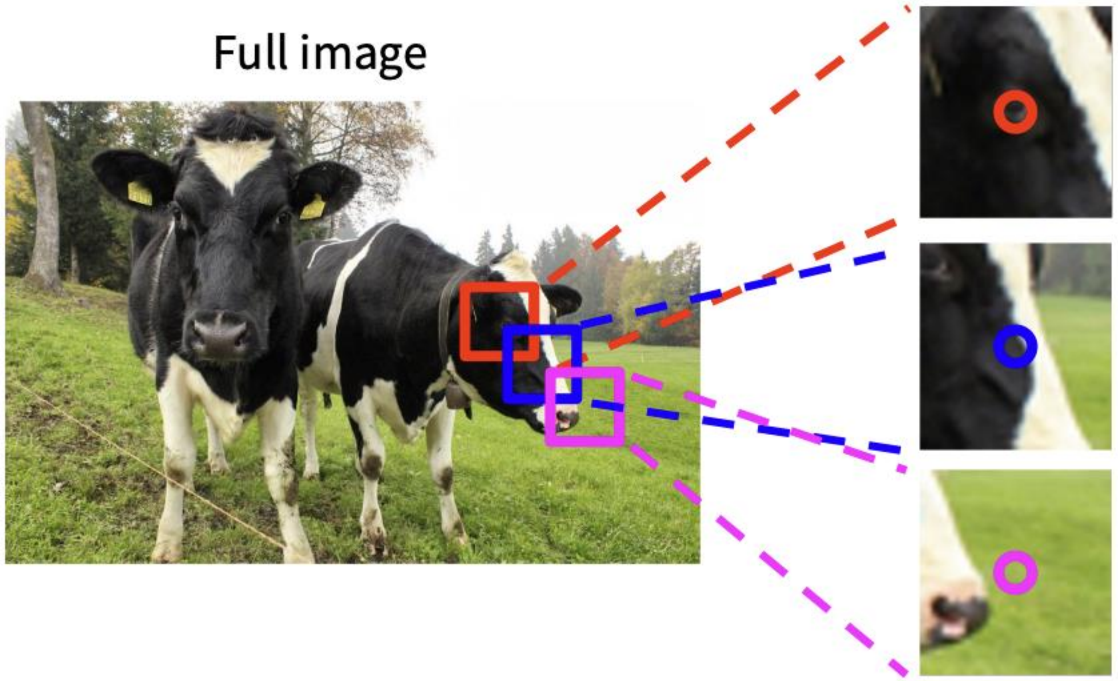
Semantic Segmentation: Classification Problem



Classify each pixel

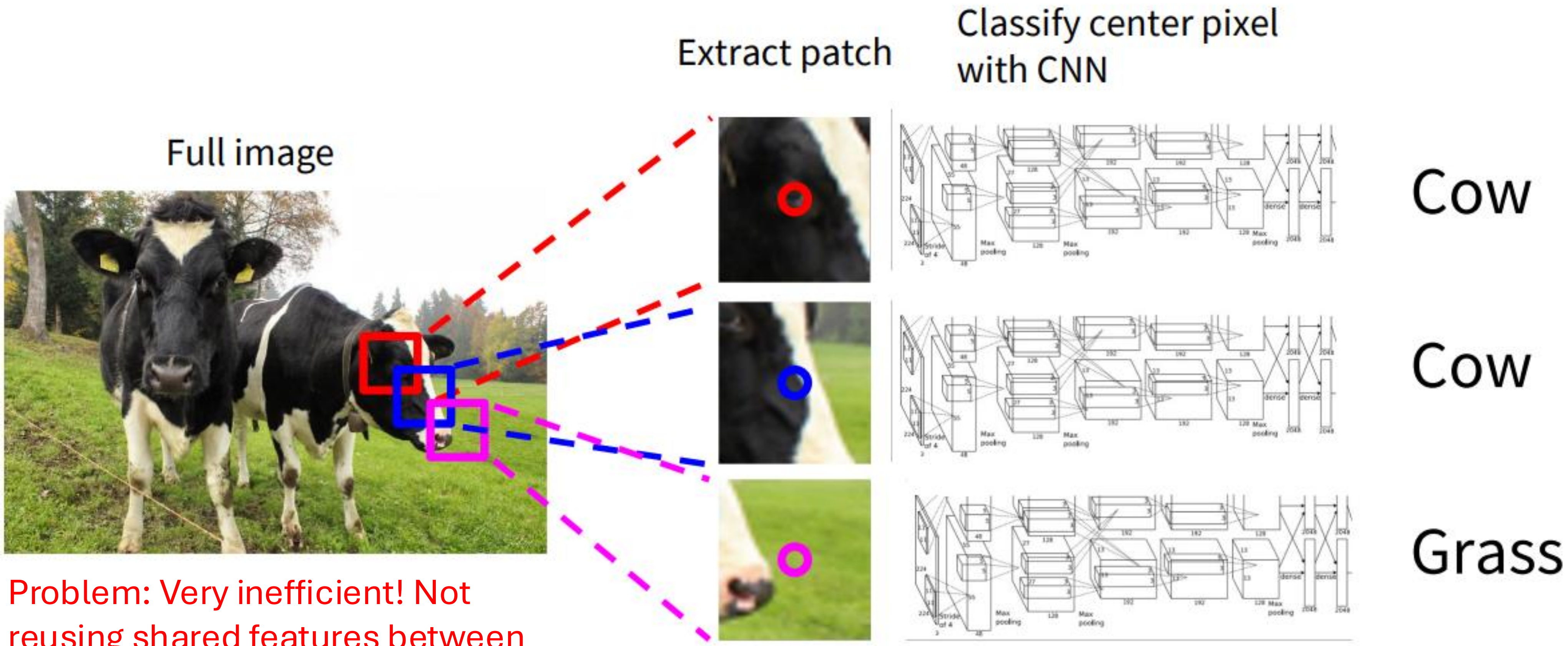
- Impossible to classify without the context
- How do we include context information?

Semantic Segmentation Idea: Sliding Window



How do we model context information?

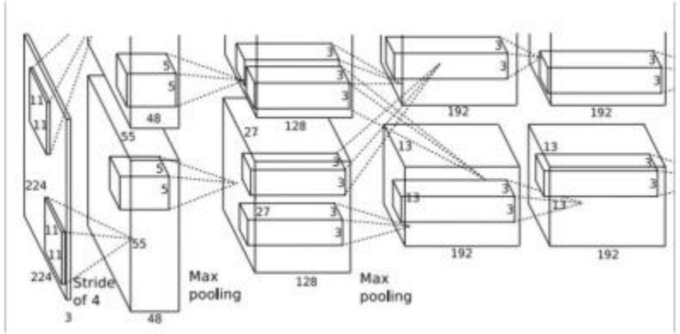
Semantic Segmentation Idea: Sliding Window



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

Semantic Segmentation: Convolution (1)

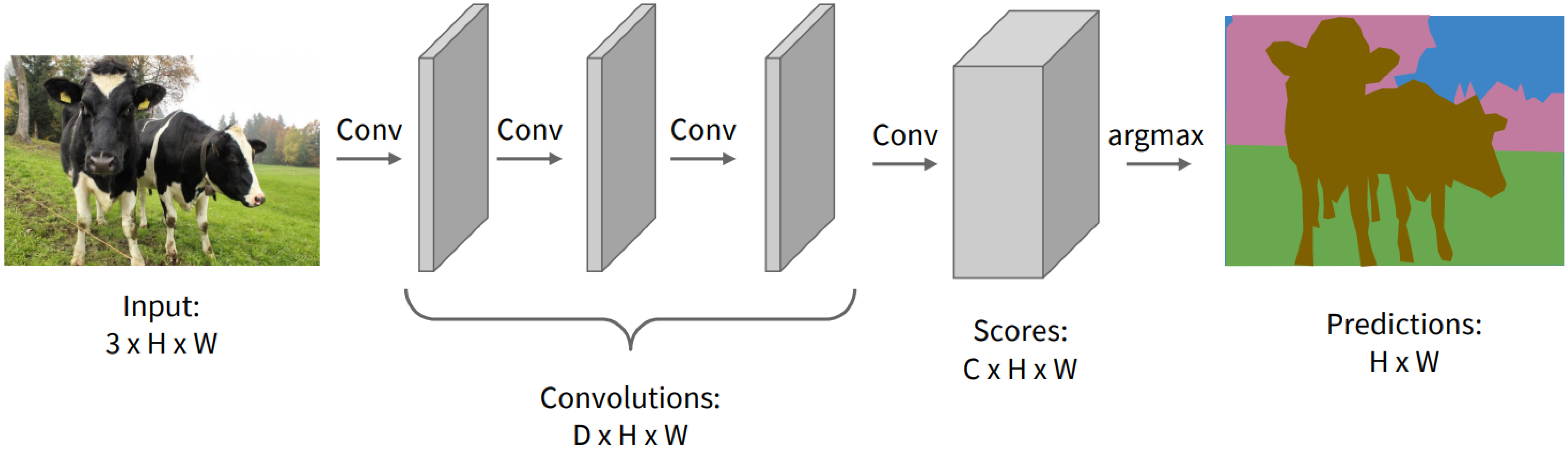
Full image



Encode the entire image with conv net, and do semantic segmentation on top

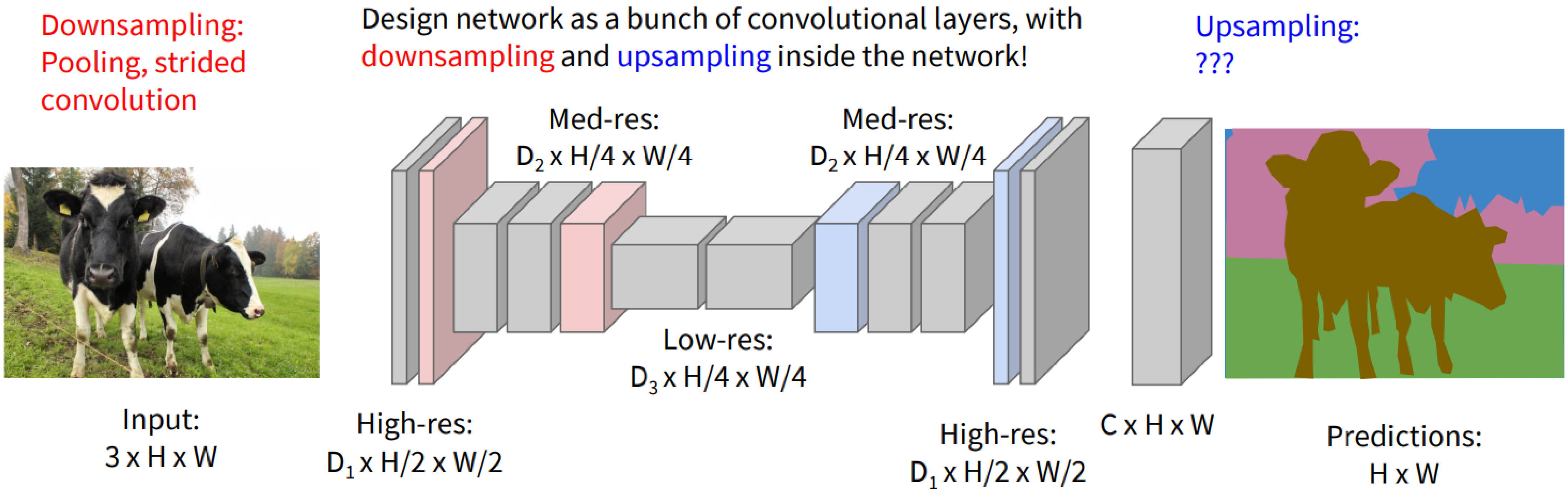
Potential problem? (hint: input shape, output shape)

Semantic Segmentation: Convolution (2)



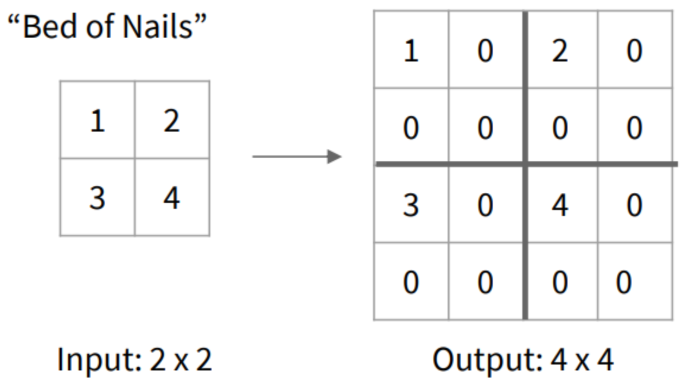
- Do not use the down-sampling operators
- Potential problem? (hint: computation)

Semantic Segmentation: Convolution (3)

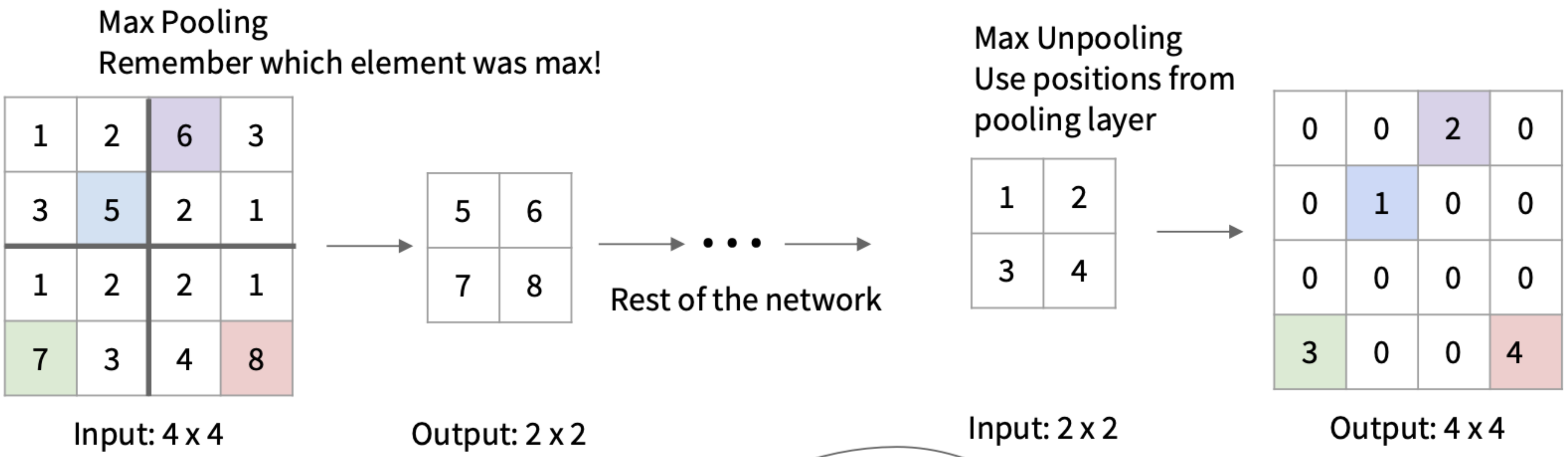


Upsampling

- Non-learnable upsampling
  - Fill the same
  - Fill zeros
  - Max Unpooling
  - You design it…
- Learnable upsampling
  - Transposed convolution



Max Unpooling: Remember location then fill



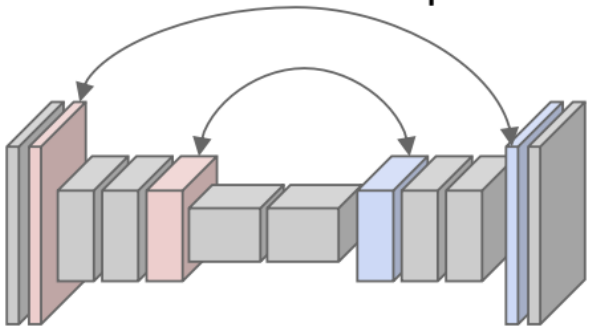
1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

5	6
7	8

1	2
3	4

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

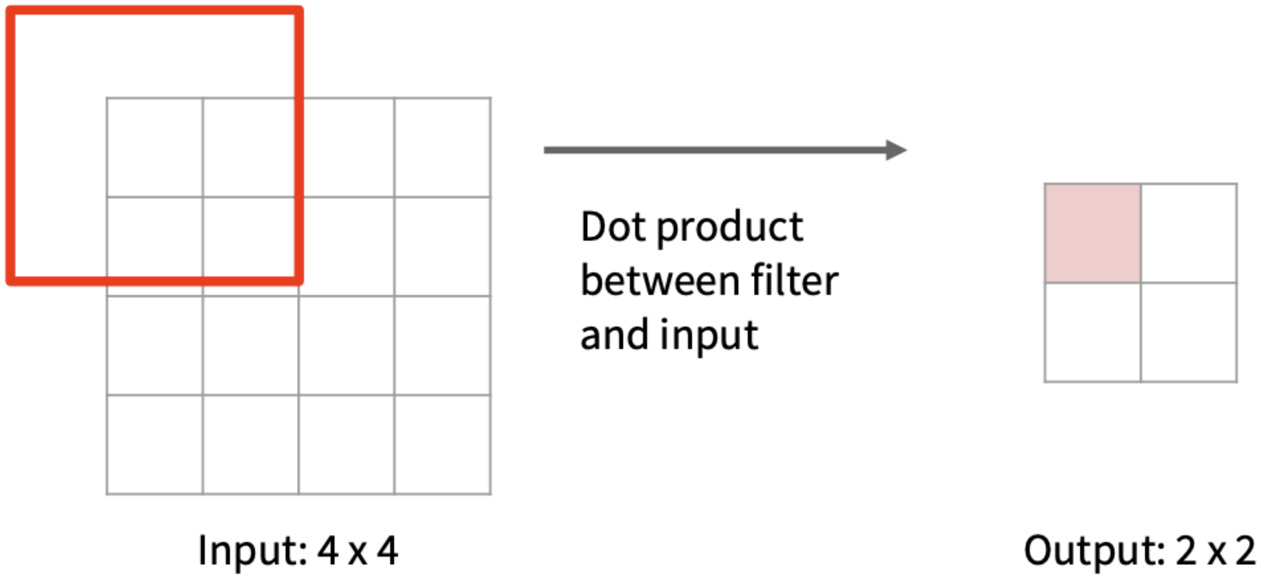
Corresponding pairs of downsampling and upsampling layers



Recall the Convolution Operation

3x3 convolution: Filter size/kernel size: 3x3

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



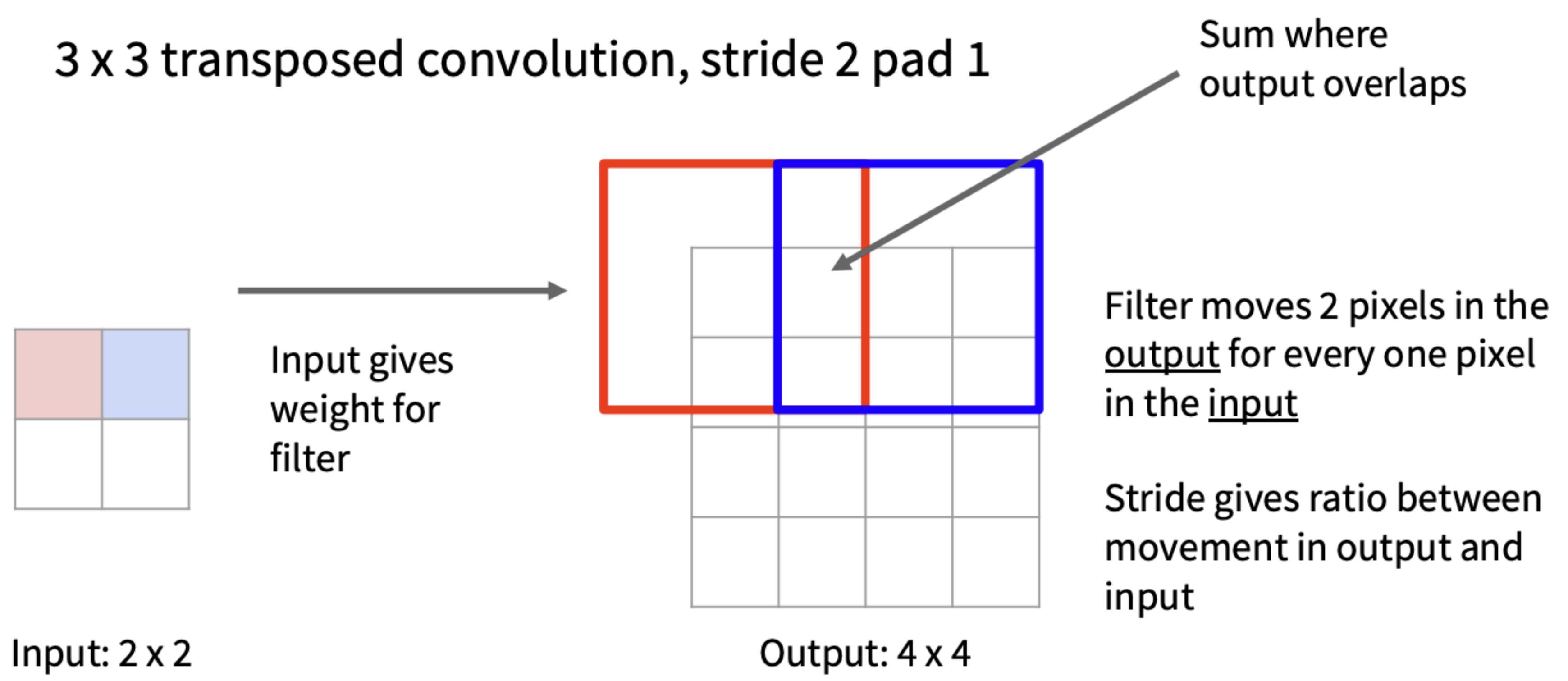
Stride gives ratio between movement in input and output

$$\mathbf{W_{out}} = \frac{\mathbf{W_{in} - K + 2P}}{\mathbf{S}} + \mathbf{1}$$

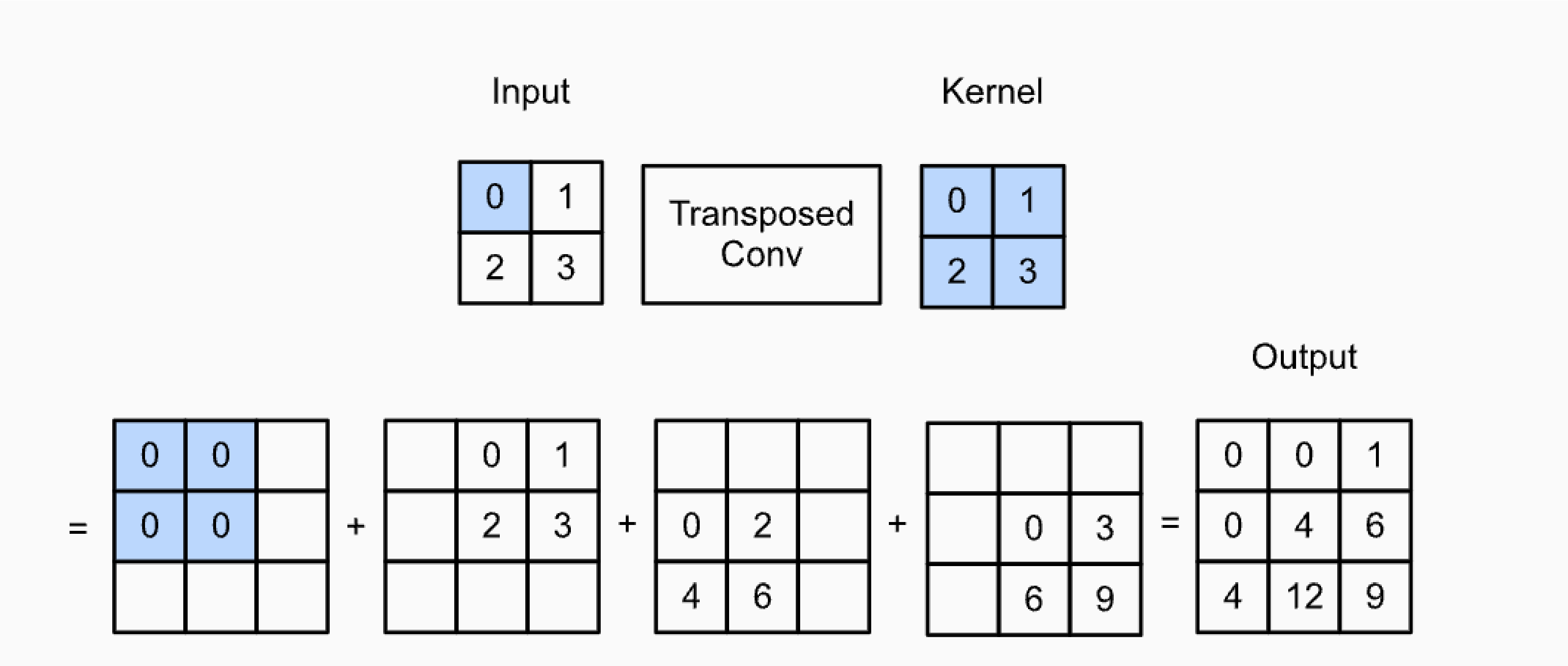
$$\mathbf{H_{out}} = \frac{\mathbf{H_{in} - K + 2P}}{\mathbf{S}} + \mathbf{1}$$

We can interpret strided convolution as "learnable downsampling"

Upsampling: Transposed Convolution

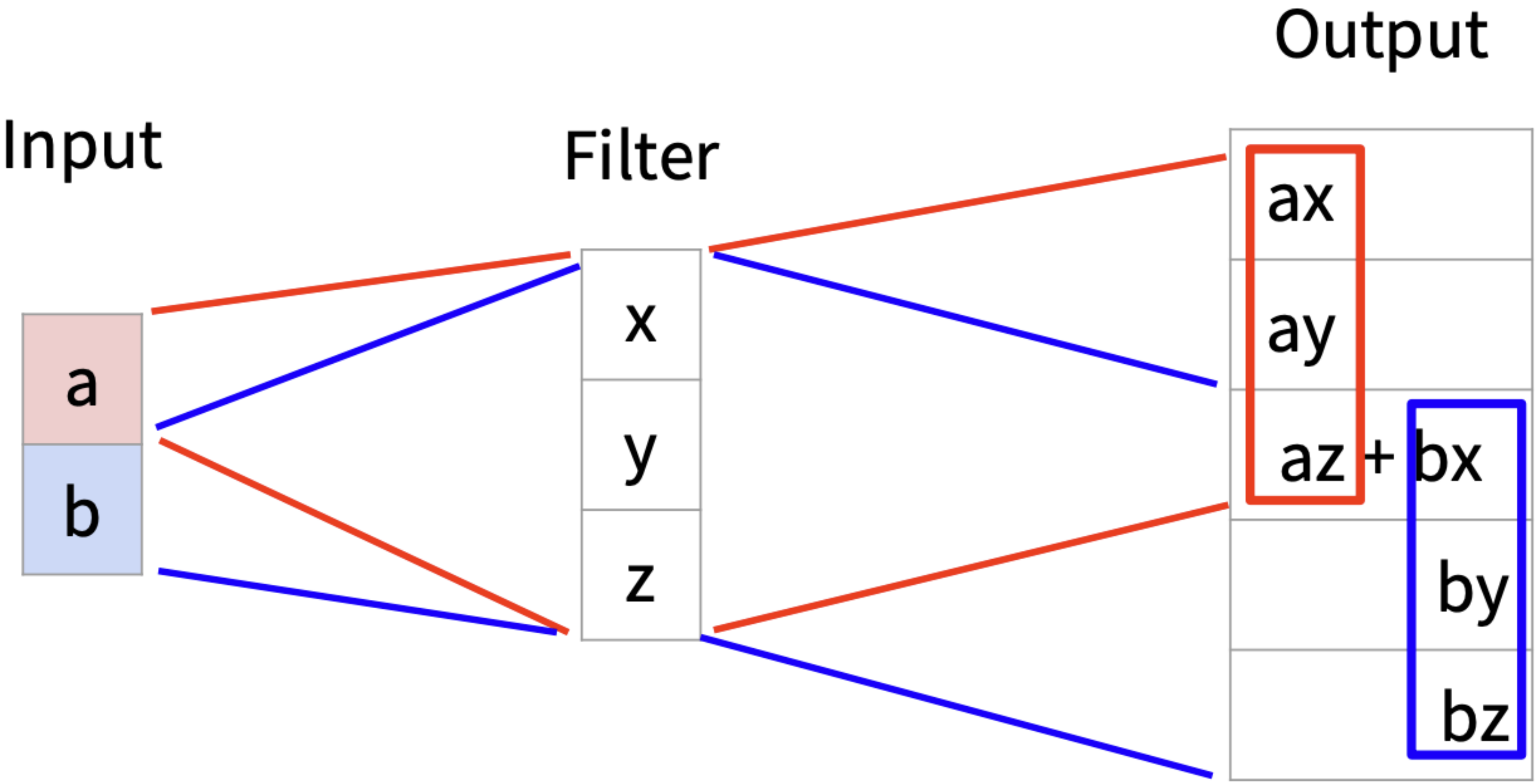


Transposed Convolution Example



Transposed convolution with a 2×2 kernel

Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Convolution as Matrix Multiplication

We can express convolution in terms of a matrix multiplication

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

kernel

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

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

Transposed convolution multiplies by the transpose of the same matrix:

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

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

Example: 1D transposed conv, kernel size=3, stride=2, padding=0

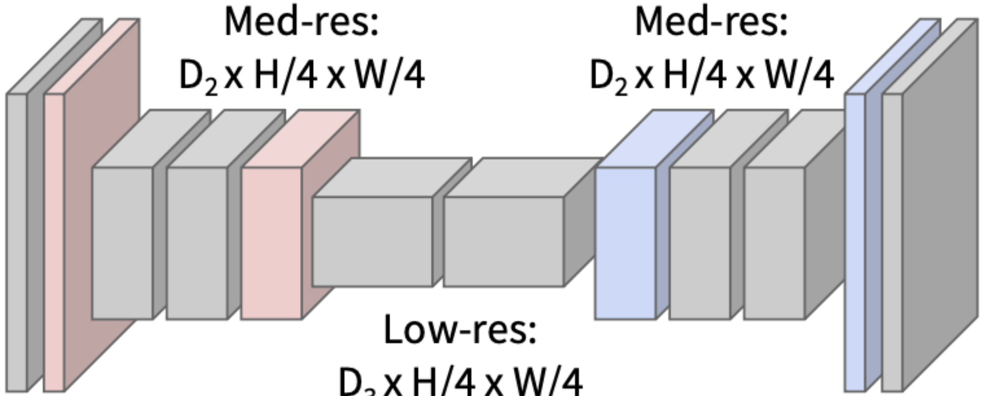
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transposed convolution

Input:
$3 \times H \times W$



Med-res:
$D_2 \times H/4 \times W/4$

Med-res:
$D_2 \times H/4 \times W/4$

Low-res:
$D_3 \times H/4 \times W/4$

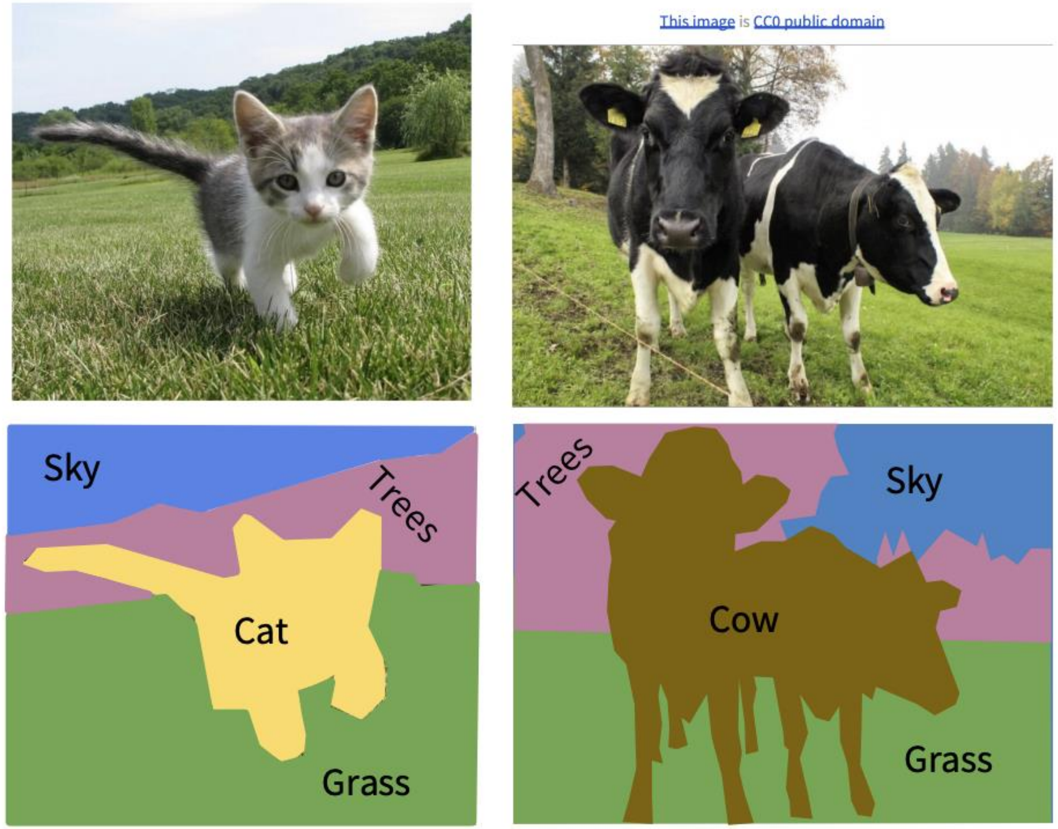
High-res:
$D_1 \times H/2 \times W/2$

High-res:
$D_1 \times H/2 \times W/2$

Predictions:
$H \times W$

Semantic Segmentation

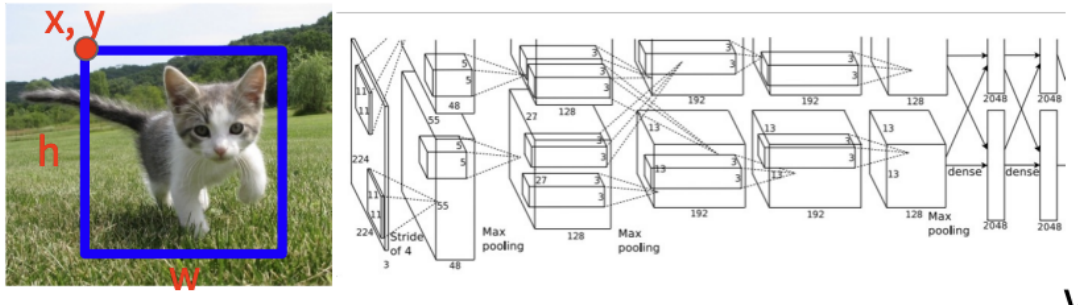
- Label each pixel in the image with a category label
- Don’t differentiate instances, only care about pixels



Take a break

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

Object Detection: Classification + Regression



This image is CC0 public domain

1000 categories

Fully Connected: 4096 to 1000

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

Softmax Loss

Multitask Loss

+ → Loss

Vector: 4096

Fully Connected: 4096 to 4

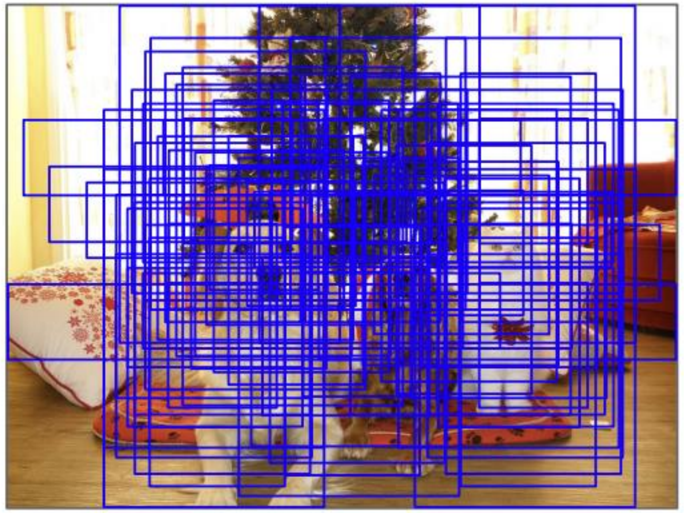
Box Coordinates (x, y, w, h) → L2 Loss

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

Treat localization as a regression problem!

Object Detection

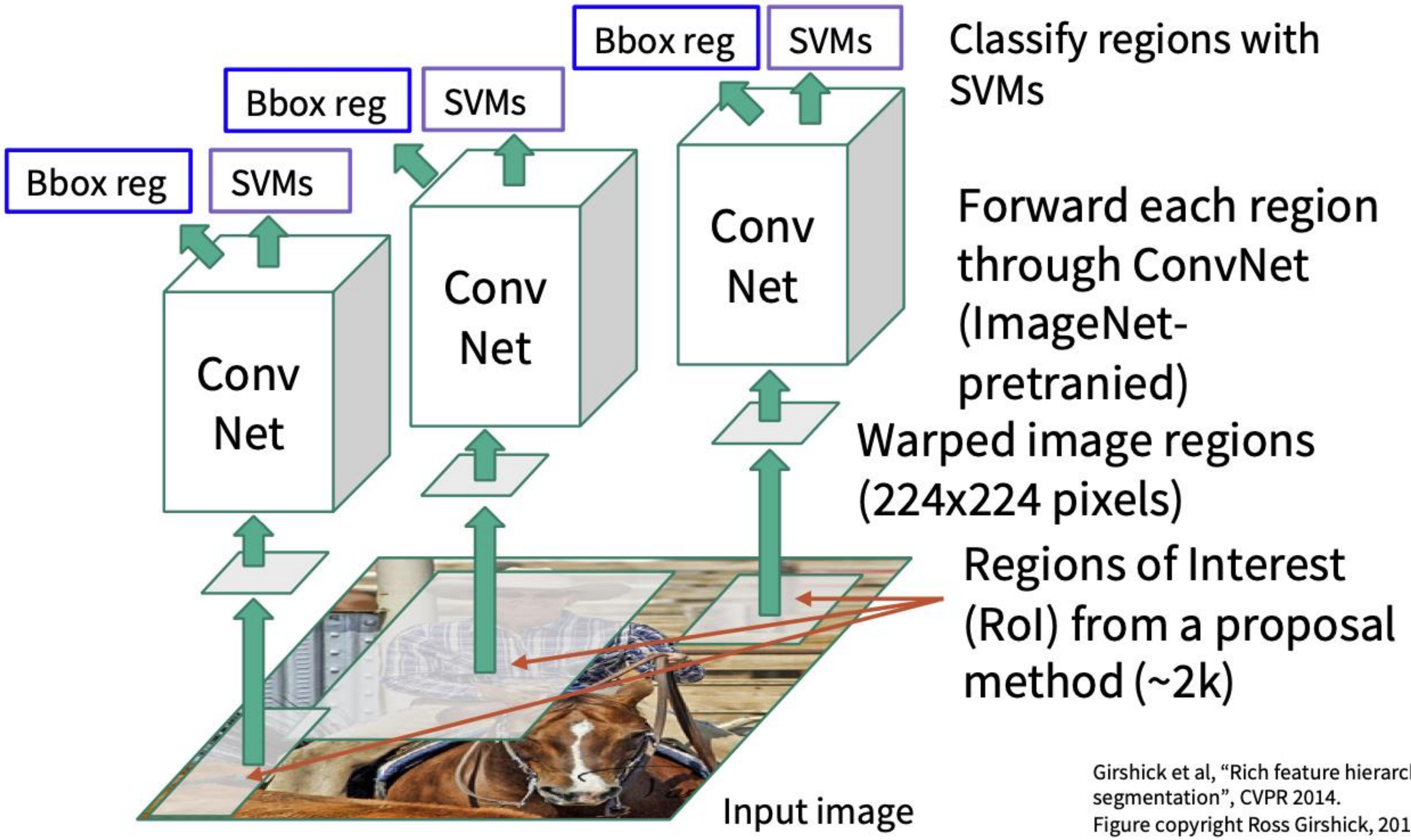
- What if there are multiple objects?
  - Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

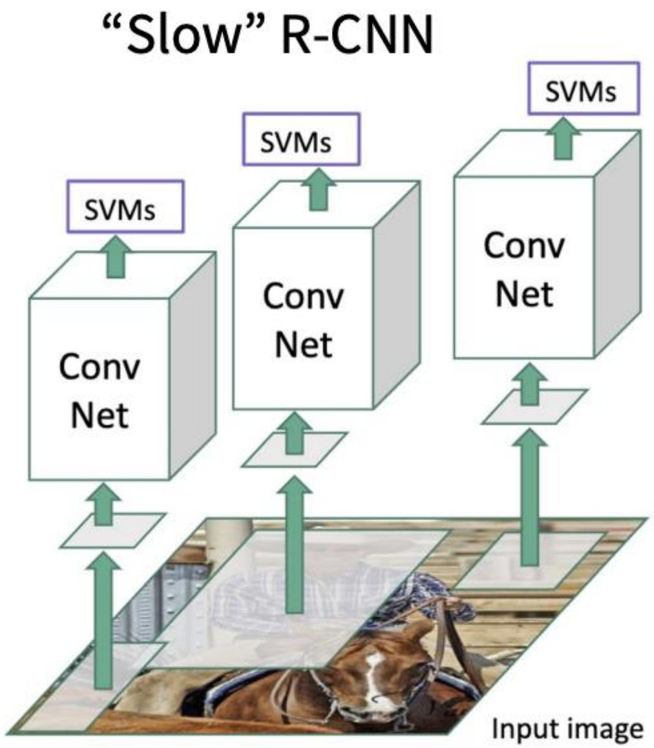
R-CNN

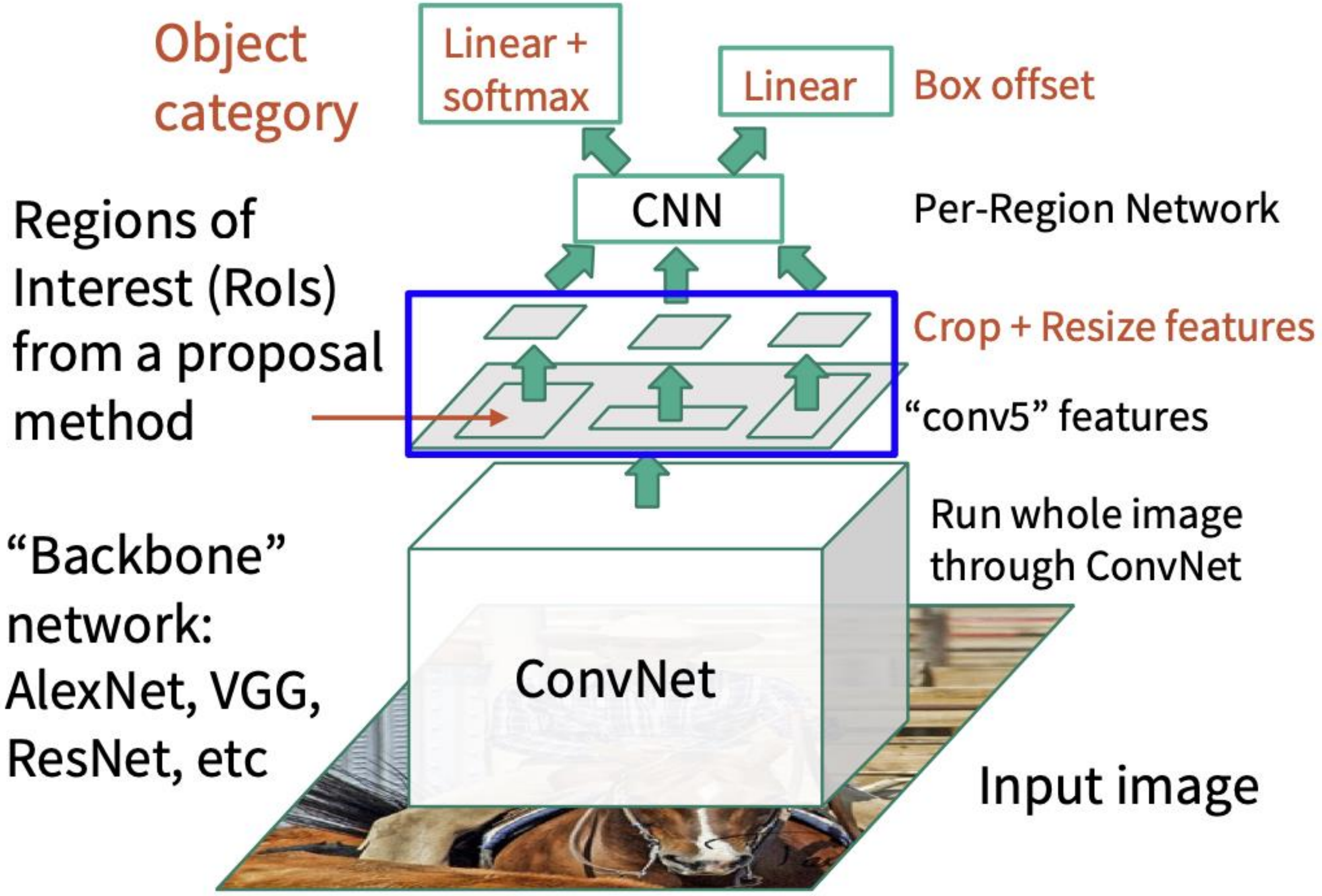
Problem: Very slow! Need to do ~2k independent forward passes for each image!



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

R-CNN and Fast R-CNN

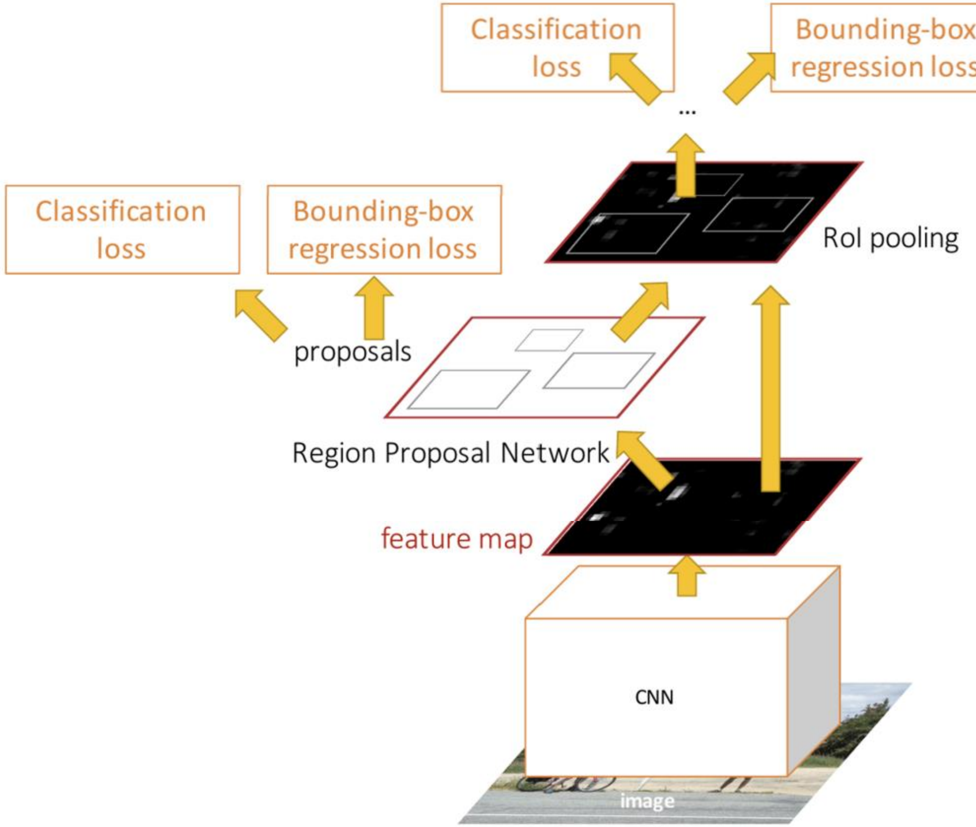




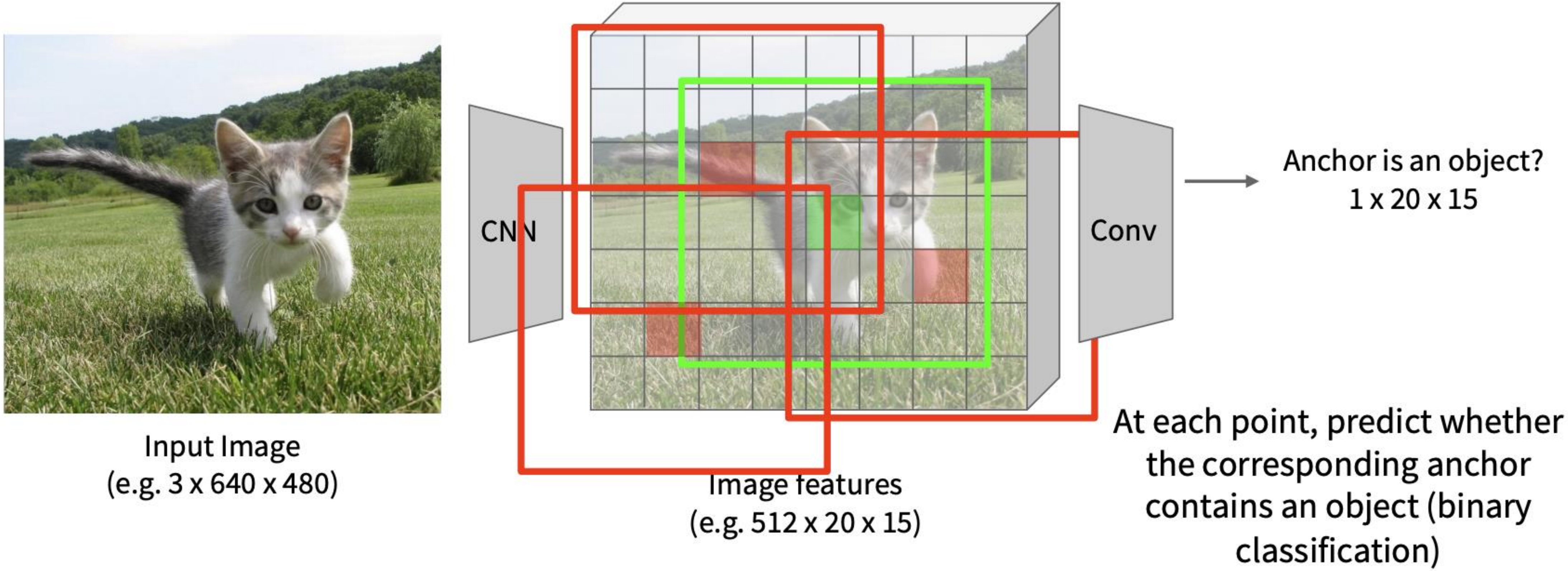
Extract around 2000 bottom-up region proposals from a proposal method

Faster R-CNN: Make CNN Do Proposals

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

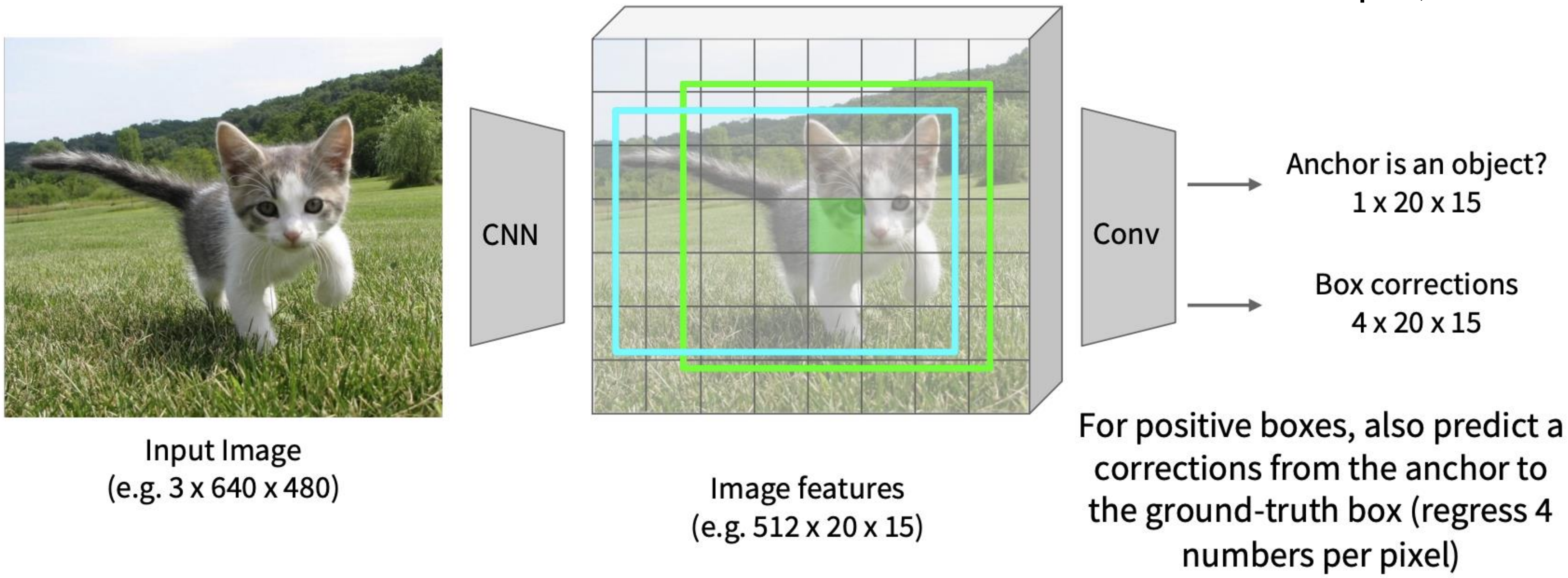


Region Proposal Network (1)



Region Proposal Network (2)

In practice use K different anchor boxes of different size / scale at each point. In this example, K is 1.



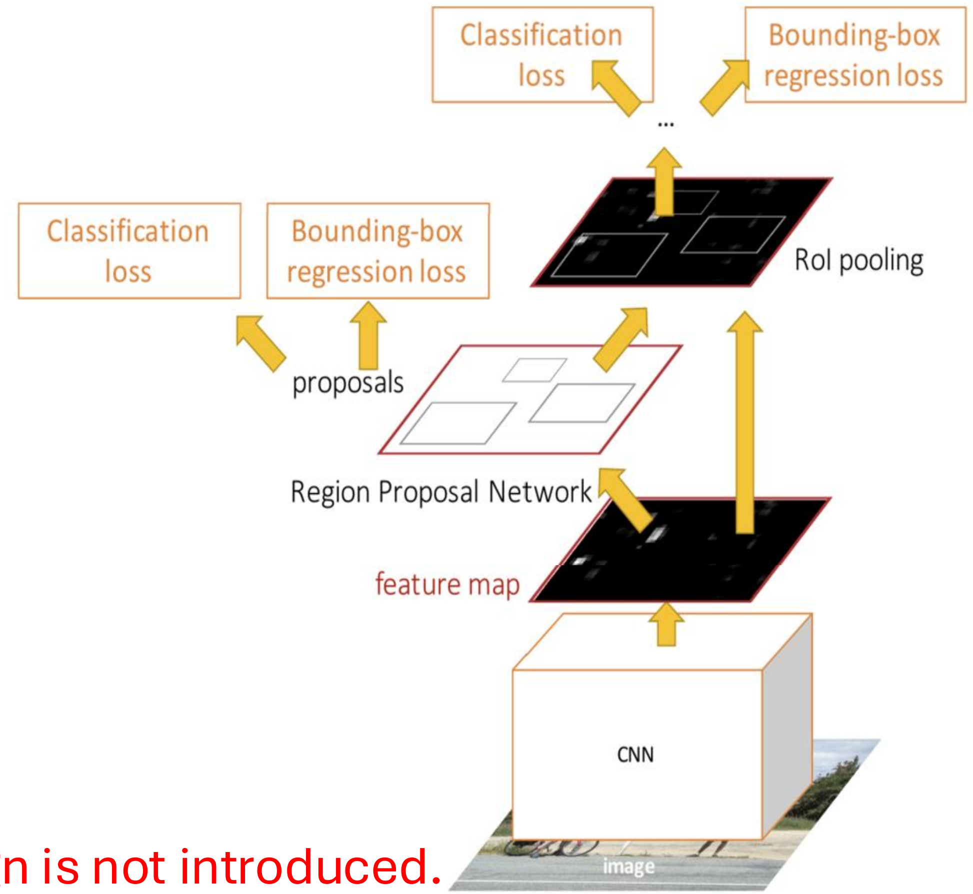
Faster R-CNN: Two Stages

Jointly train with 4 losses:

- RPN classify object / not object
- RPN regress box coordinates
- Final classification score (object classes)
- Final box coordinates

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

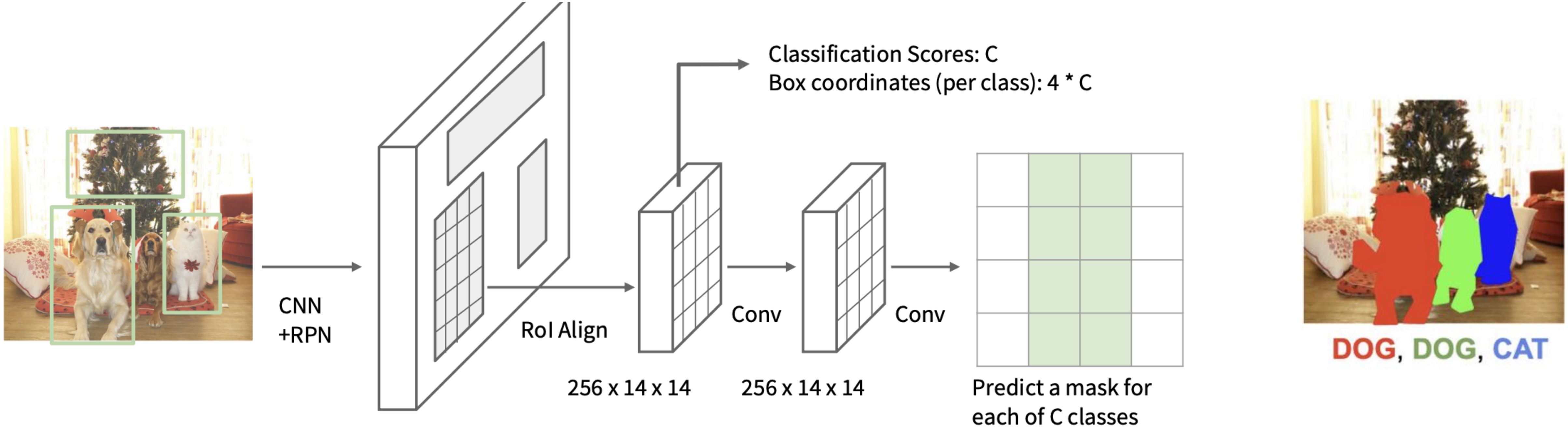
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Note: RoI pool/align is not introduced.

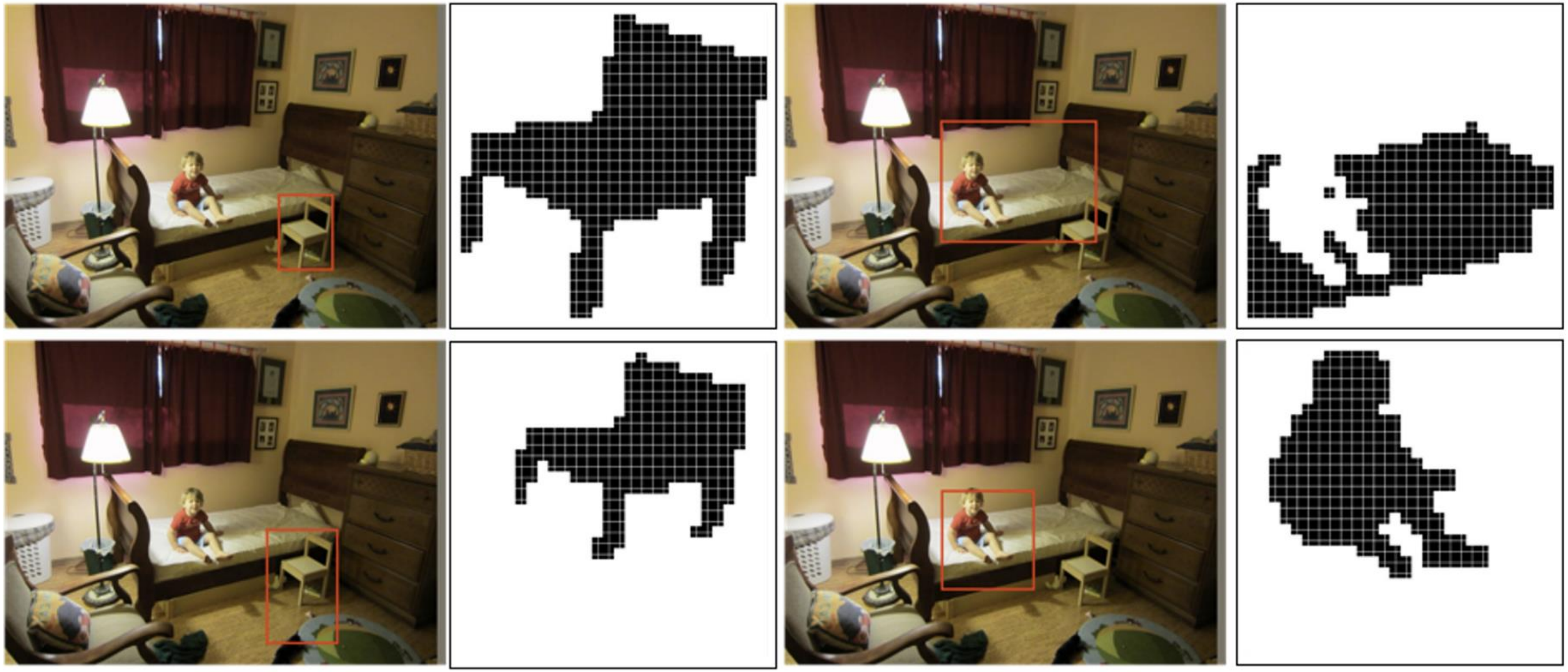
Extracting fixed-size feature maps from a variable-sized region of interest (RoI) in a feature map

Instance Segmentation: Mask R-CNN



Mask R-CNN: Learn by yourself

Mask R-CNN: Example Mask Training Targets



Instance Segmentation in GPT5

ChatGPT 5

Analysis errored

python

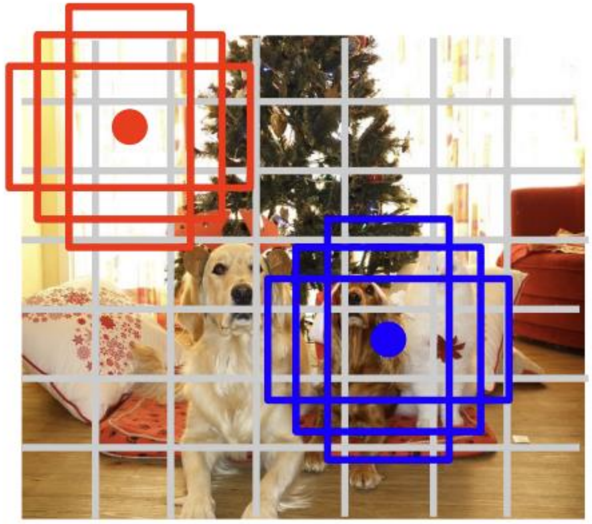
Always show details

Copy code

```python
import torch
import torchvision
from torchvision.transforms import functional as F

# Load a pre-trained Mask R-CNN model for instance segmentation
model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Transform the image into tensor format
image_tensor = F.to_tensor(image).unsqueeze(0)

# Perform instance segmentation
with torch.no_grad():
    predictions = model(image_tensor)
```

Yolo: Single Stage Object Detector

Input image
3 x H x W



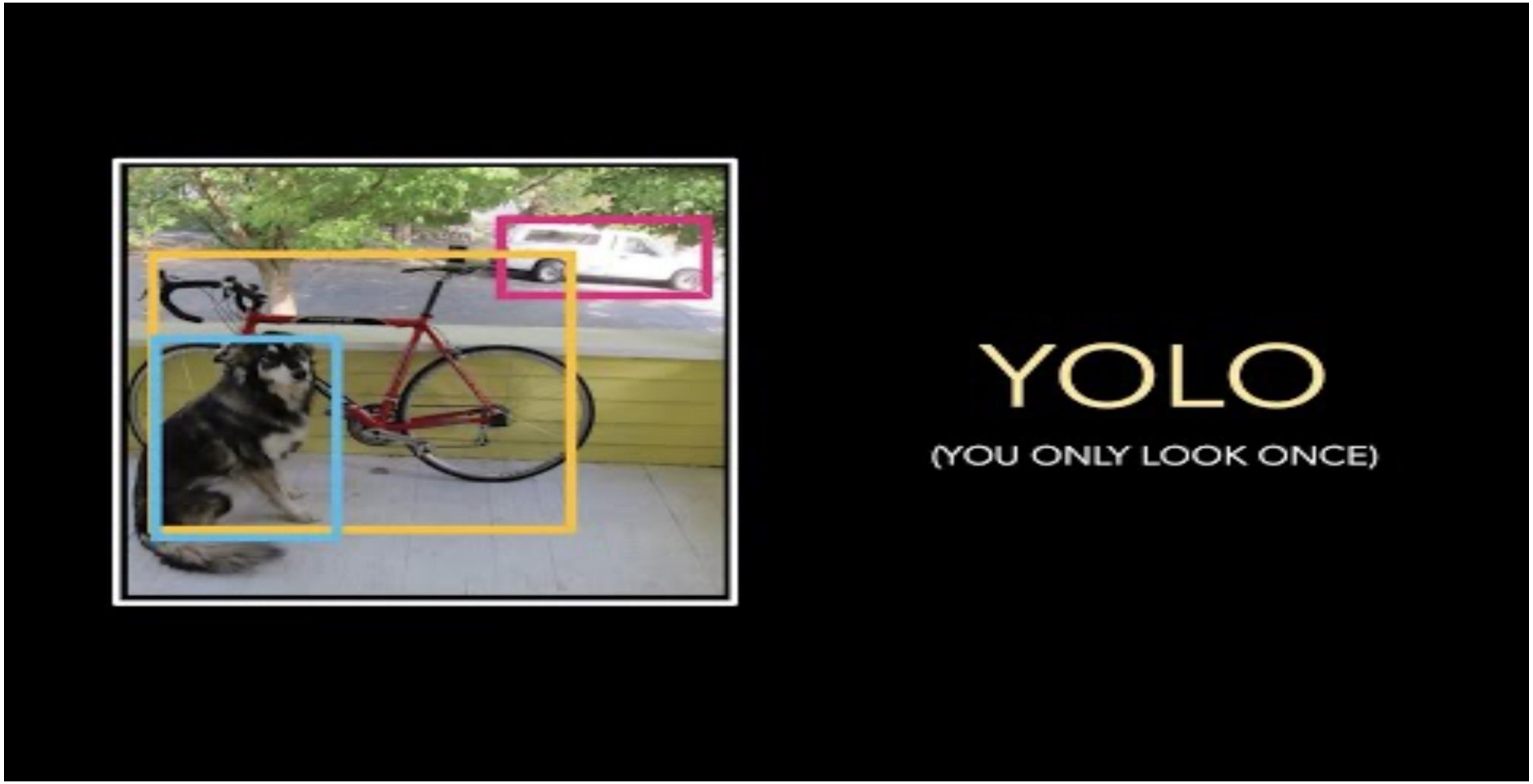
Divide image into grid
7 x 7

Image a set of base boxes centered at each grid cell Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!
- Output: 7x7x(5*B+C)

YOLO: Model as a Regression Problem



https://youtu.be/svn9-xV7wjk

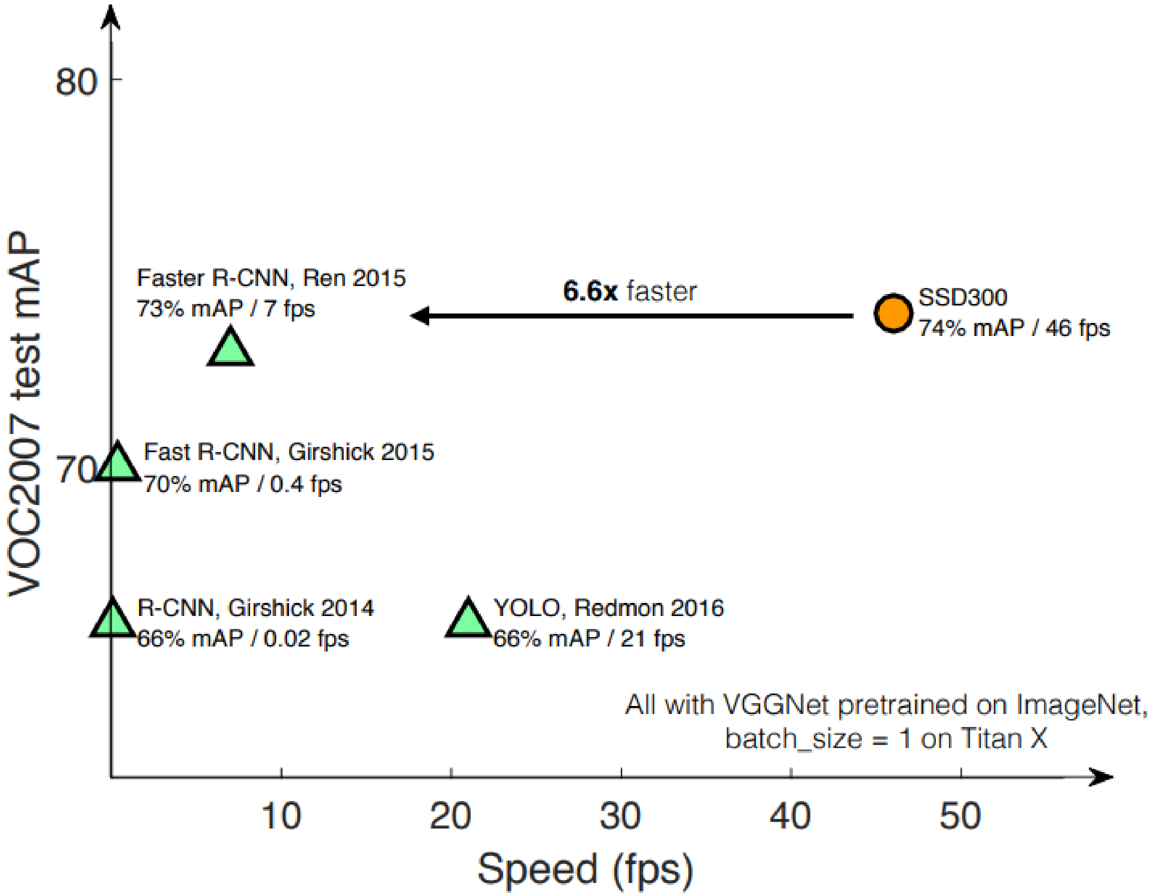
Object Detection: Evaluation Metrics

- Intersection over Union (IoU)
  - Predicted bounding box (A) and ground truth bounding box (B)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

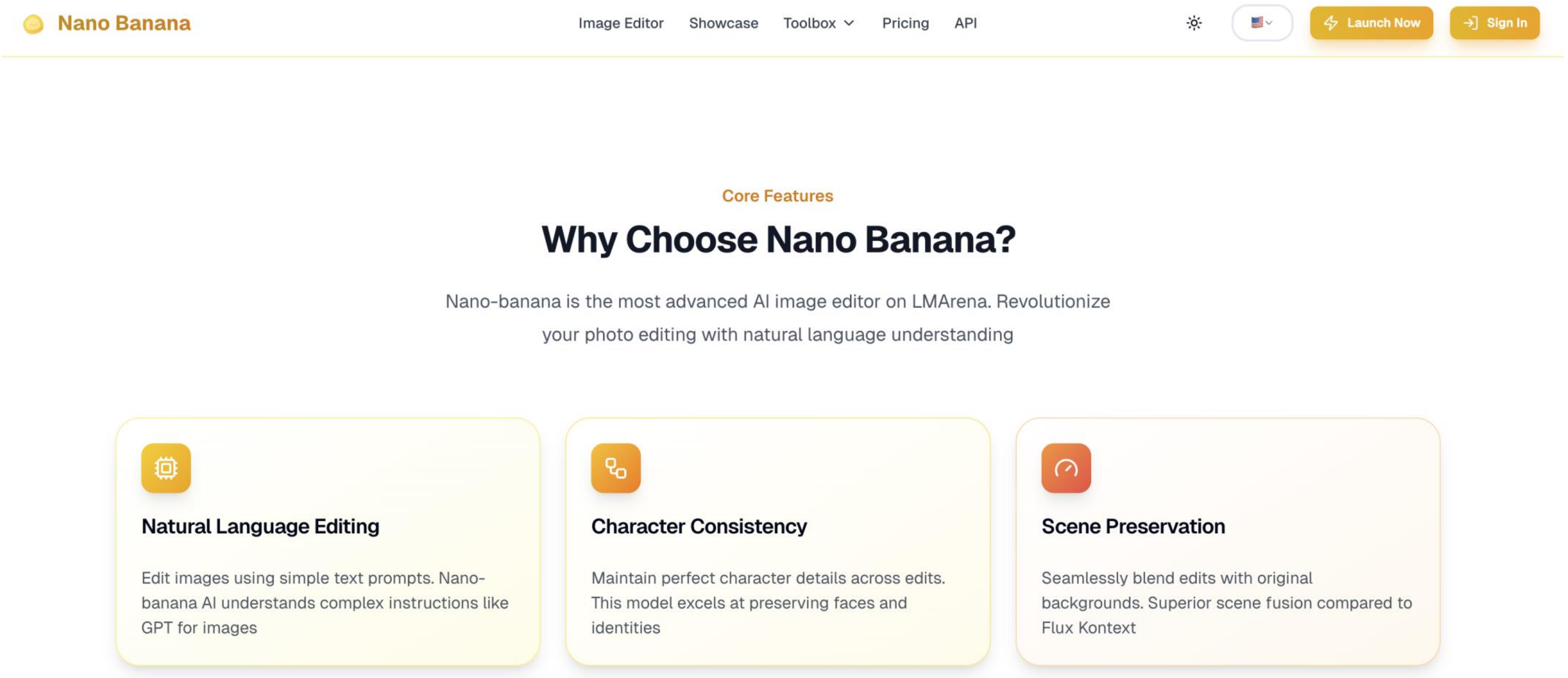
- Average Precision (AP)
  - The precision-recall curve that is created by varying the detection threshold.
  - mean Average Precision (mAP), which calculates AP for each class and then take the average

Single-shot VS Two-shot Detector



https://www.cs.unc.edu/~wliu/papers/ssd_eccv2016_slide.pdf

Try Nano Banana or Midjourney



References

- https://cs231n.stanford.edu/slides/2024/lecture_9.pdf
- https://encord.com/blog/yolo-object-detection-guide/
- https://github.com/ultralytics/ultralytics
- https://github.com/facebookresearch/detectron2