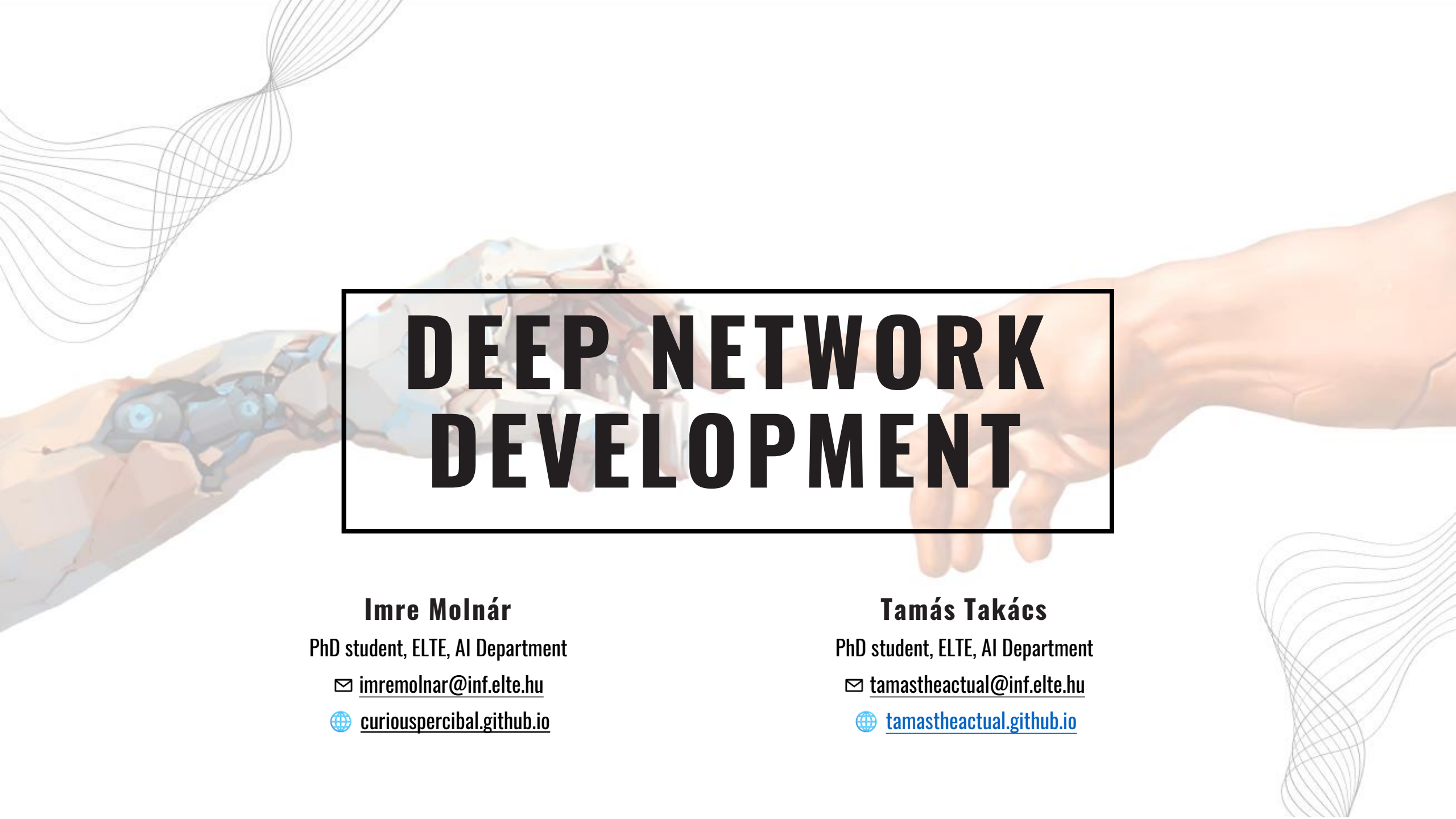


DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

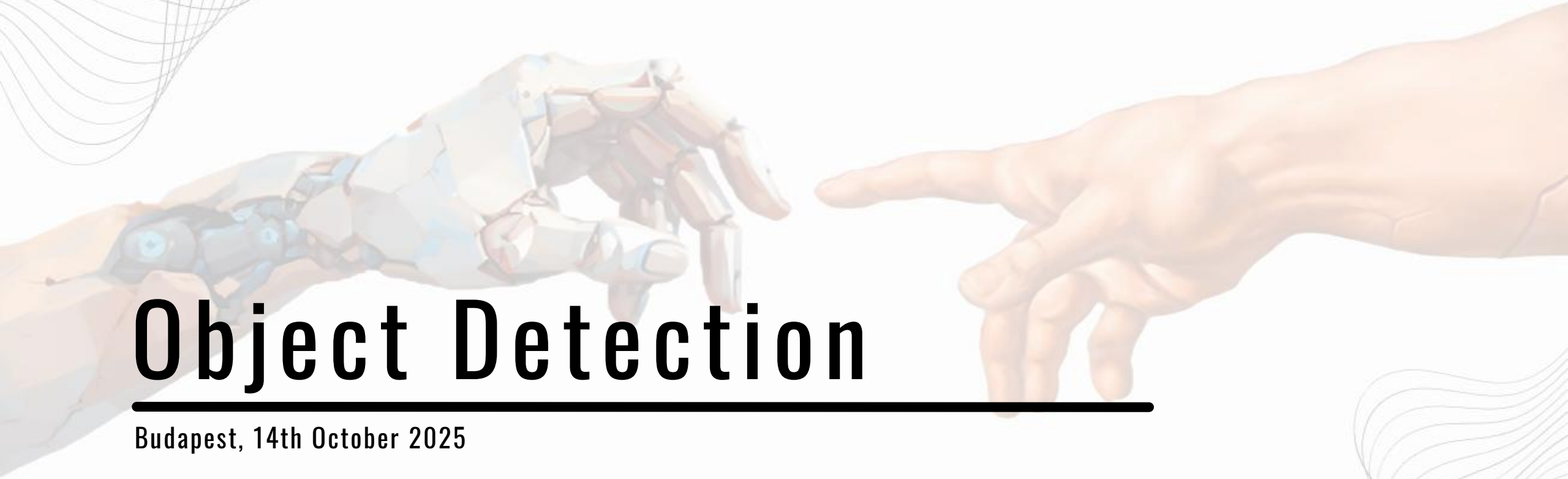
PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

Lecture 8.



Object Detection

Budapest, 14th October 2025

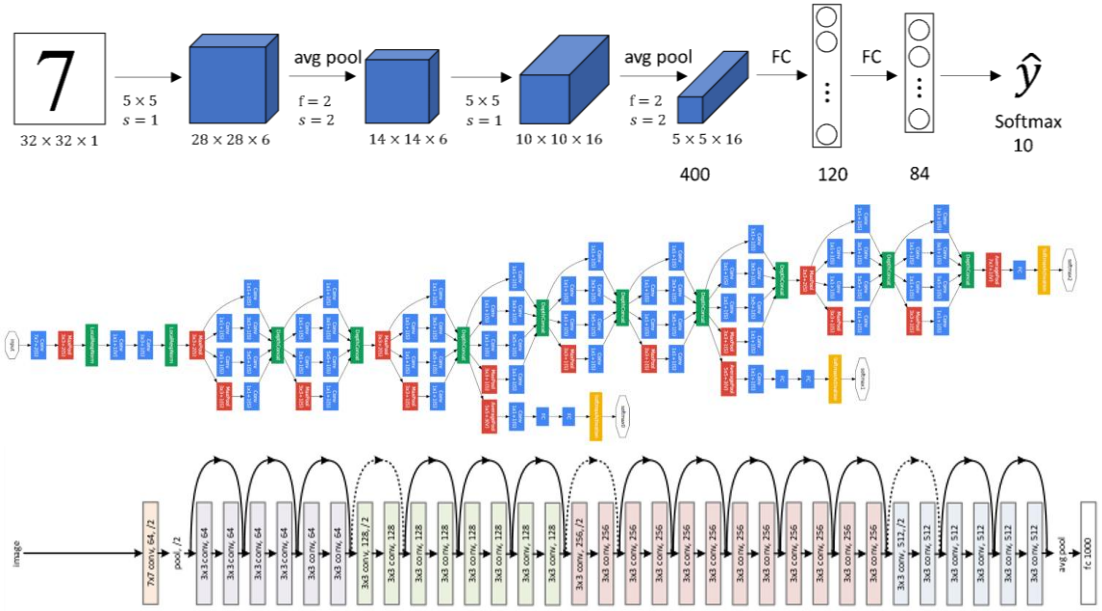
1 Two Stage Detectors

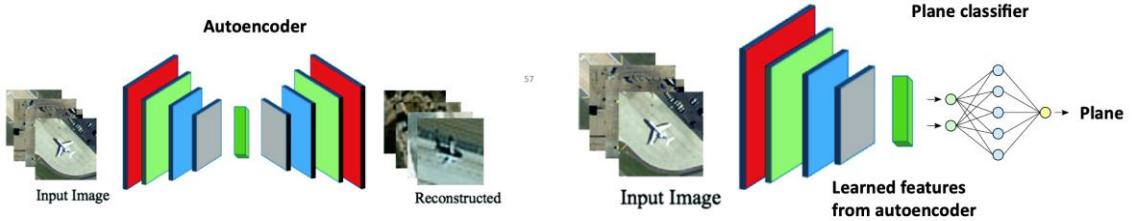
2 One Stage Detectors

3 Object Detection Metrics

Previously on Lecture 4

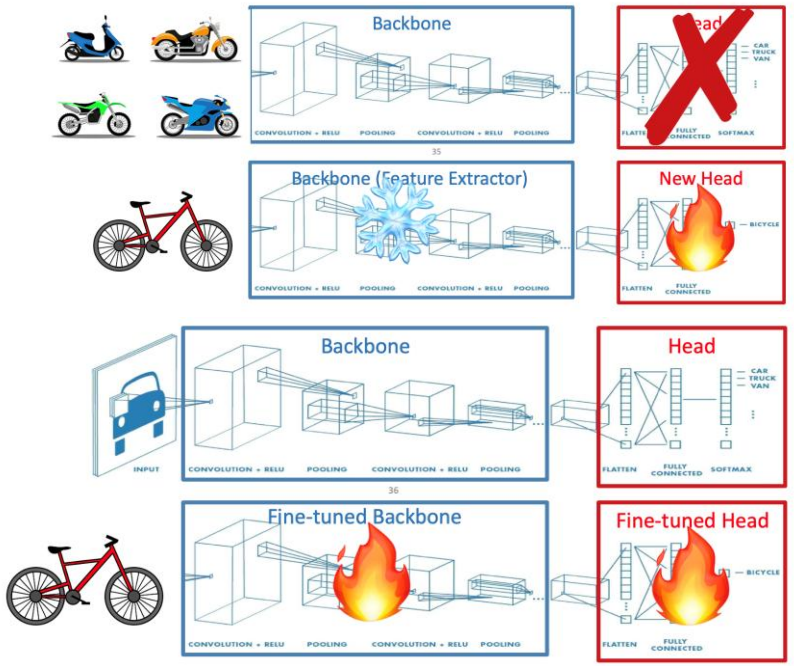
CNN architectures



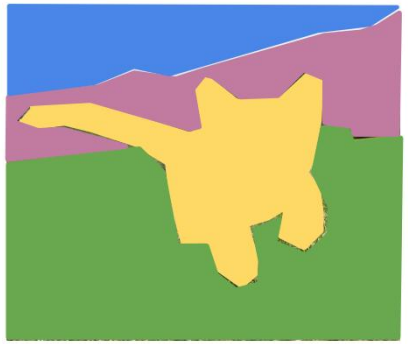
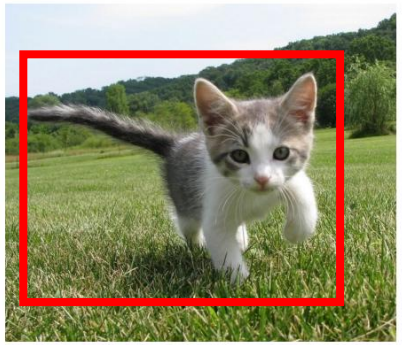
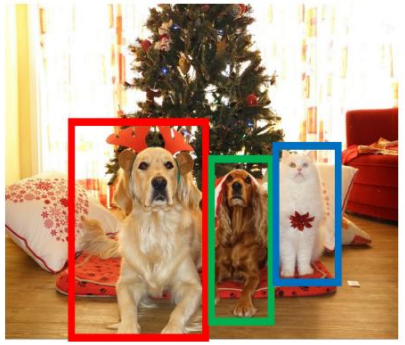


When to use Transfer Learning

- Task A and B have the same input x
- When you have a lot of data for the problem you are transferring from (A) and few data for the problem you are transferring to (B)
- Low level features from A could be helpful for learning B
- Faster training. Use pre-trained weights as initialization point whether than randomly initializing weights



Supervised Learning tasks

Classification	Semantic Segmentation	Classification + Localization	Object Detection	Instance Segmentation
				
CAT	GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT
Single Object	No objects, just pixels	Single Object	Multiple Objects	Multiple Objects
✓		⟳	⟳	

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

What is Object Detection?

Classification
[cat, dog, car]

[0.9, 0.05, 0.05]

[0.7, 0.21, 0.09]

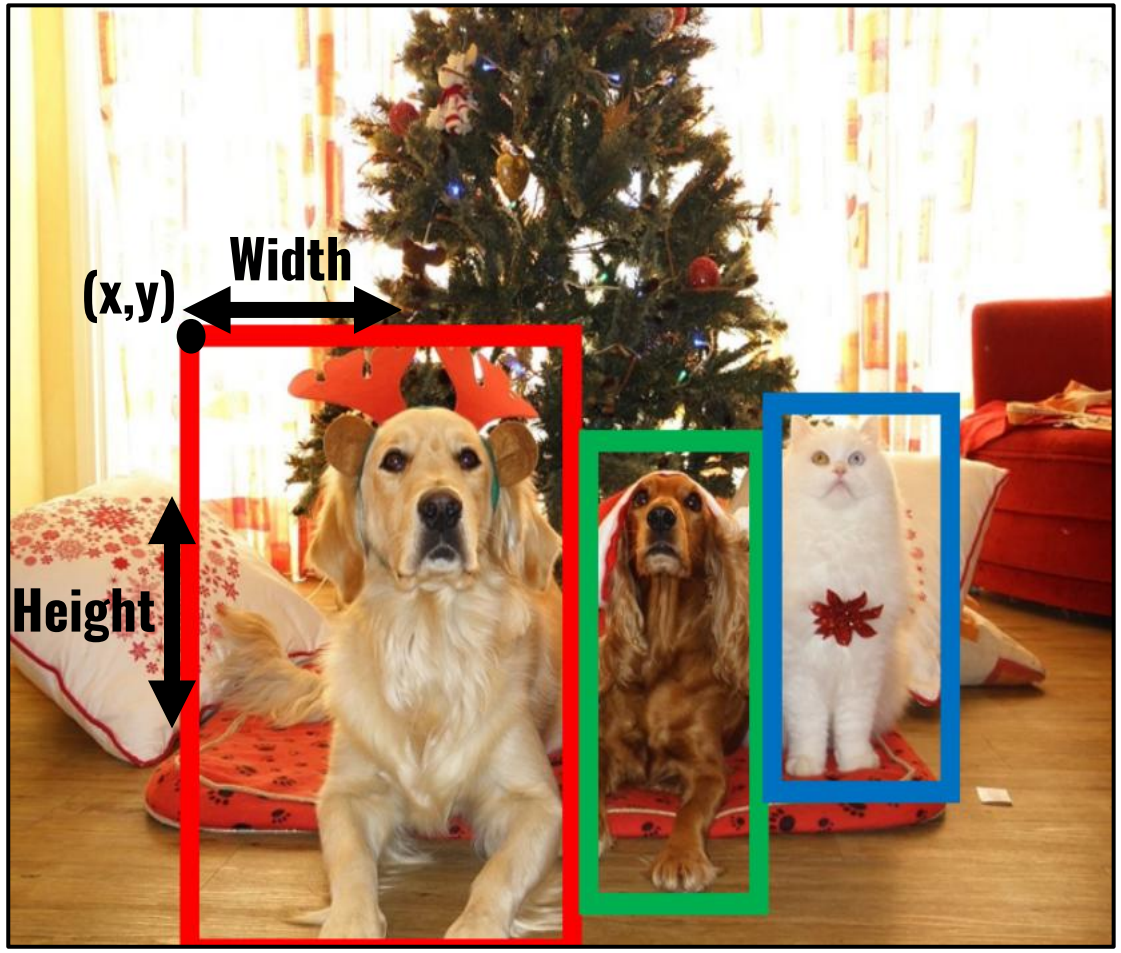
WHERE?

[0.48, 0.51, 0.01]

What is Object Detection?

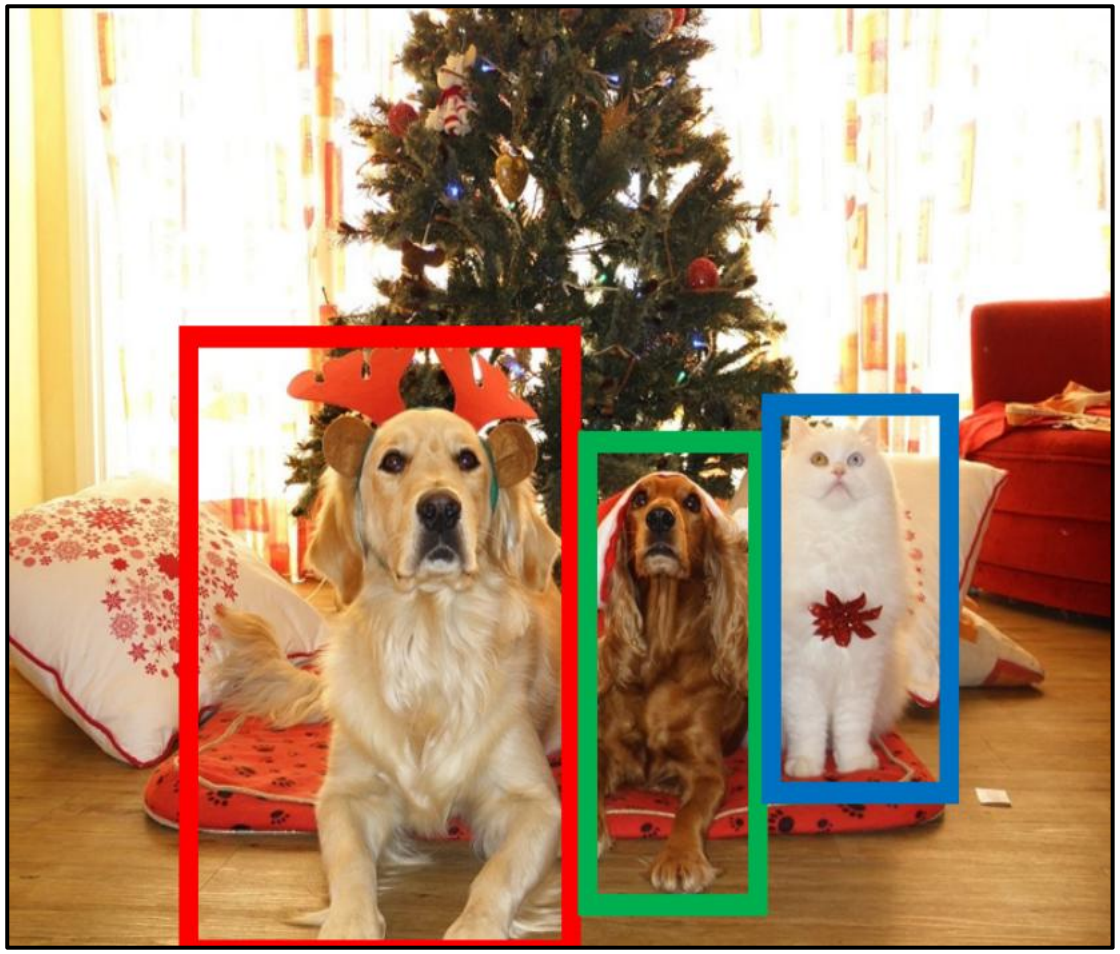
- **Supervised Learning Task**
- **Input**: RGB image
- **Output**: A set of detected objects;

For each object predict:
- **Category label** (from fixed, known set of categories)
- **Bounding box** (four numbers: x, y, width, height)

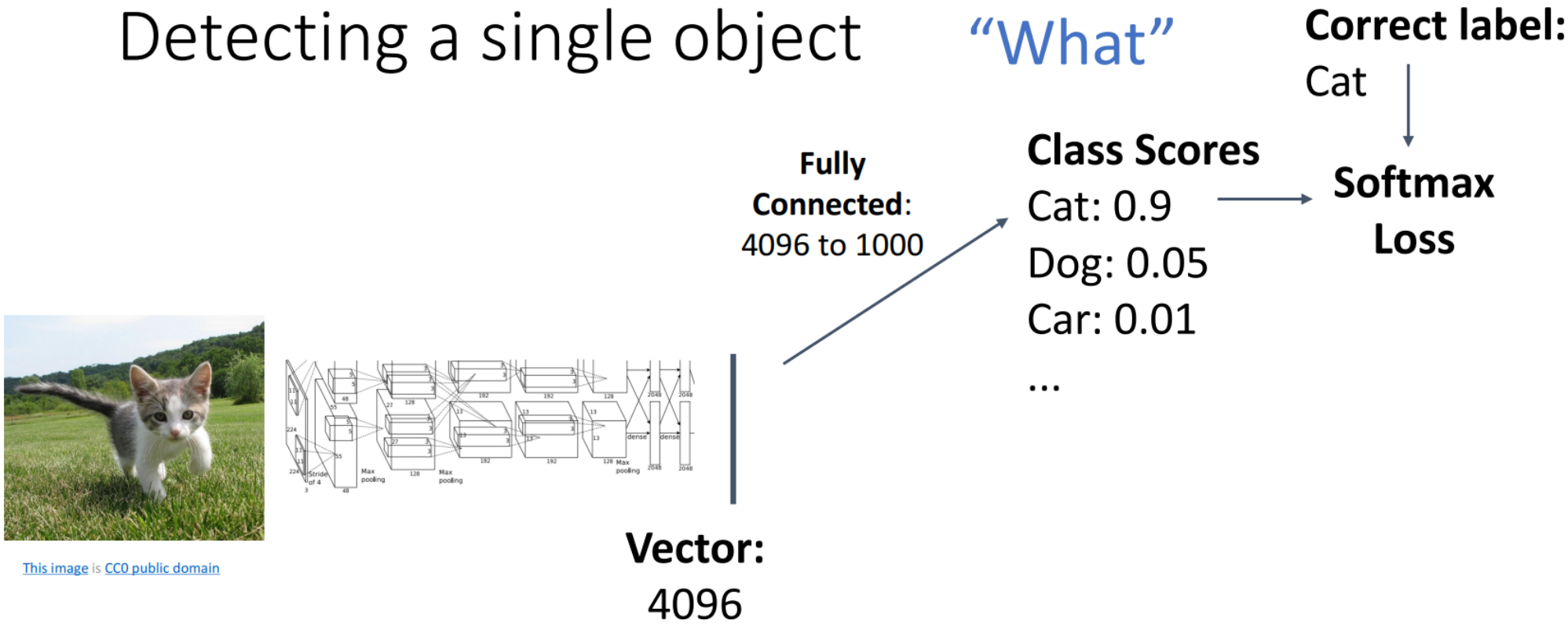


What is Object Detection?

- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of outputs**: 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

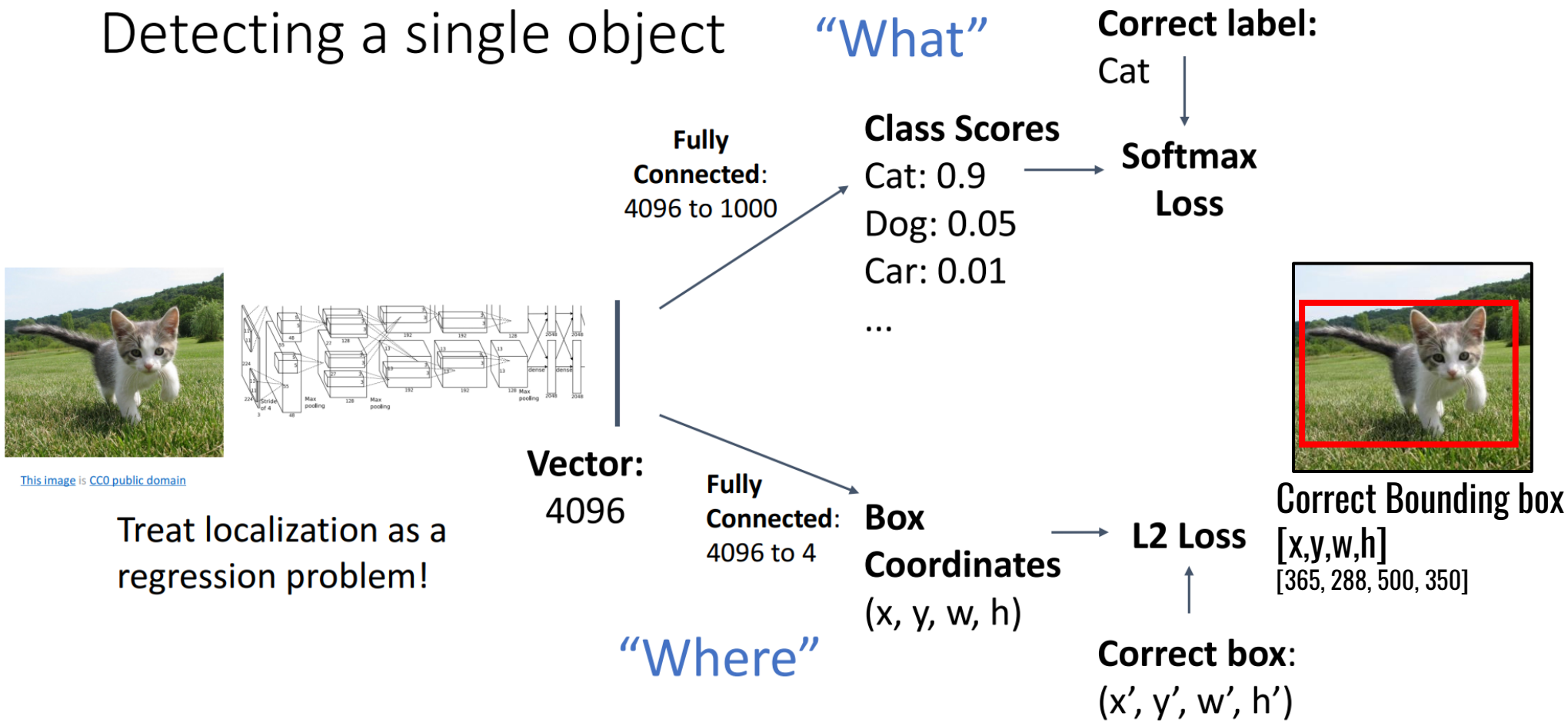


Classification + Localization (Object Detection for a single object)



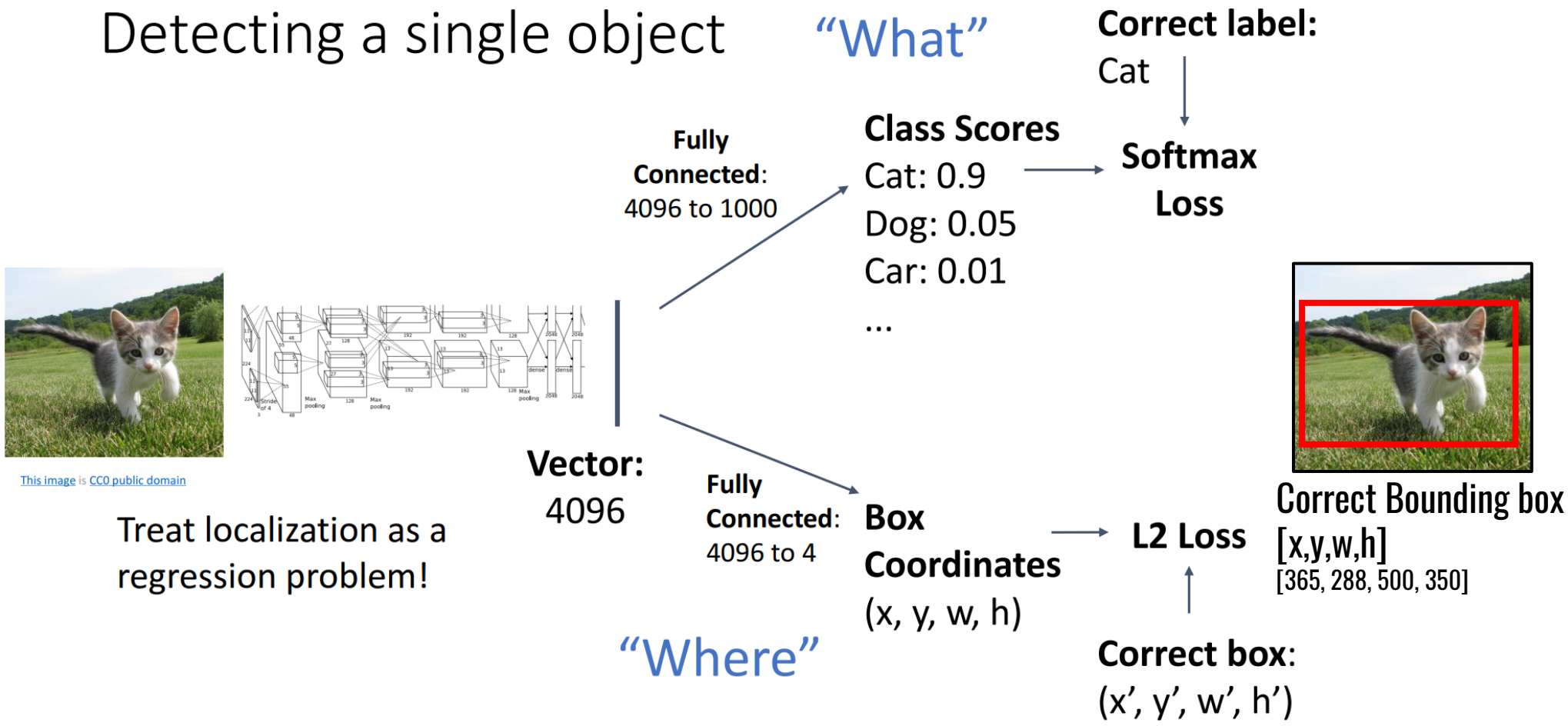
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Classification + Localization (Object Detection for a single object)



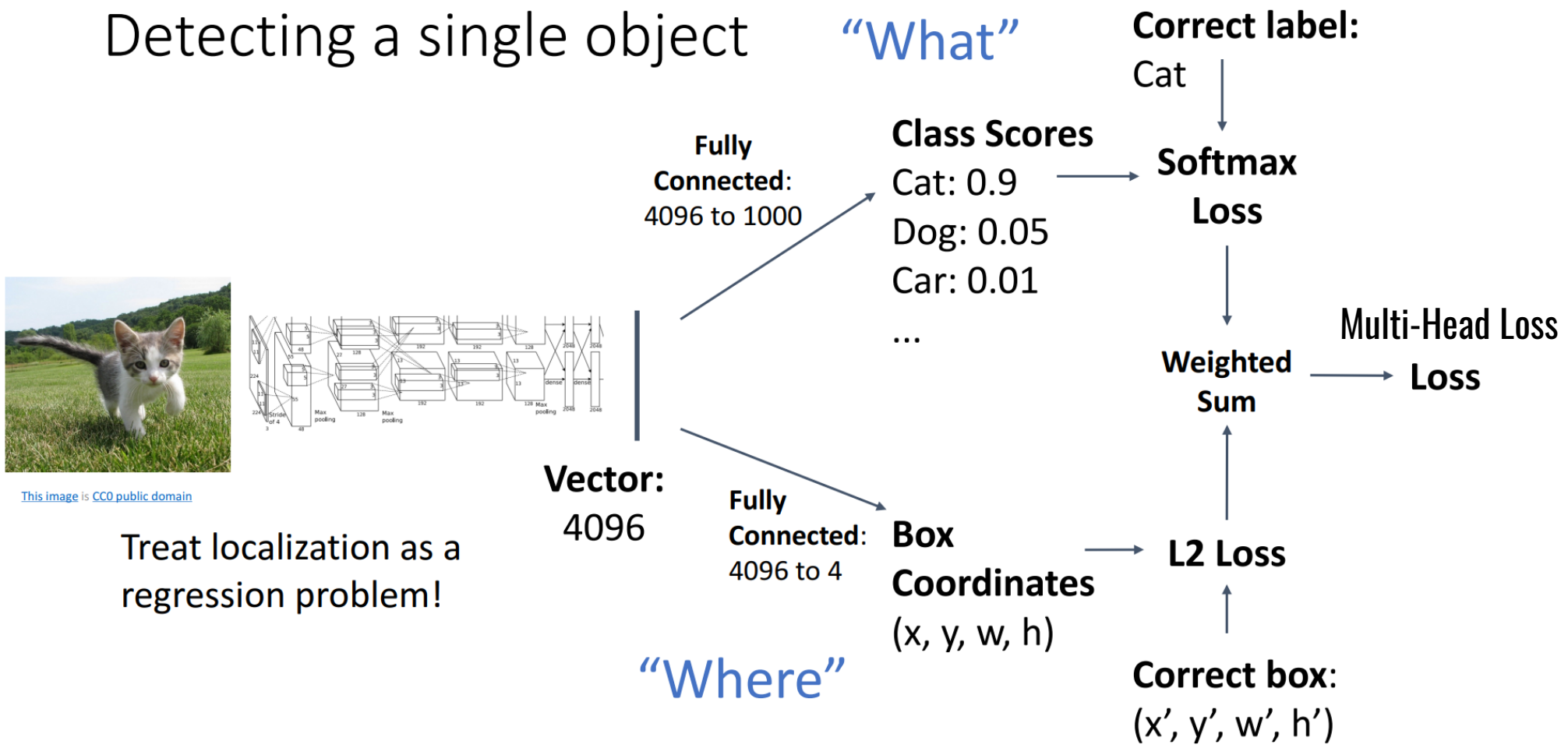
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Classification + Localization (Object Detection for a single object)



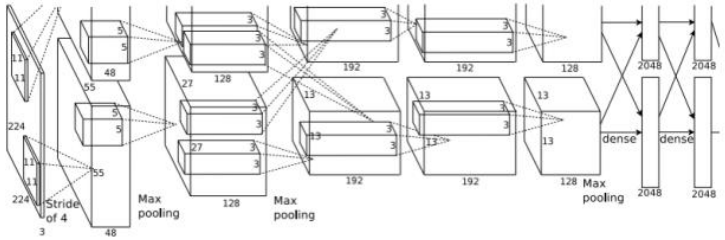
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Classification + Localization (Object Detection for a single object)



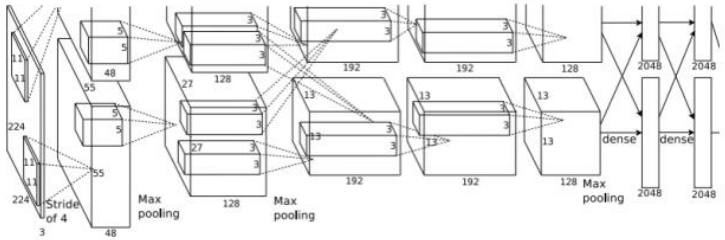
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Classification + Localization (Object Detection for a single object)



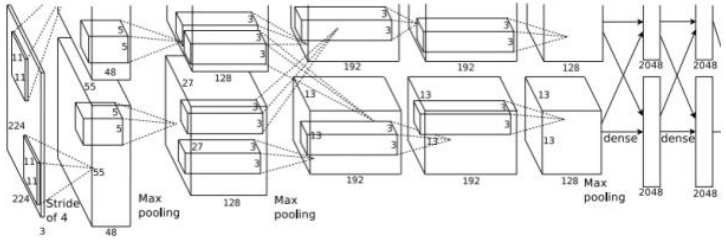
CAT: (x, y, w, h)

4 numbers



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

16 numbers



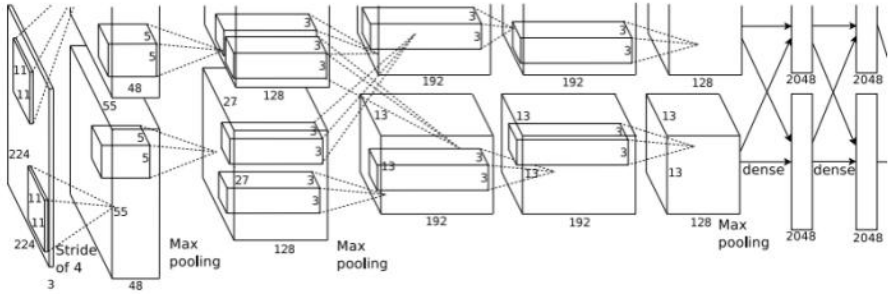
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Sliding Window (Naïve approach)

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

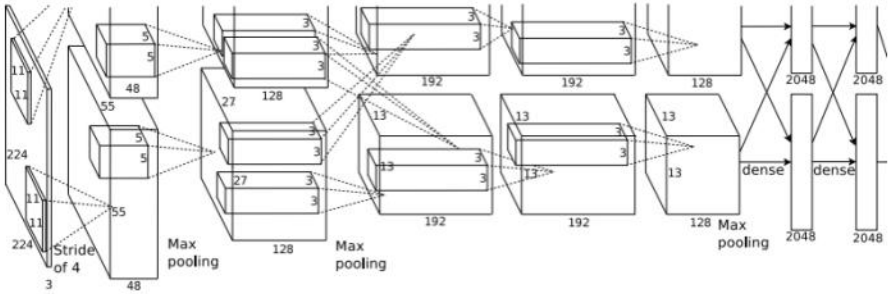


Dog? NO
Cat? NO
Background? YES

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Sliding Window (Naïve approach)

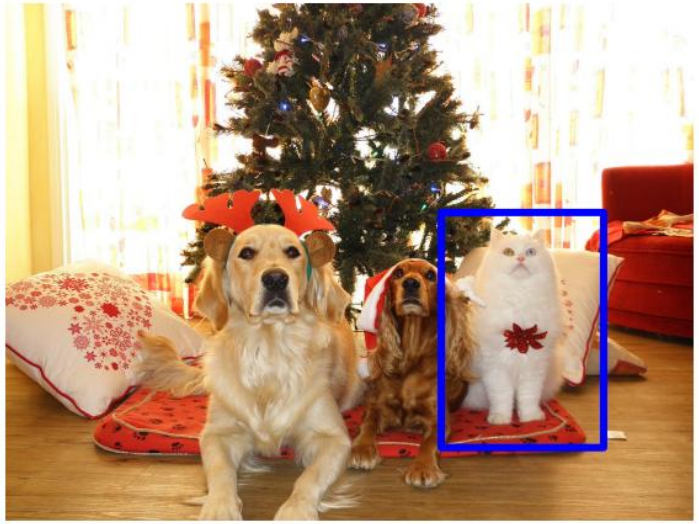
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Sliding Window (Naïve approach)



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:

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

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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

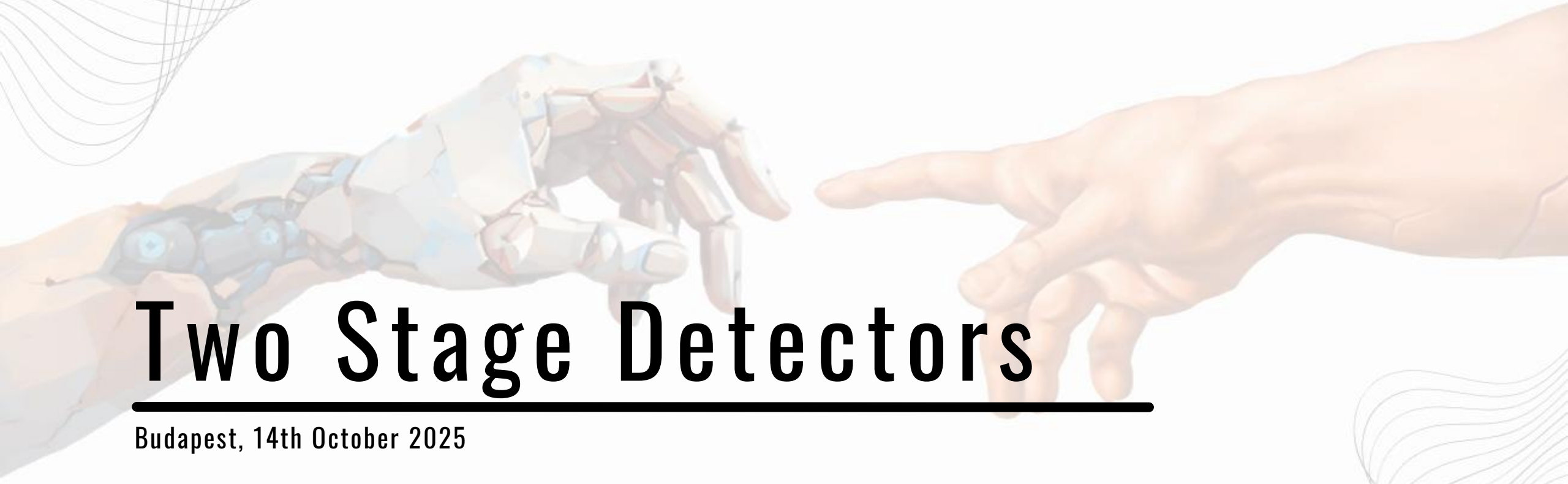
Object Detection for multiple objects

Region proposal based (Two Stage Detectors)

- R-CNN
- Fast R-CNN
- Faster R-CNN

Deep Network Development

Lecture 8.



Two Stage Detectors

Budapest, 14th October 2025

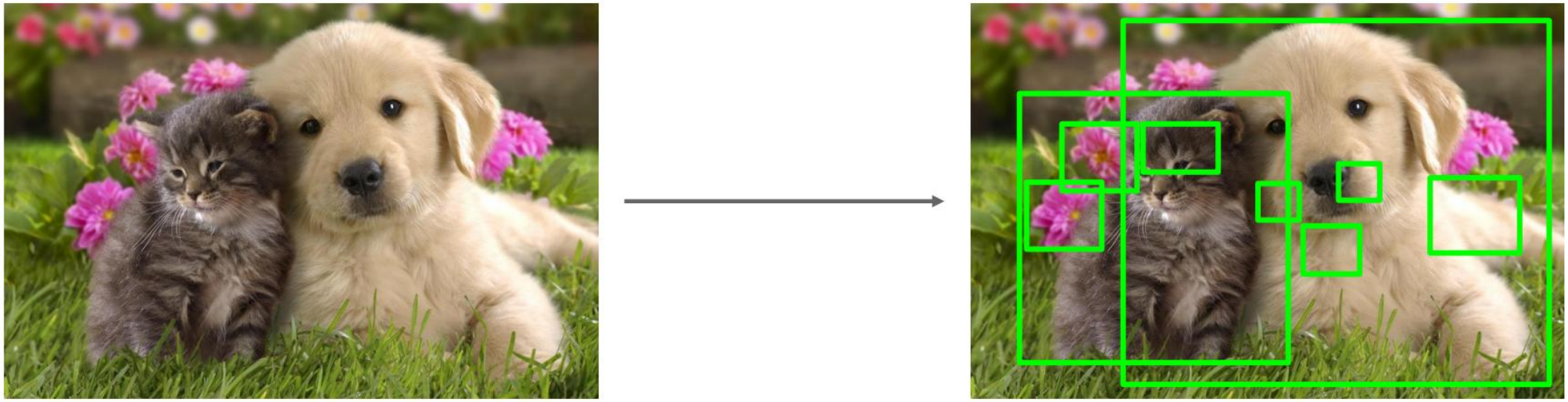
1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

Region Proposal Detectors (Two Stage Detectors)

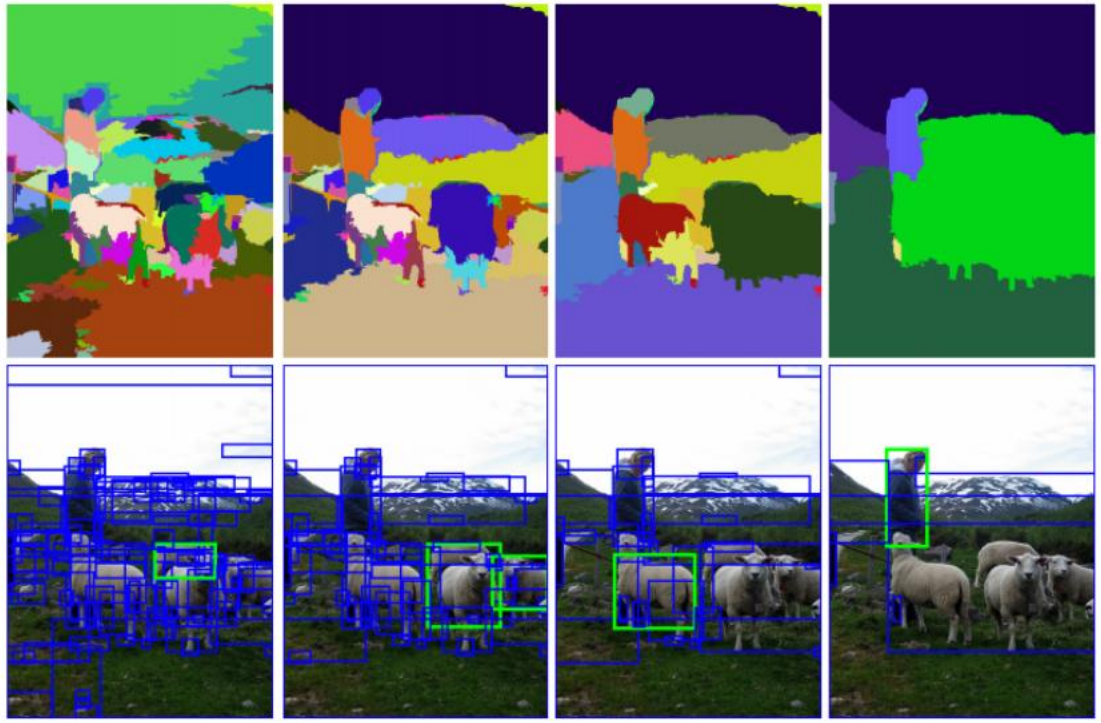
- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

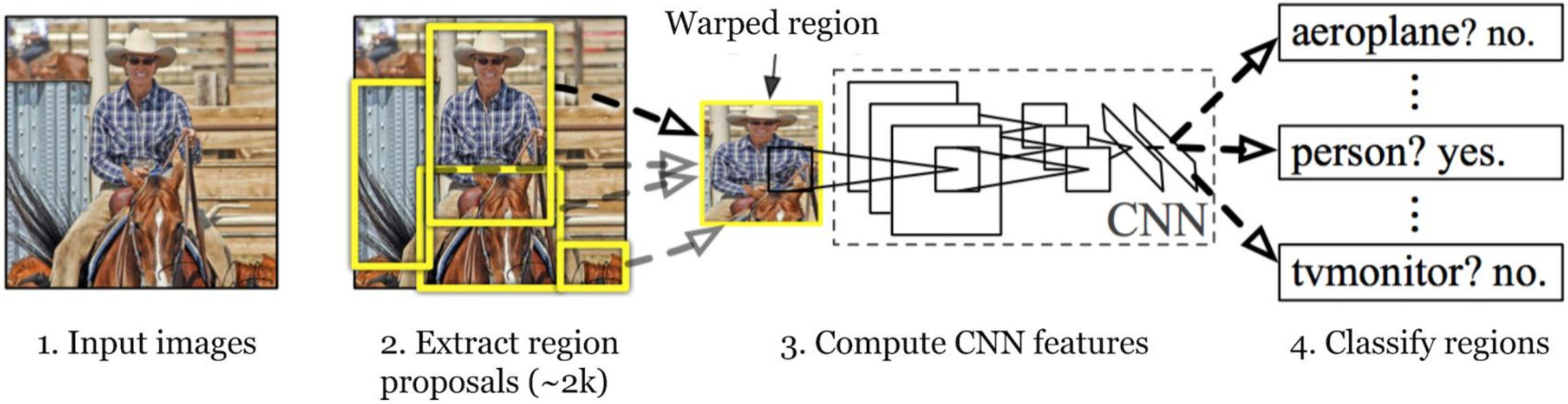
Region Proposal Detectors (Two Stage Detectors)

- Selective Search



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network [1]

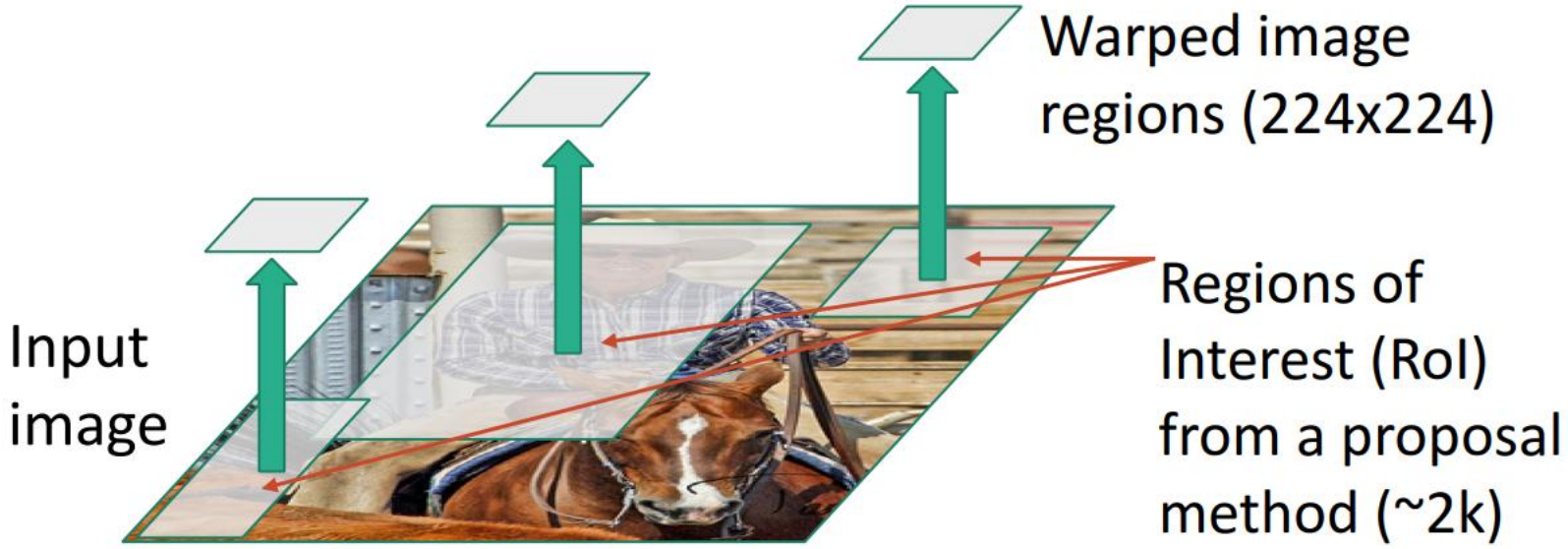


Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/
[1] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524

R-CNN: Region-Based Convolutional Neural Network

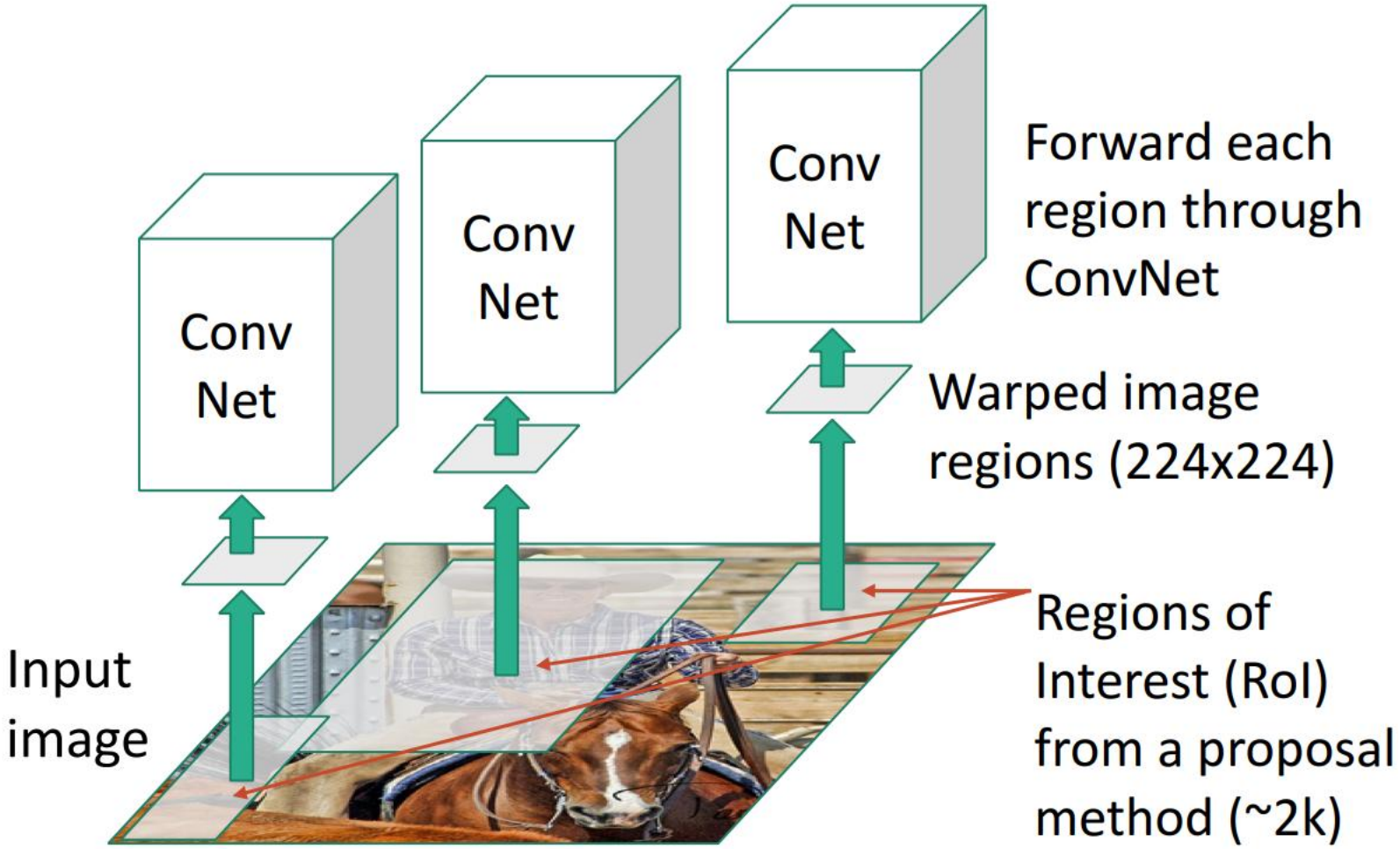
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



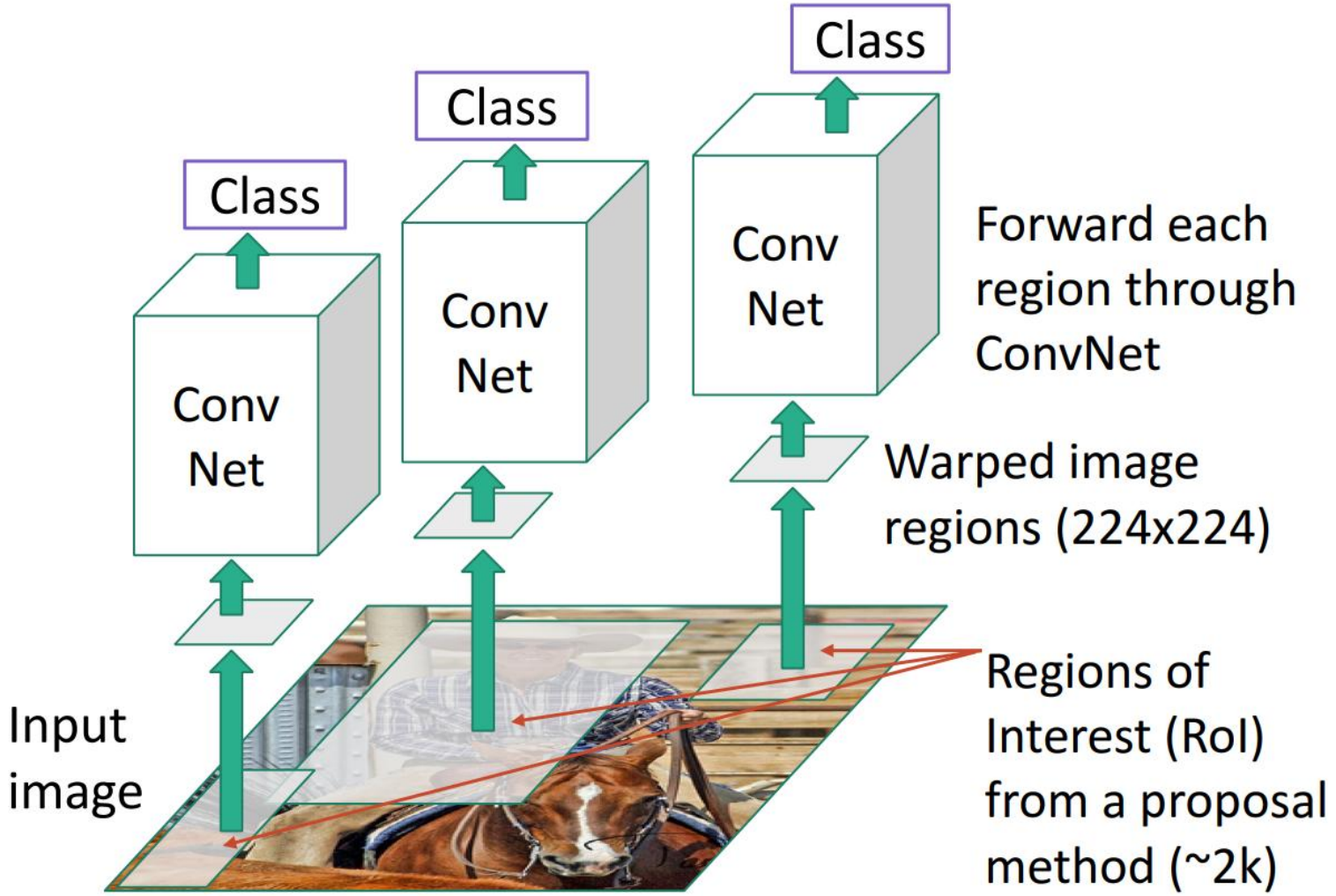
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



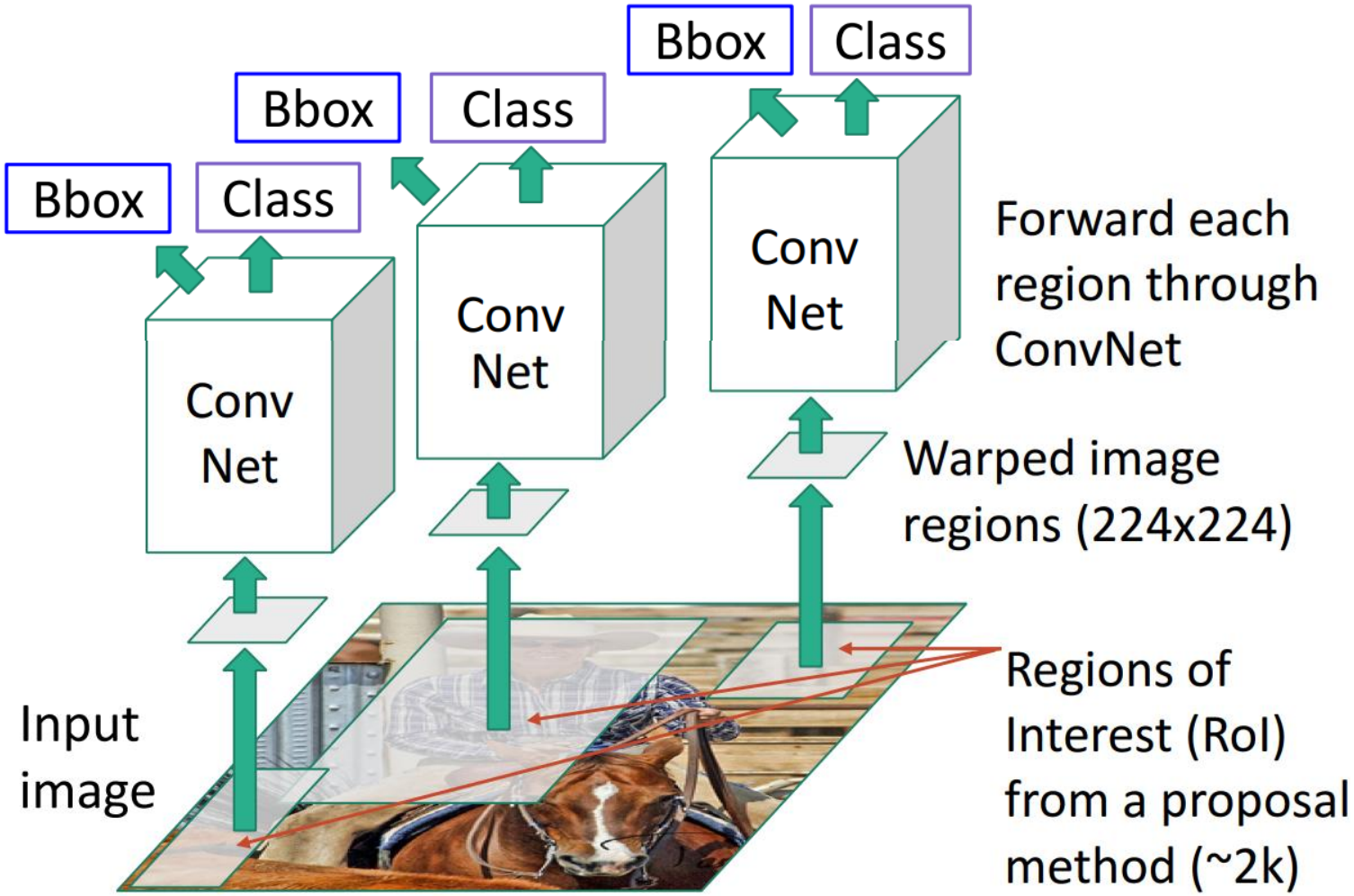
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



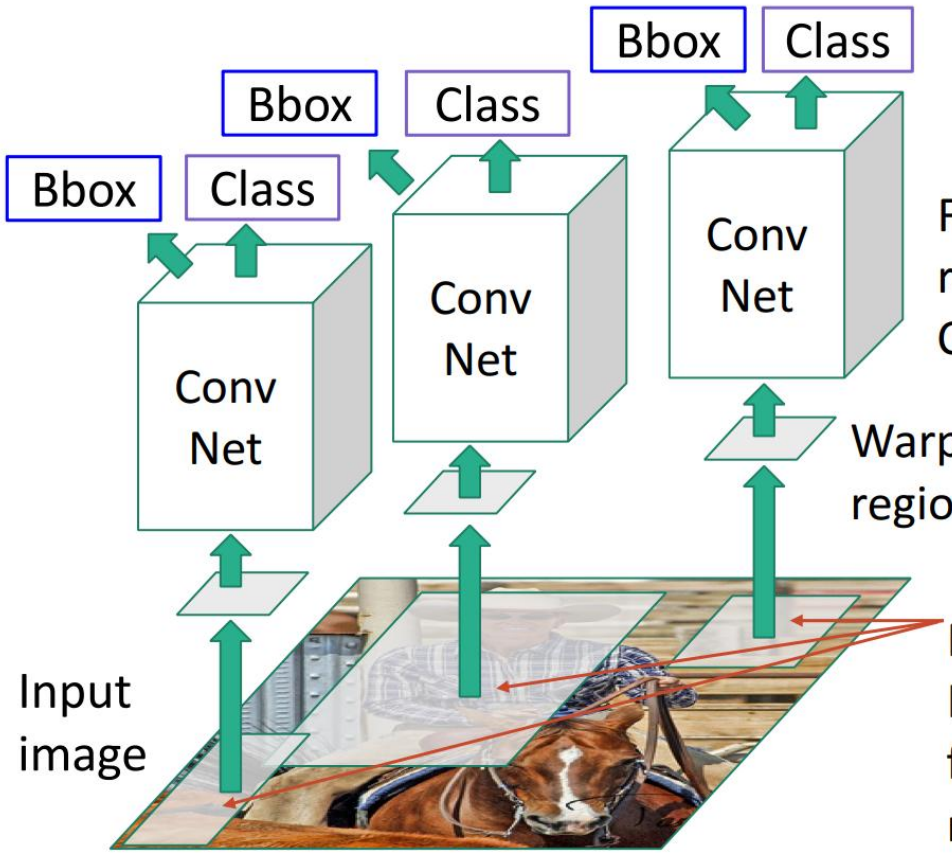
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



Forward each region through ConvNet

Classify each region

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

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

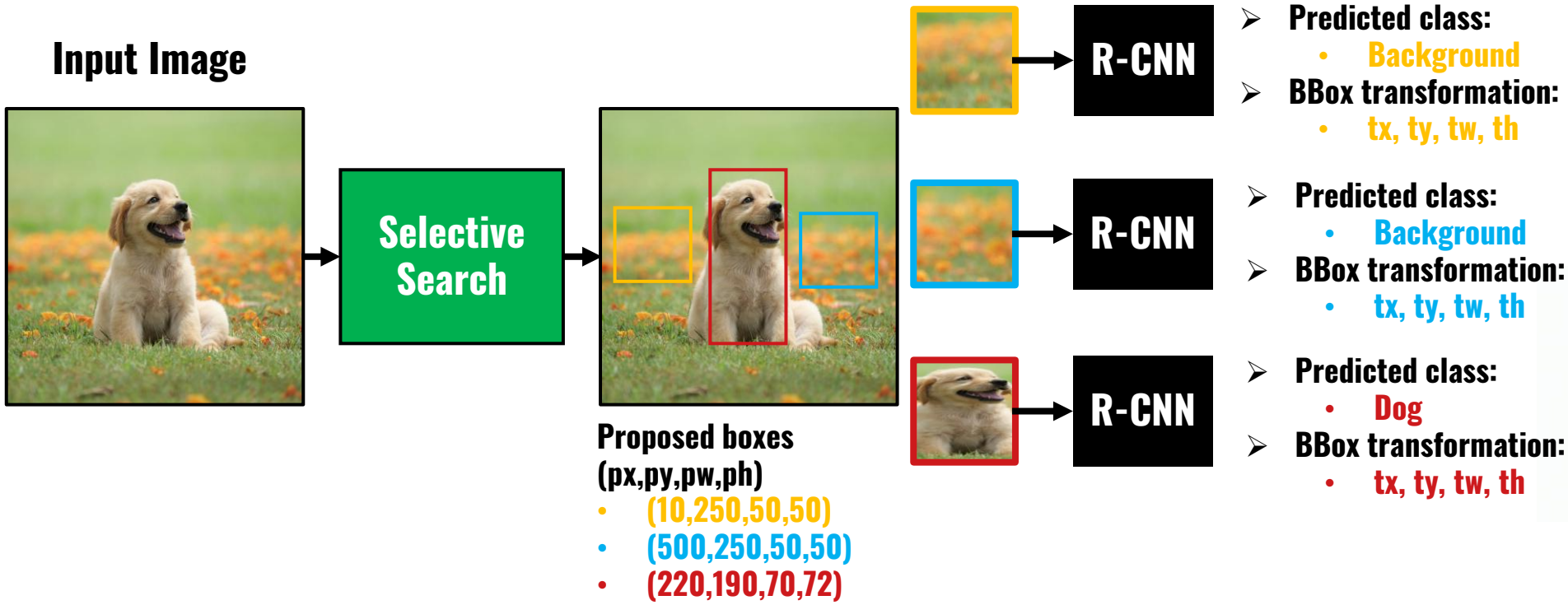
Translate relative to box size:
$b_x = p_x + p_w t_x \qquad b_y = p_y + p_h t_y$

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

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

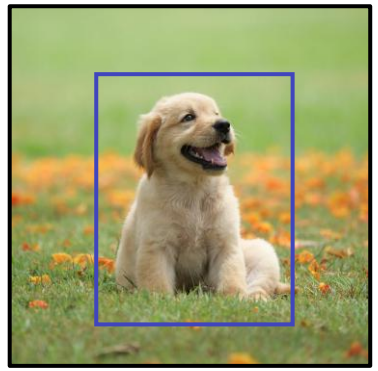
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

R-CNN: Region-Based Convolutional Neural Network



R-CNN: Region-Based Convolutional Neural Network

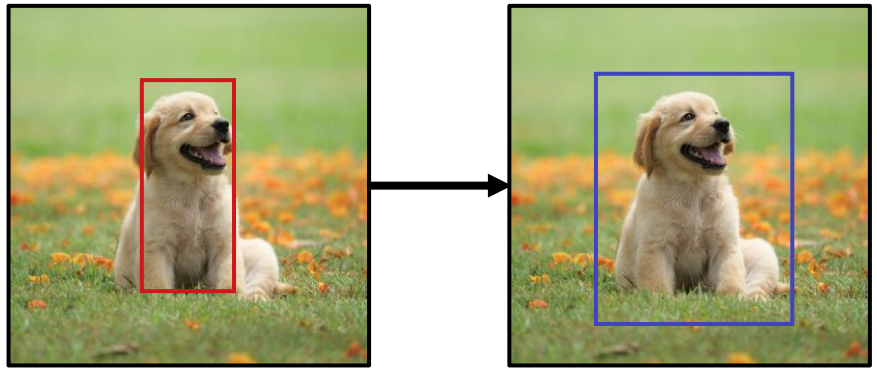
Ground Truth:
- Class: Dog
- Bounding Box (x,y,w,h) = (200,200,120,80)



How to transform the proposed box into the correct bounding box?
We use the predicted transformation values (tx,ty,tw,th)

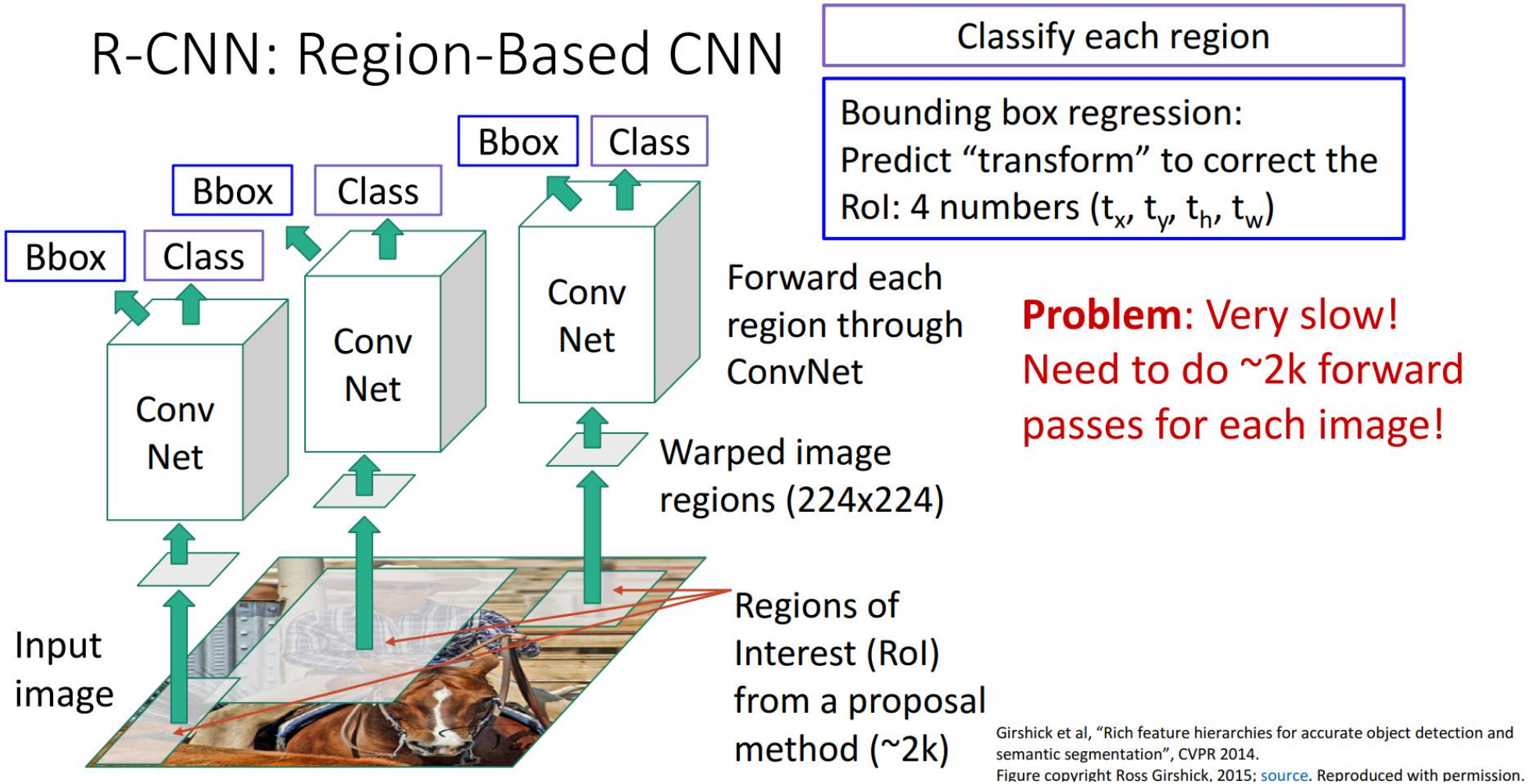
For example:

x = px + tx



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

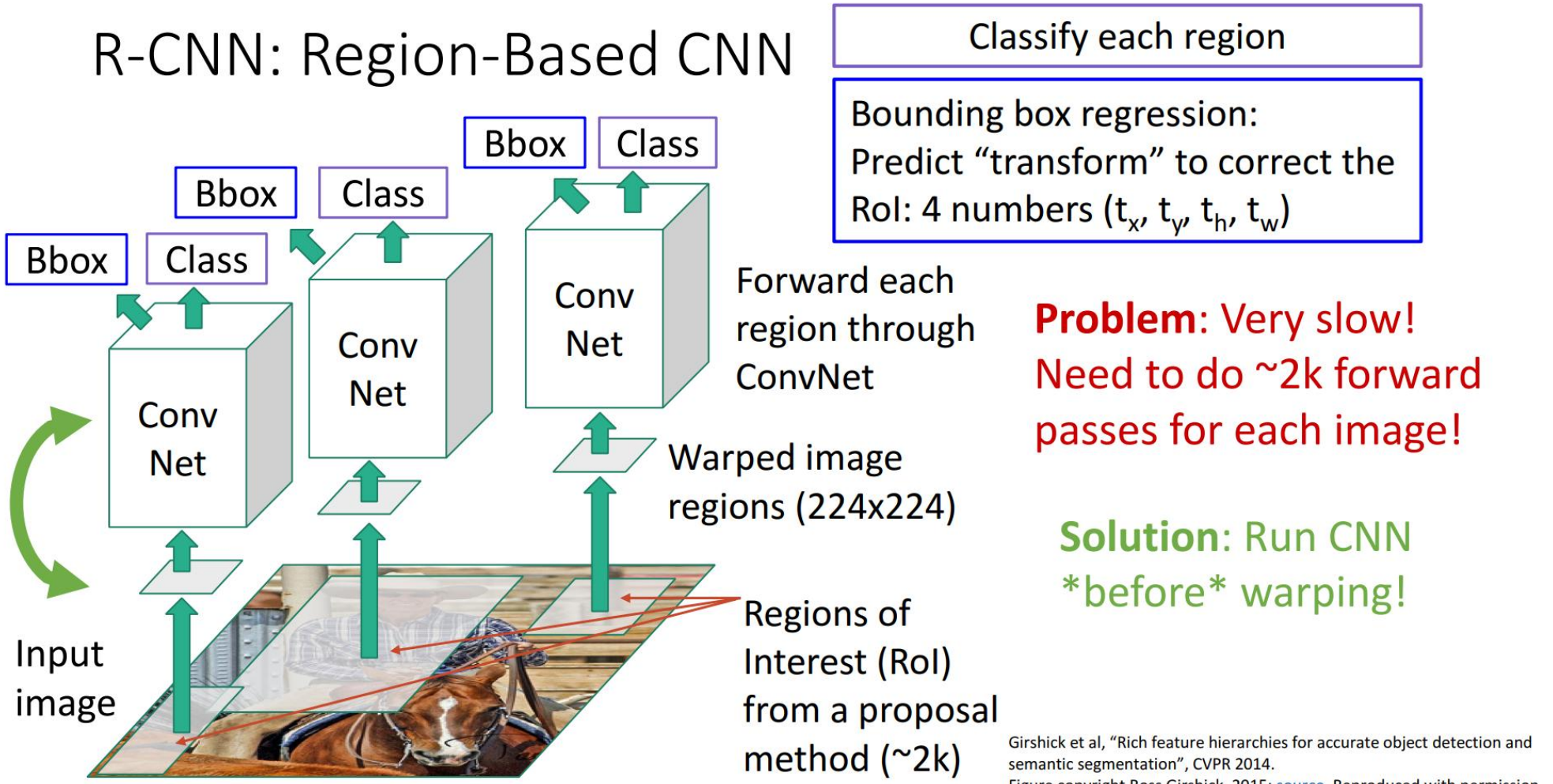
R-CNN: Region-Based Convolutional Neural Network



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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

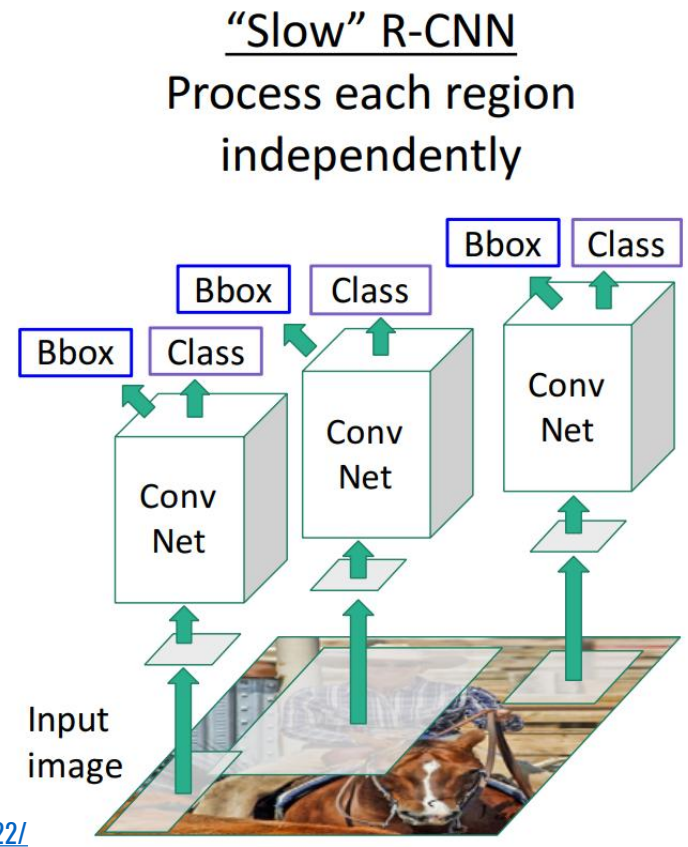
R-CNN: Region-Based Convolutional Neural Network



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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

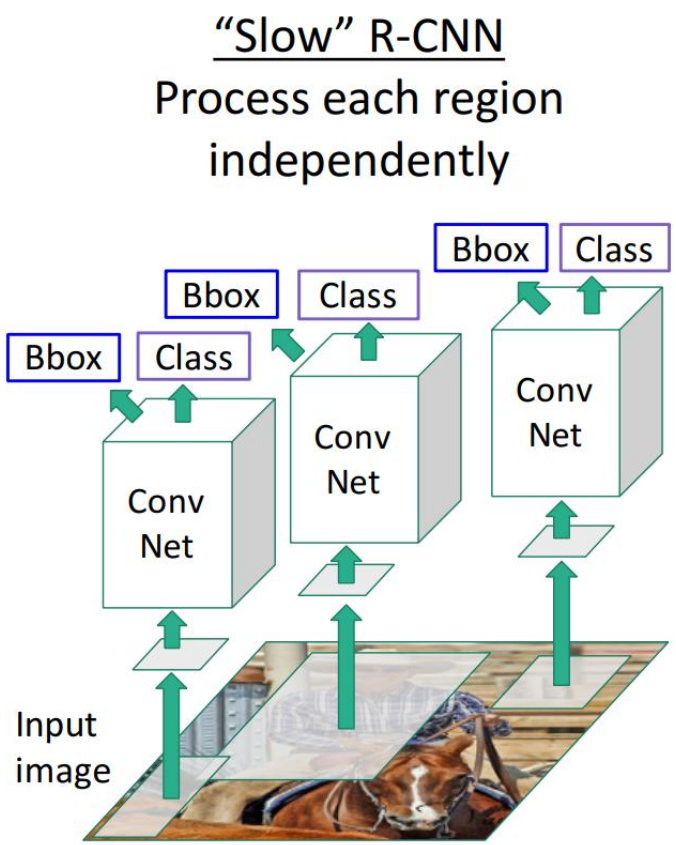
Fast R-CNN: Region-Based Convolutional Neural Network [2]



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

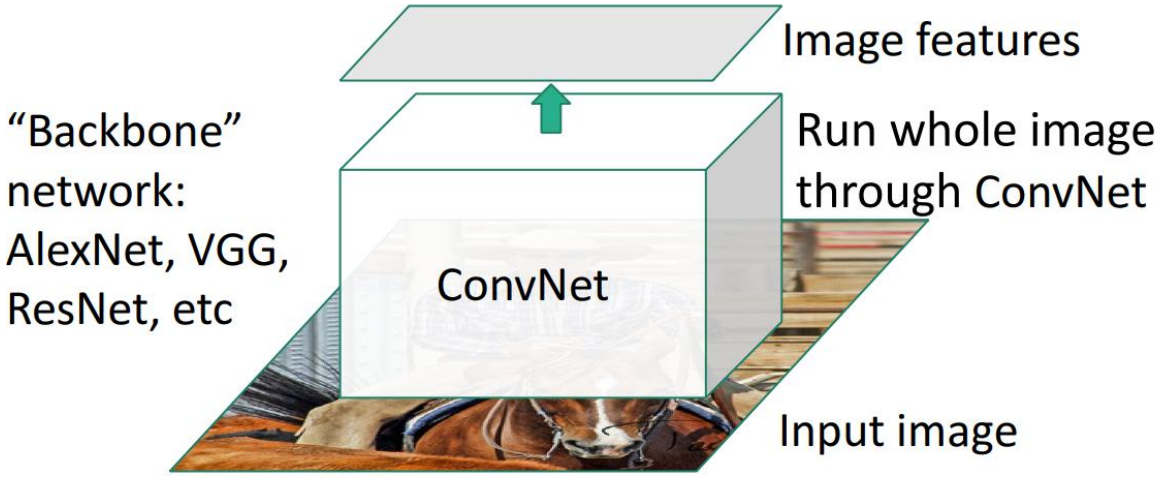
[2] Girshick, R. (2015). Fast R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1504.08083

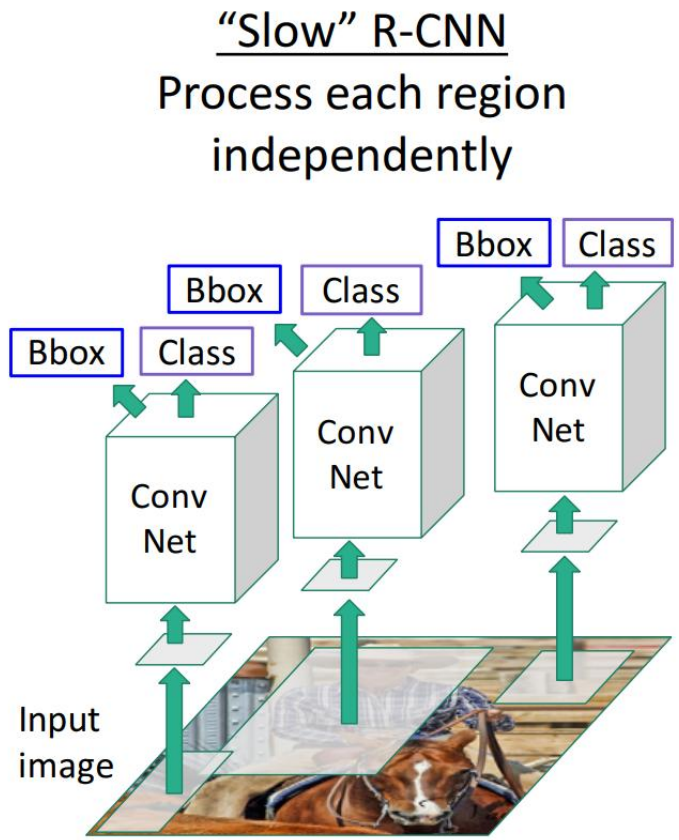
Fast R-CNN: Region-Based Convolutional Neural Network



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

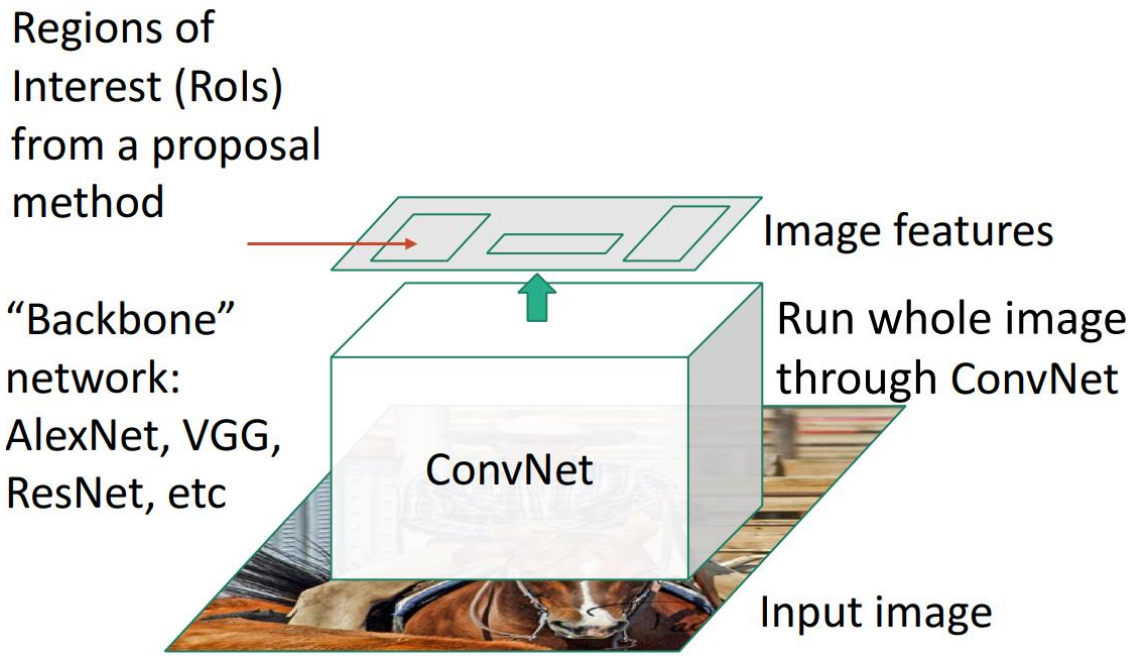
Fast R-CNN: Region-Based Convolutional Neural Network

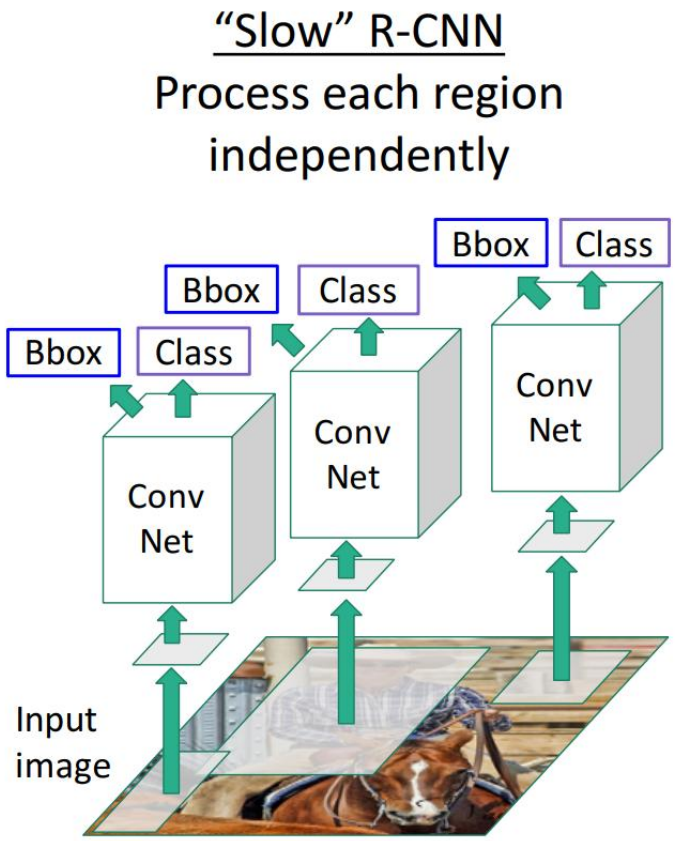




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Fast R-CNN: Region-Based Convolutional Neural Network





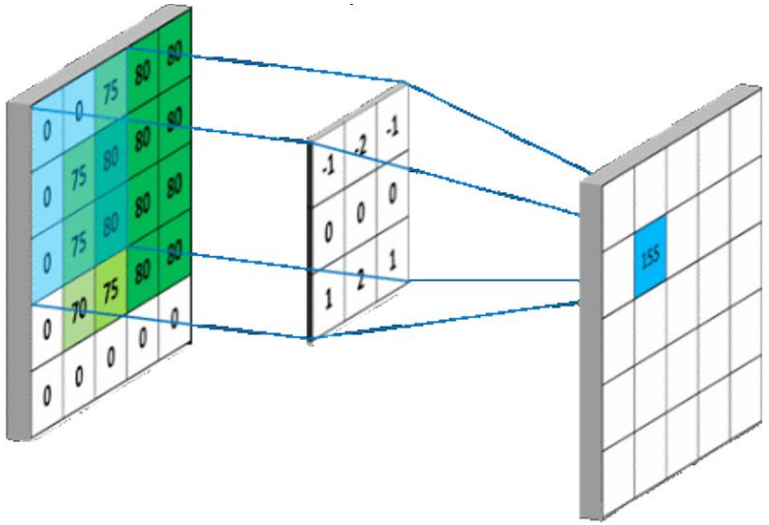
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

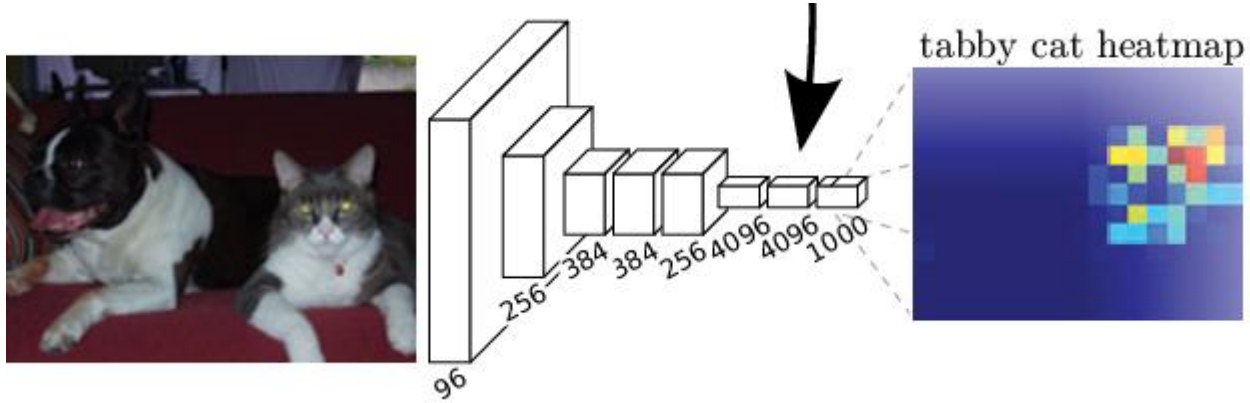
Quick ROI projection to feature map insight

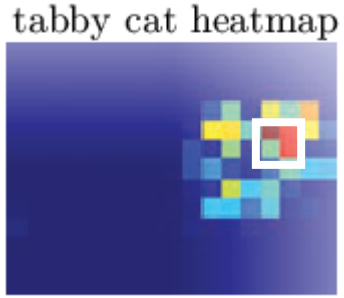
Convolutional Layer

- **Filter / Kernel size = f x f**

Convolution
Input: 5x5
Filter: 3x3
Output: 3x3

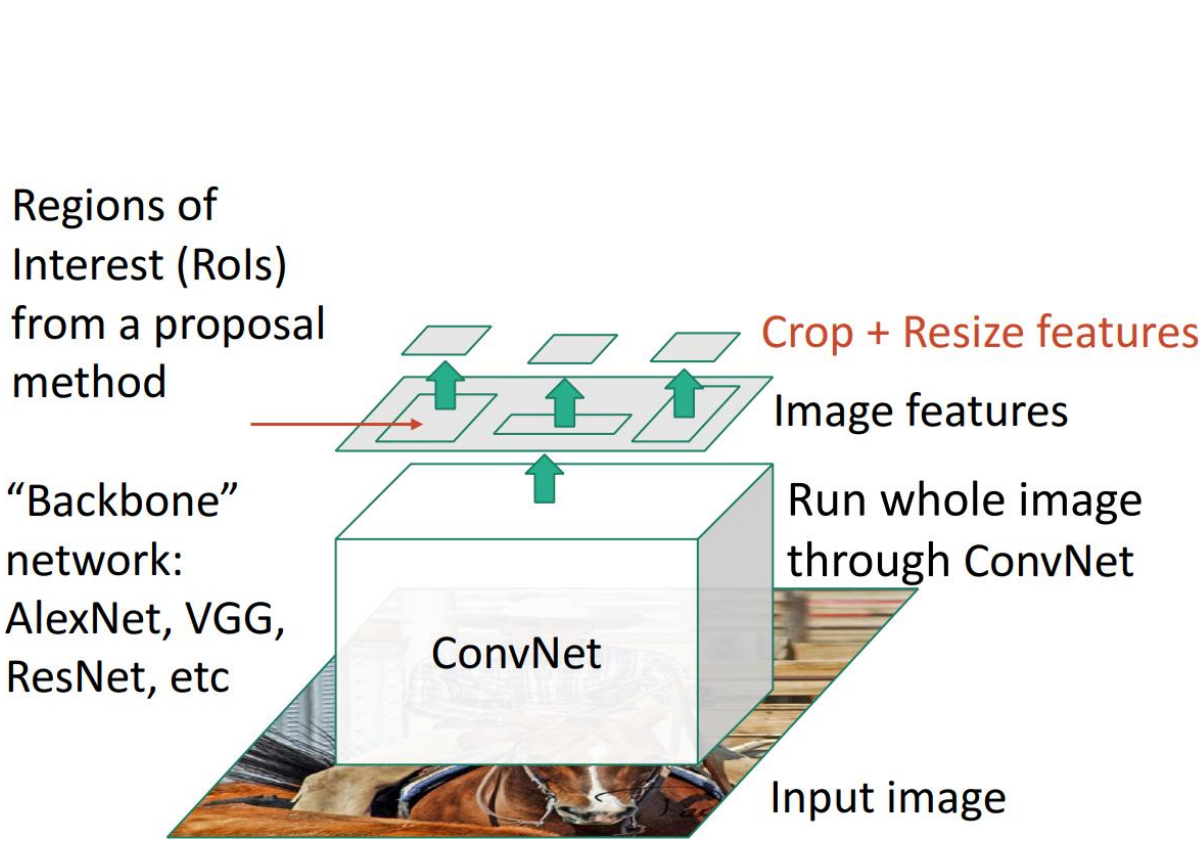


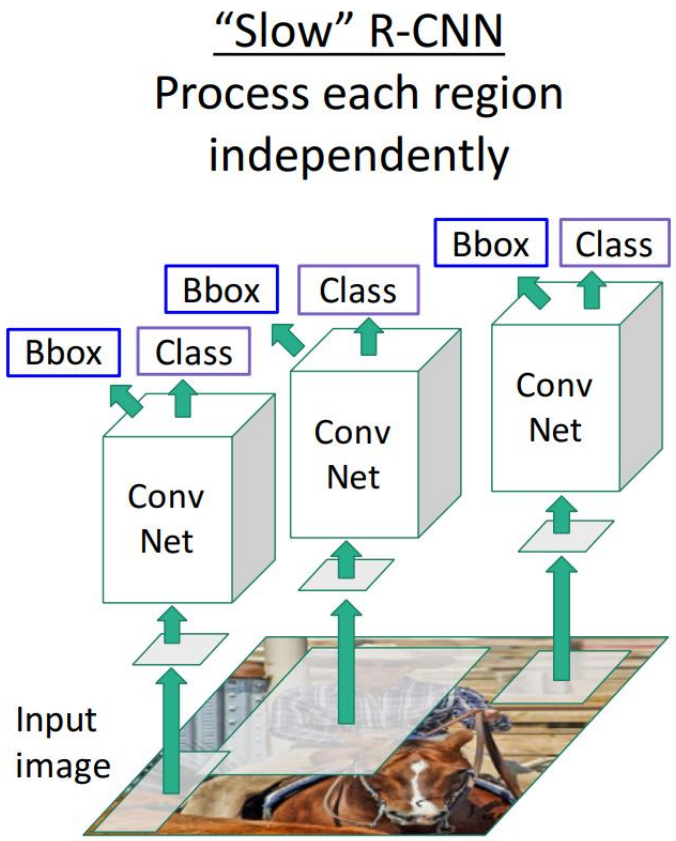




In practice it is harder, there are several tricks. For example: ROI Pool and ROI Align

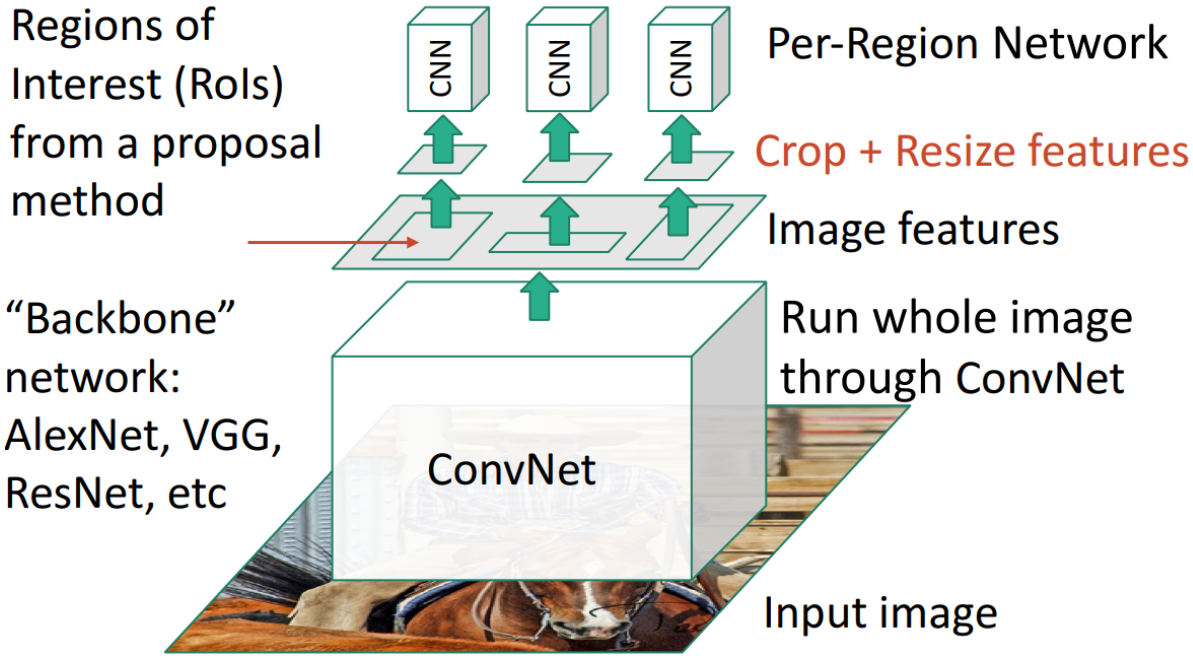
Fast R-CNN: Region-Based Convolutional Neural Network

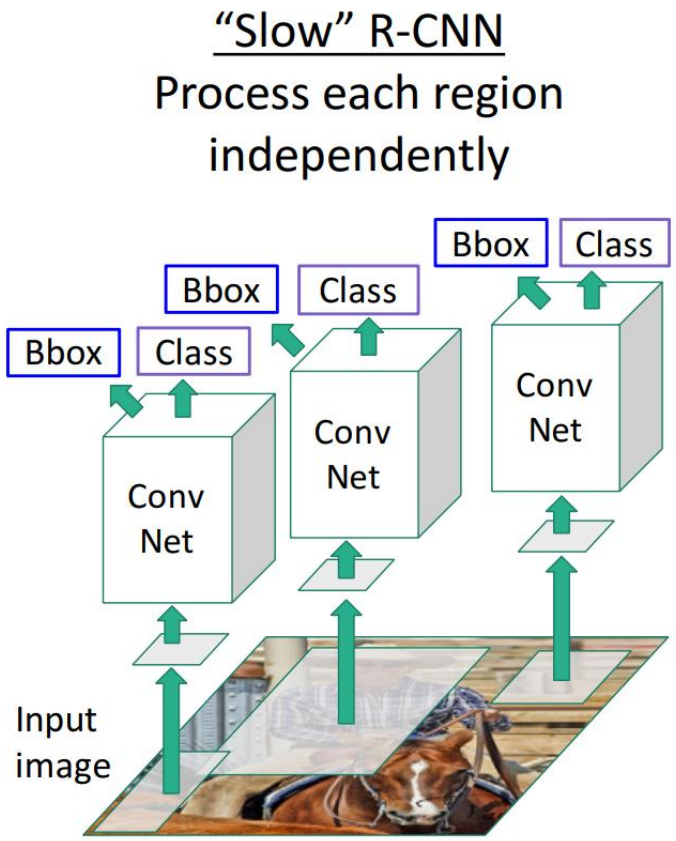




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

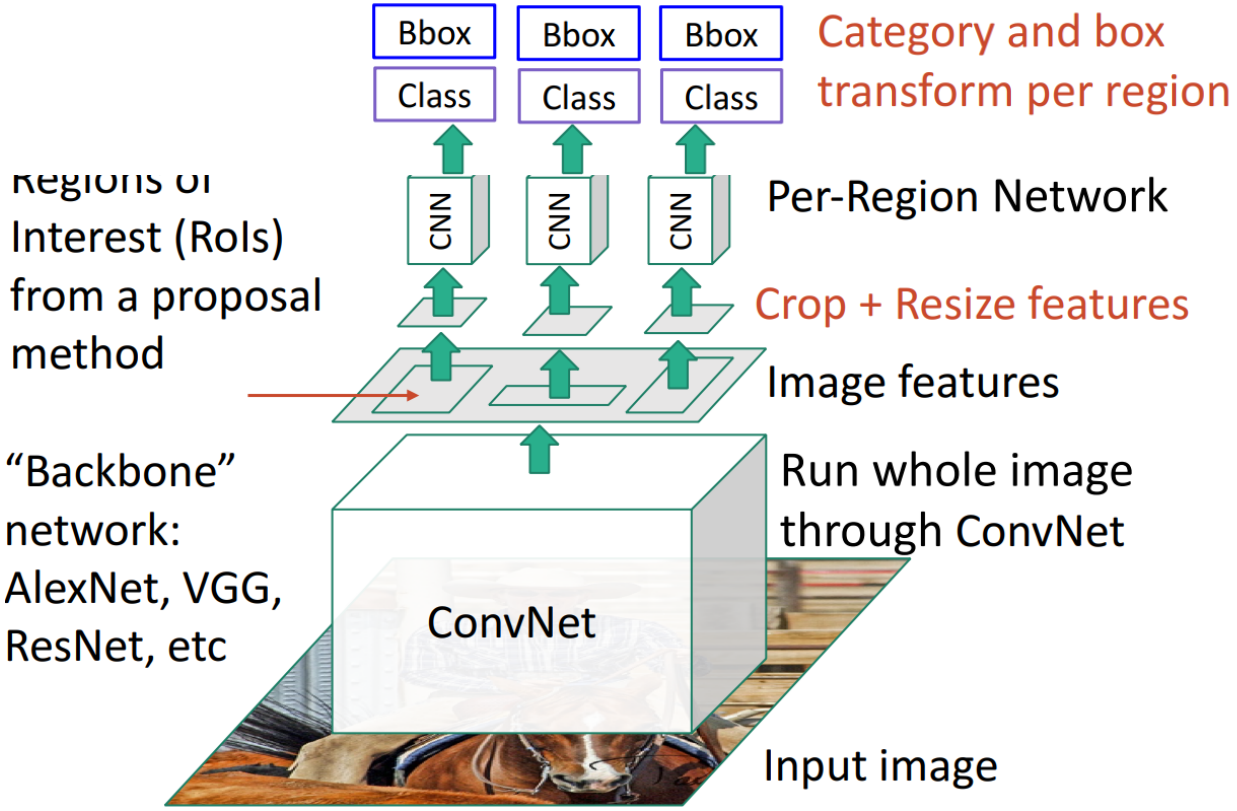
Fast R-CNN: Region-Based Convolutional Neural Network

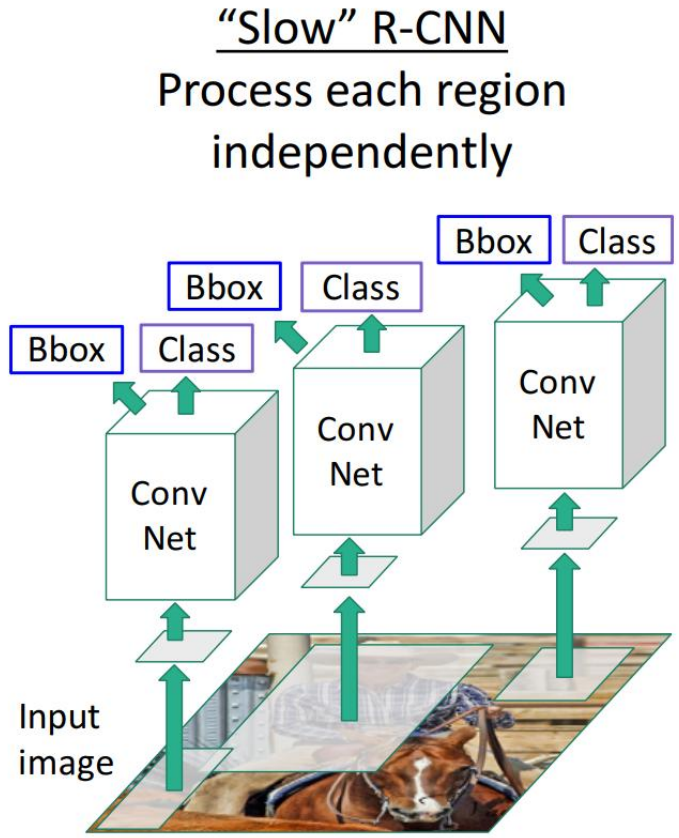




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

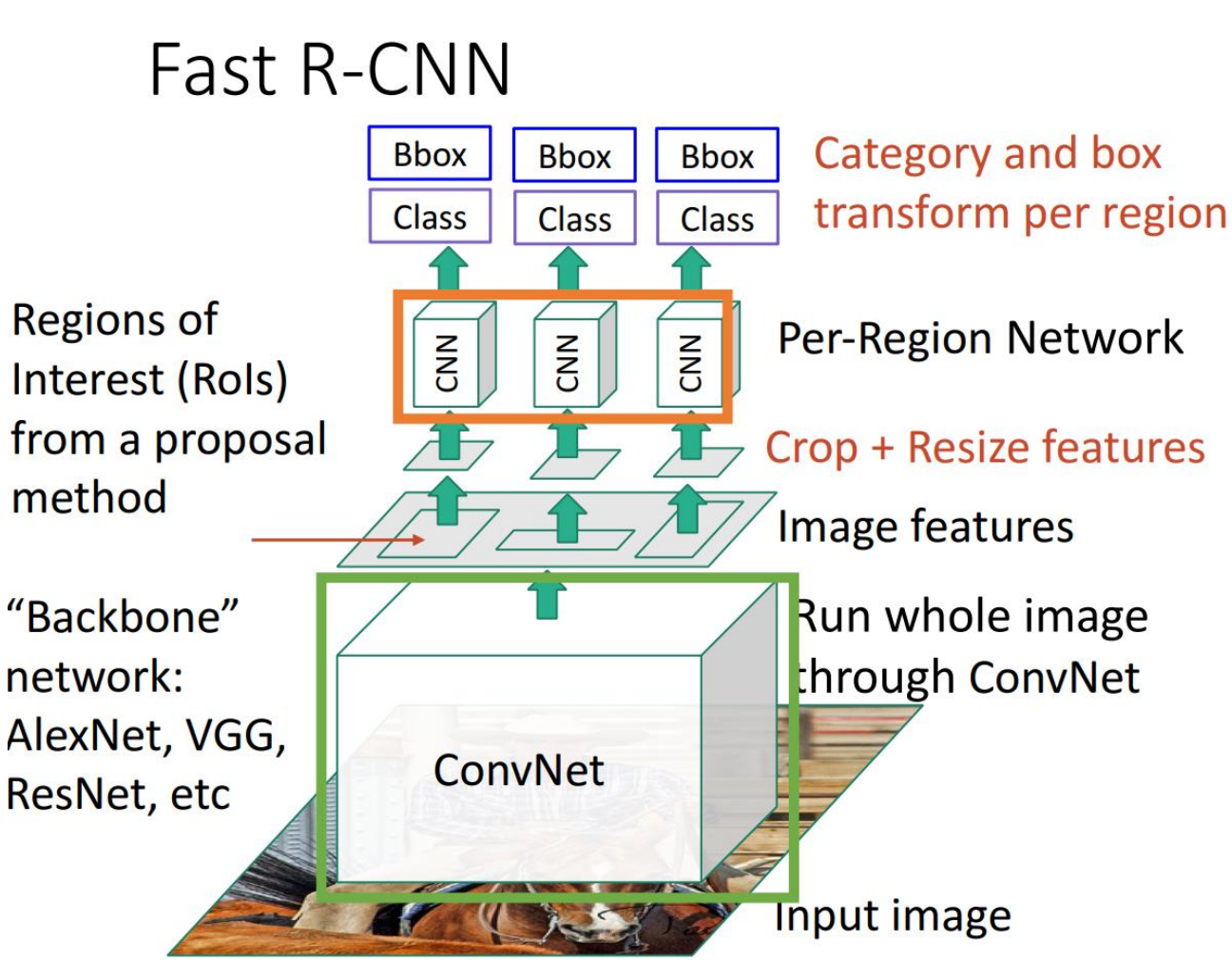
Fast R-CNN: Region-Based Convolutional Neural Network

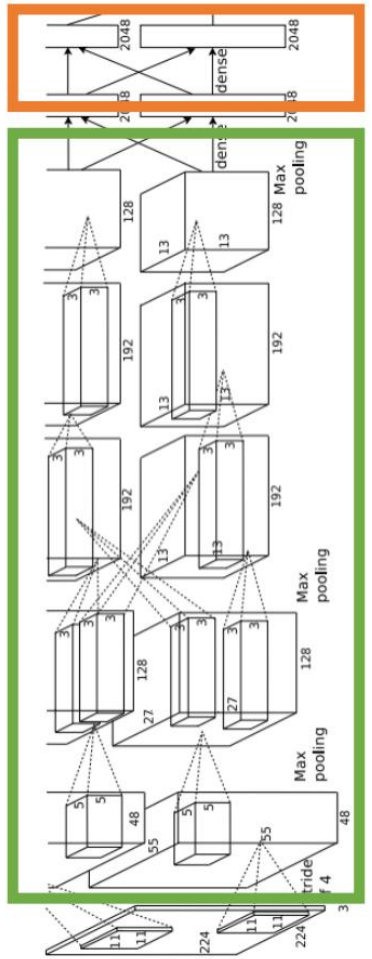




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Fast R-CNN: Region-Based Convolutional Neural Network

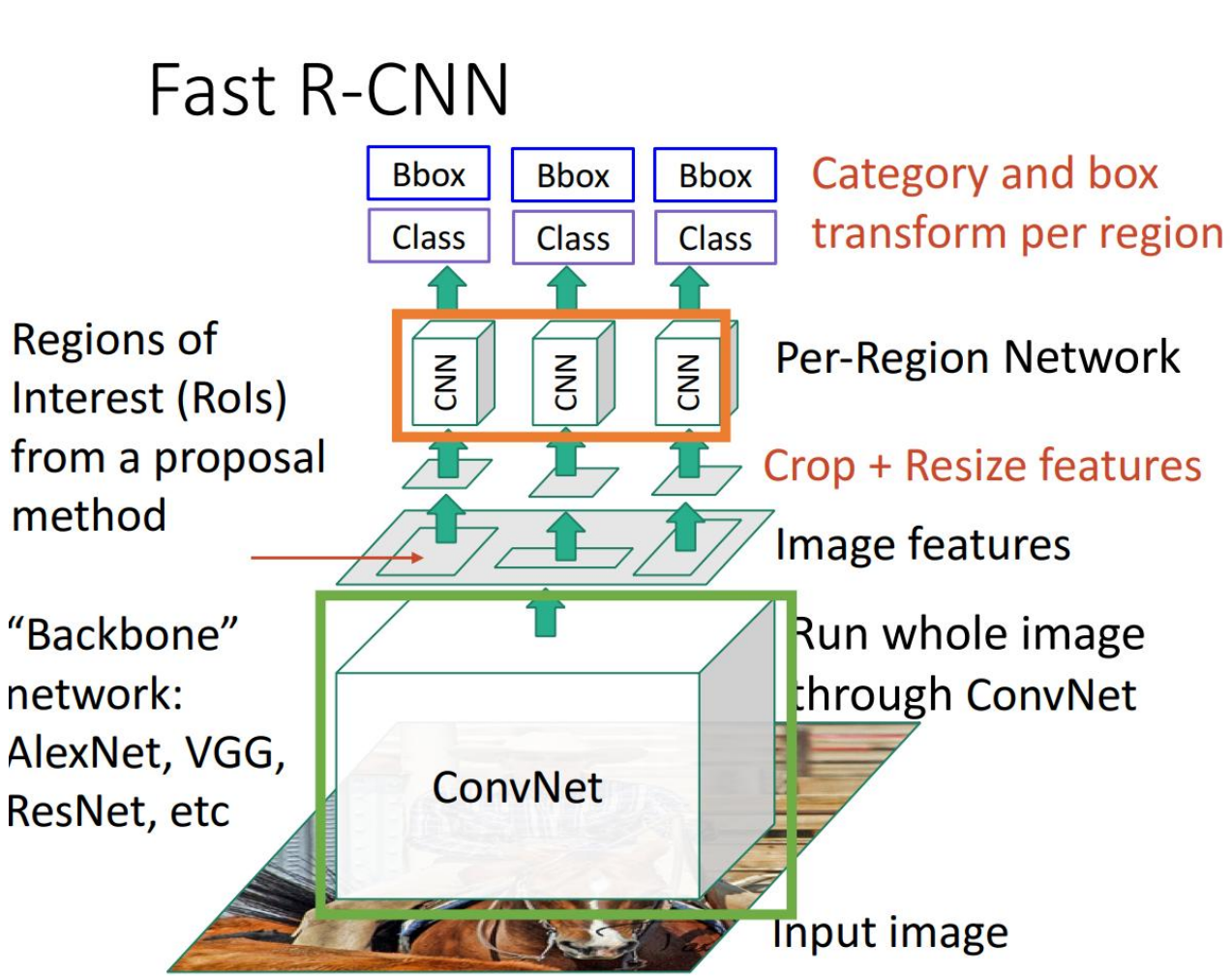


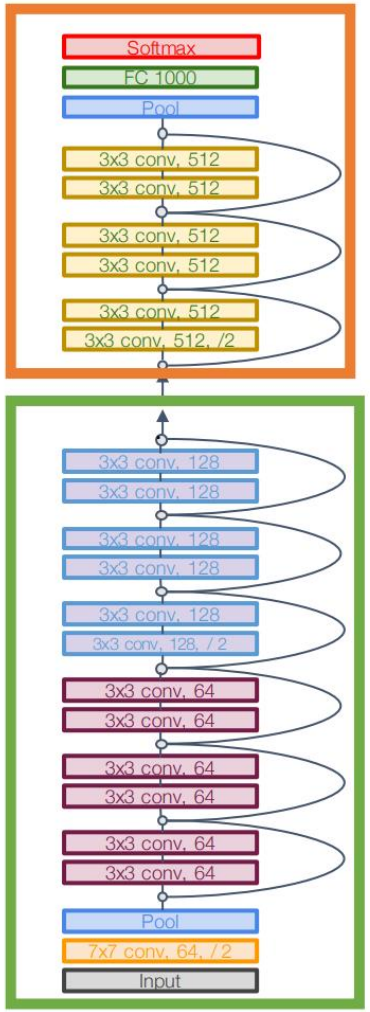


Example:
When using AlexNet for detection, five conv layers are used for backbone and two FC layers are used for per-region network

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Fast R-CNN: Region-Based Convolutional Neural Network

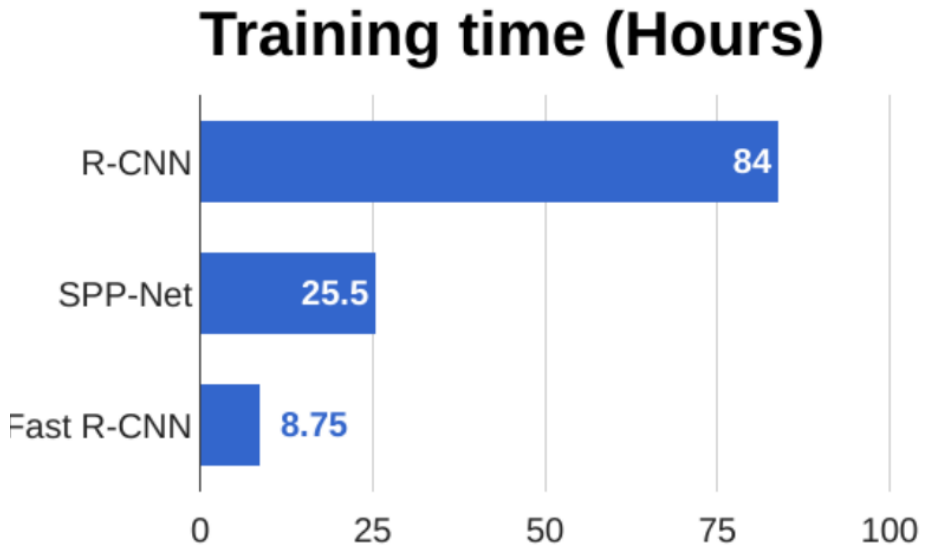


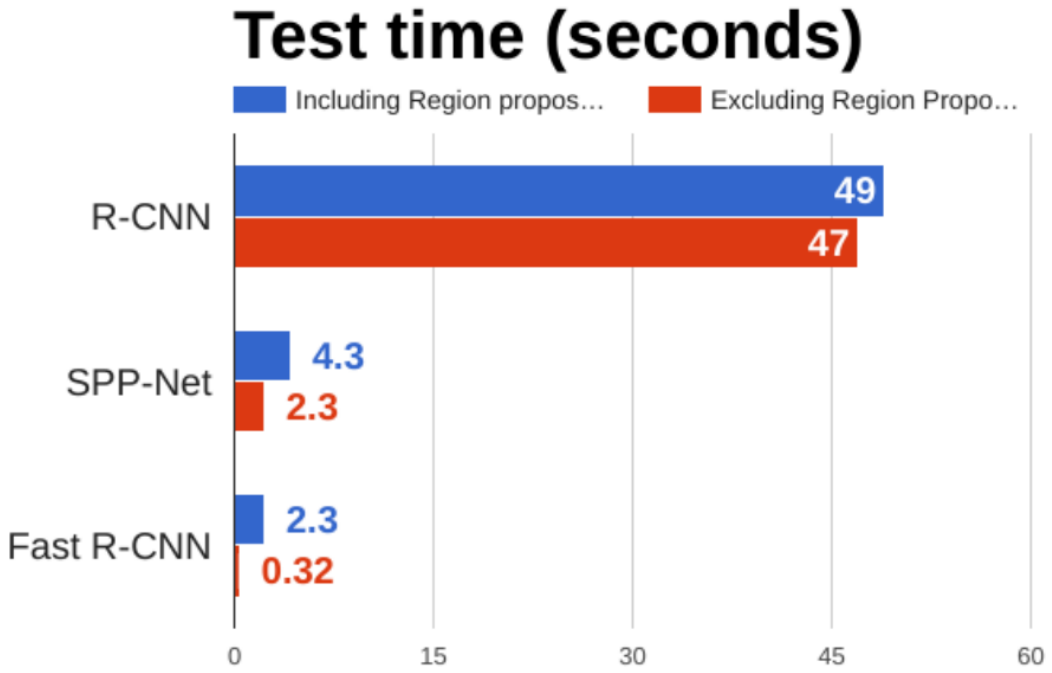


Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

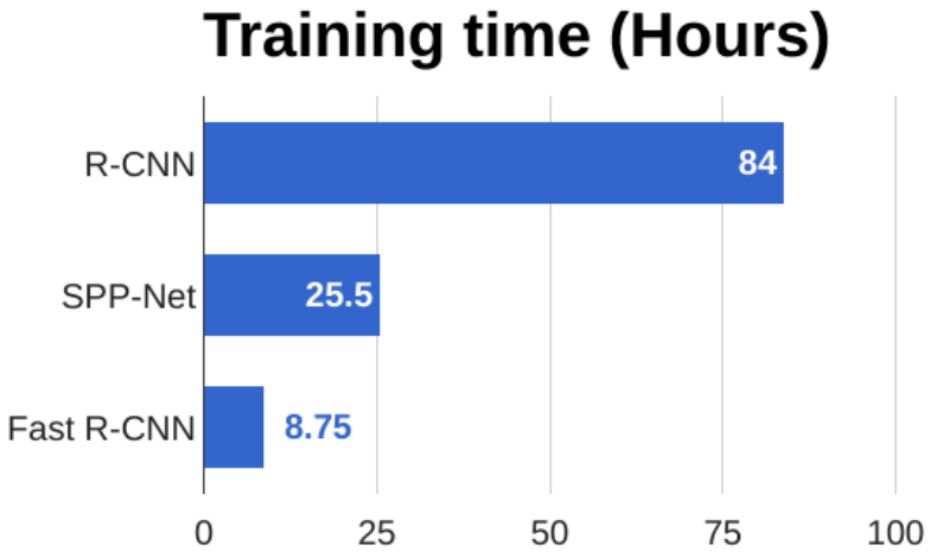
Fast R-CNN vs “Slow” R-CNN

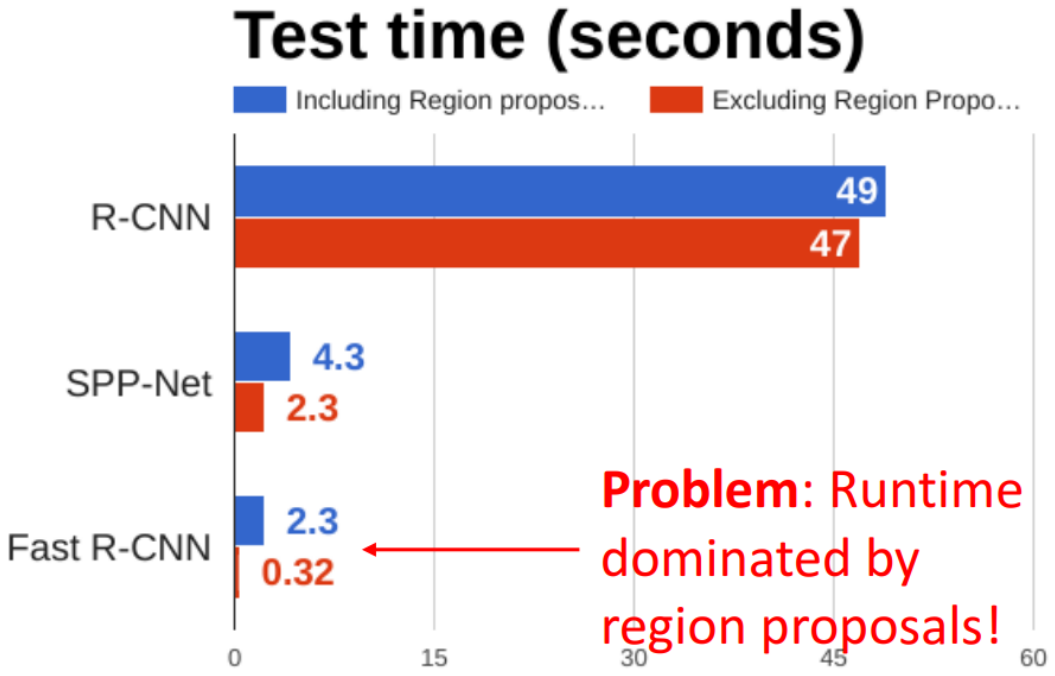




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Fast R-CNN vs “Slow” R-CNN

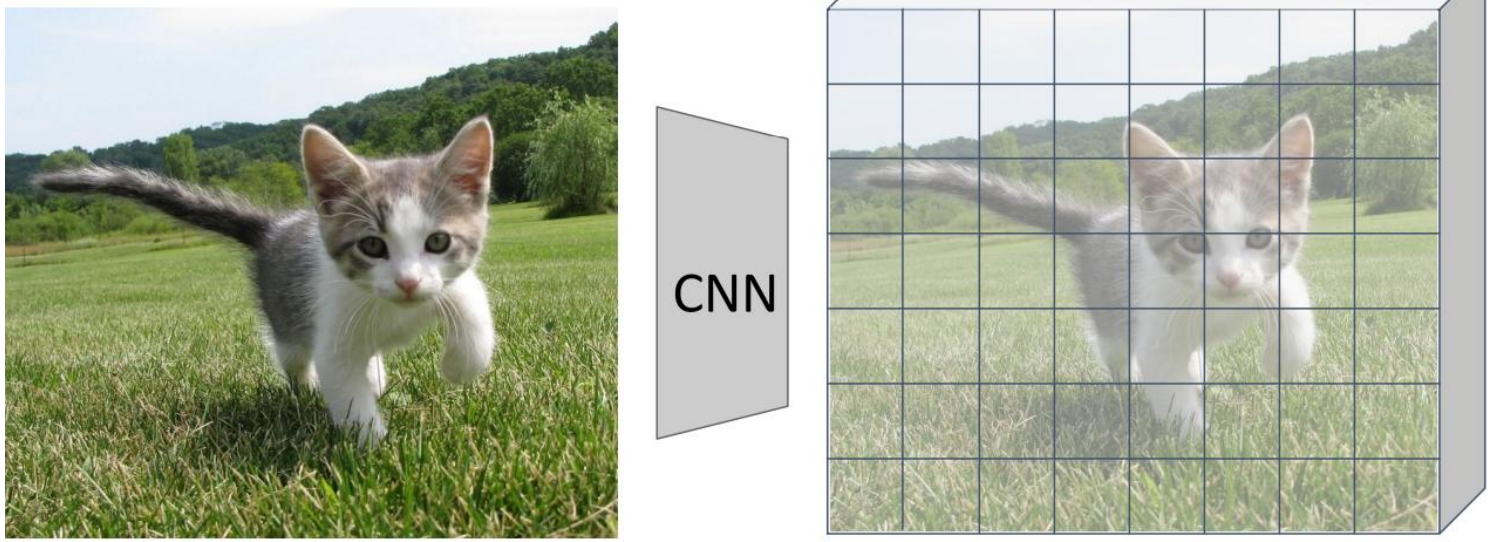




Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

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)

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

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

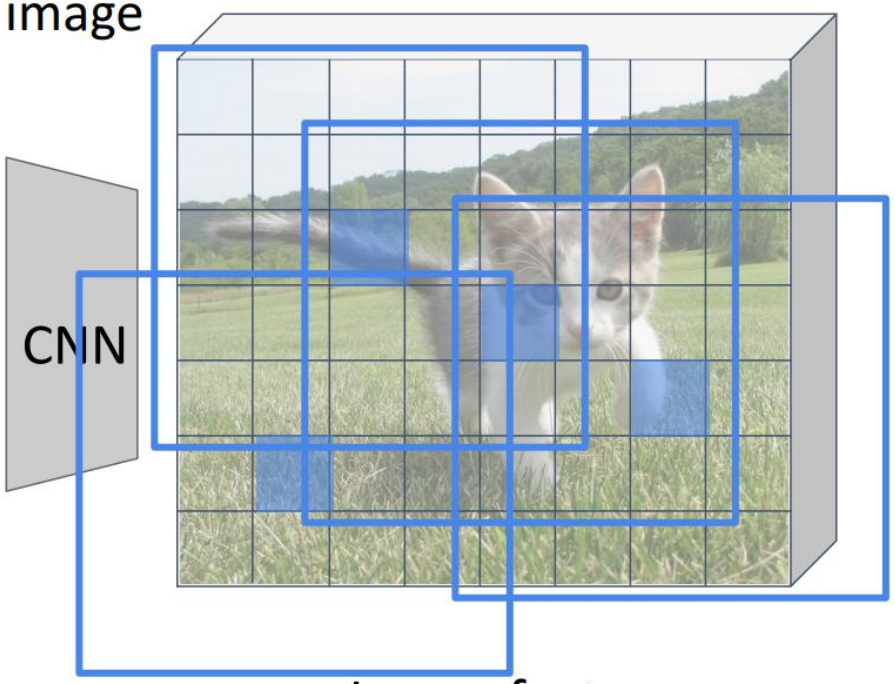


Image features
(e.g. 512 x 20 x 15)

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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

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

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

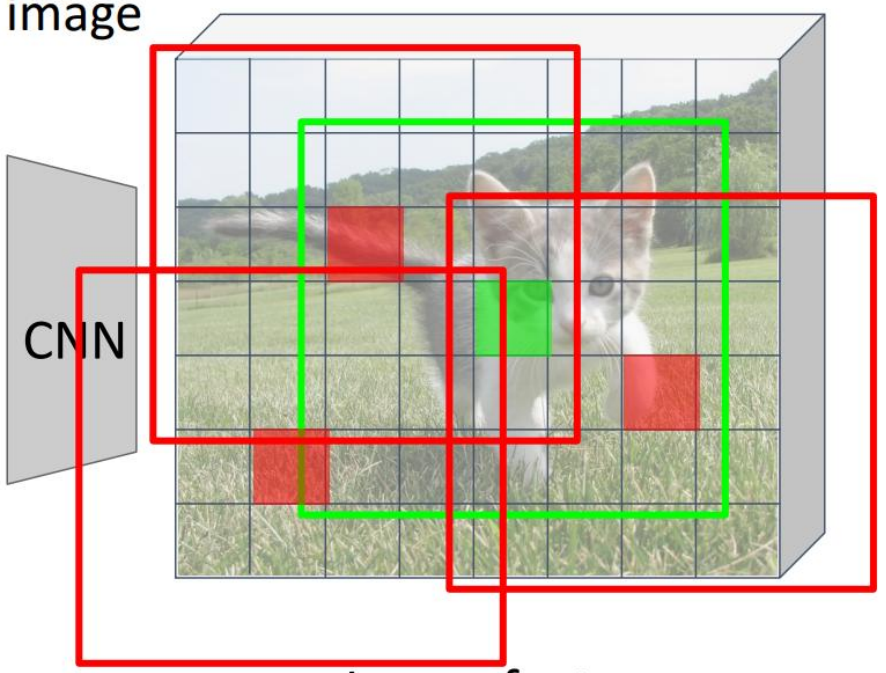


Image features
(e.g. 512 x 20 x 15)

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)

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

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

CNN

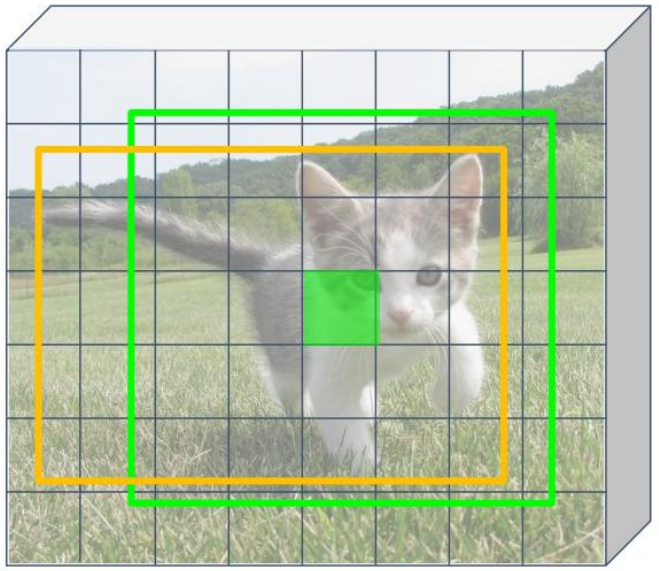


Image features
(e.g. 512 x 20 x 15)

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

Conv

Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

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

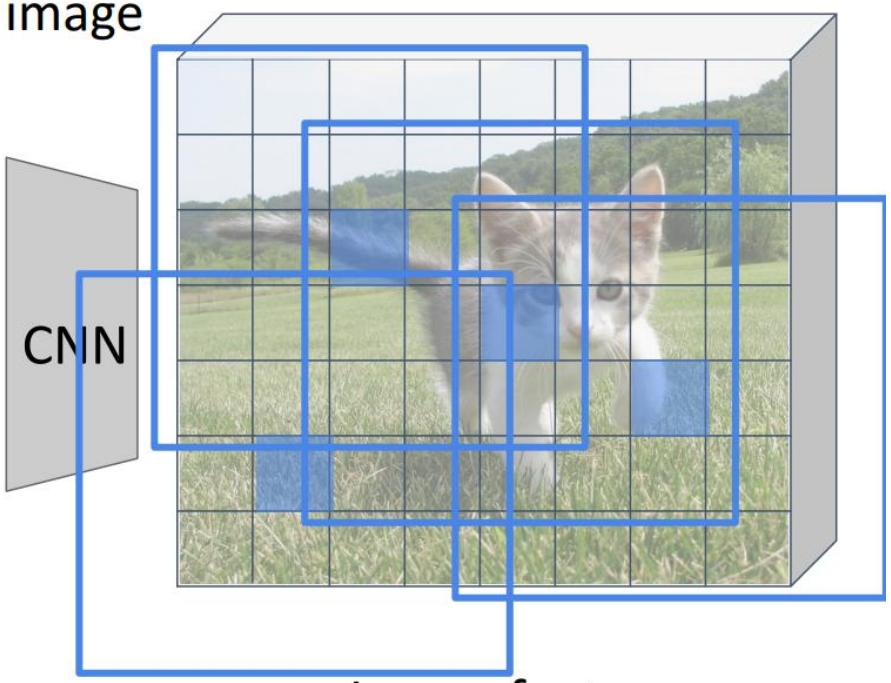
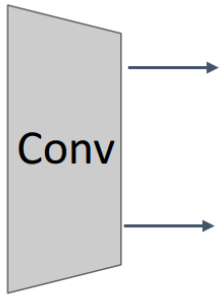


Image features
(e.g. 512 x 20 x 15)

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



Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

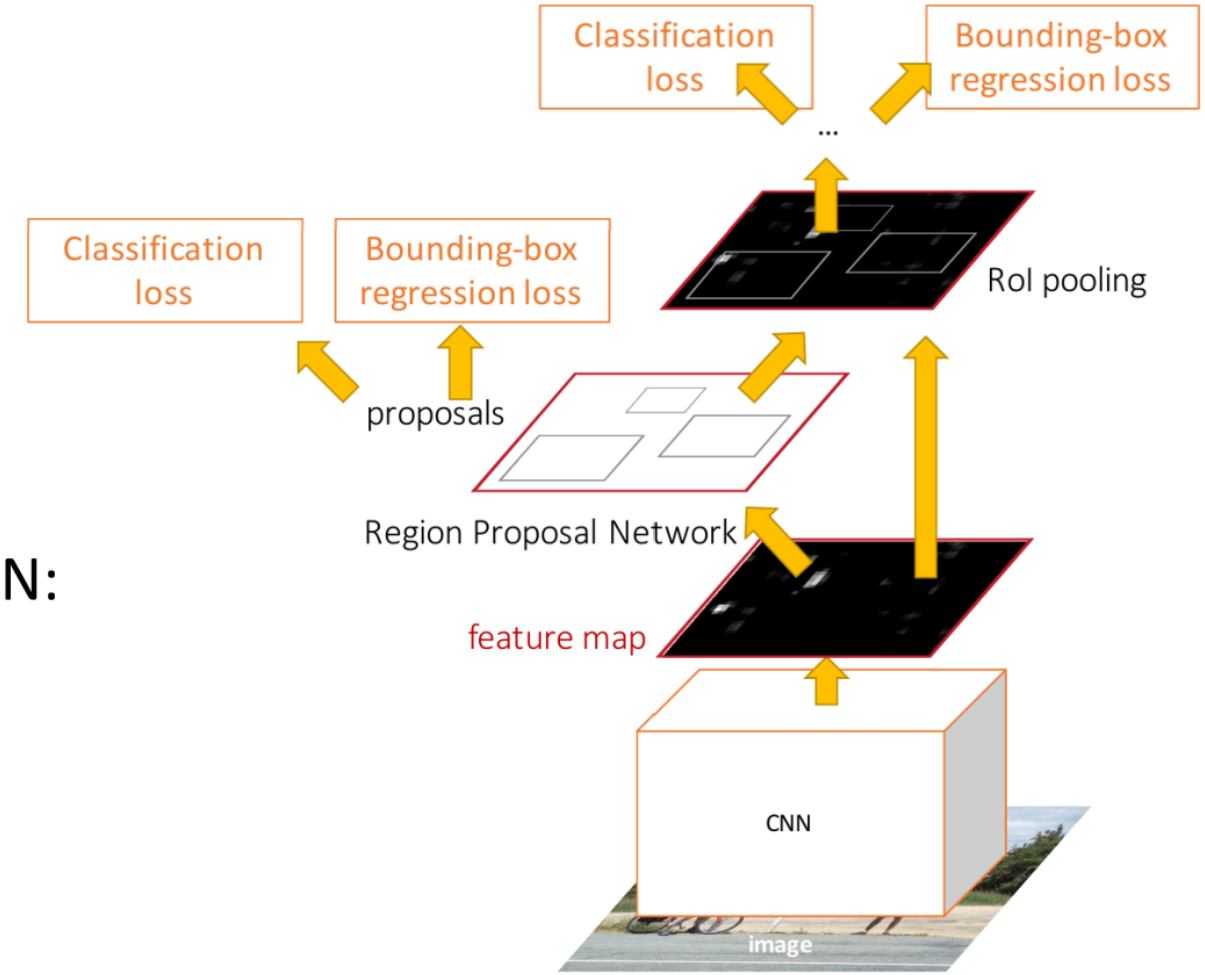
At test time: sort all K*20*15 boxes by their score, and take the top ~300 as our region proposals

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Faster R-CNN: Learnable Region Proposals [3]

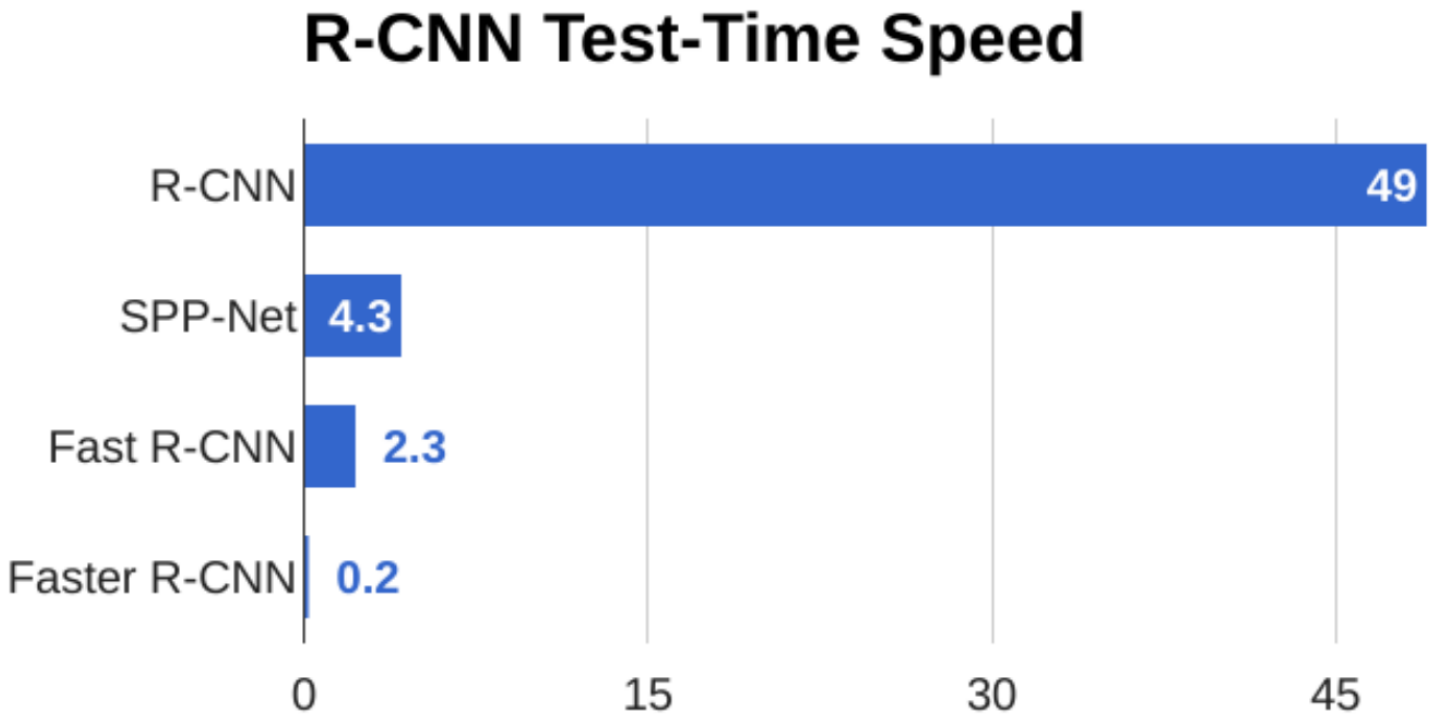
Insert **Region Proposal Network (RPN)** to predict proposals from features

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



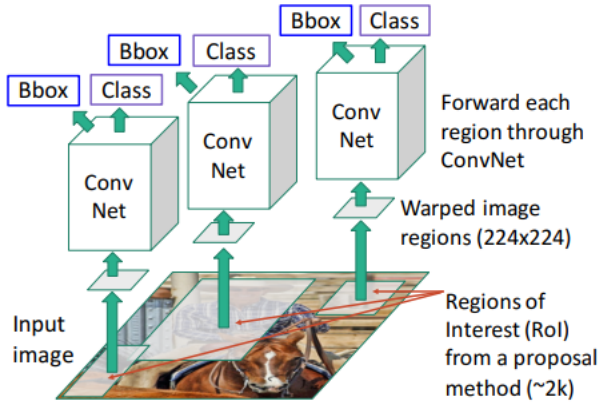
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/
[3] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

Faster R-CNN: Learnable Region Proposals [3]

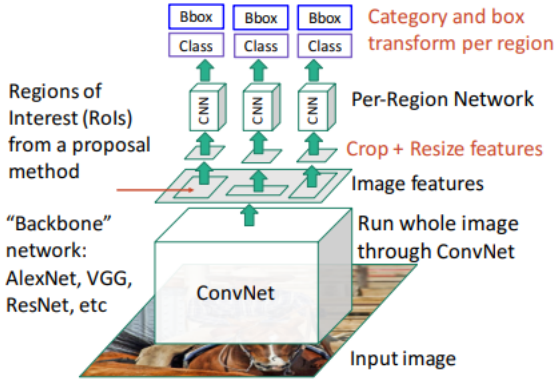


R-CNN Family Comparison

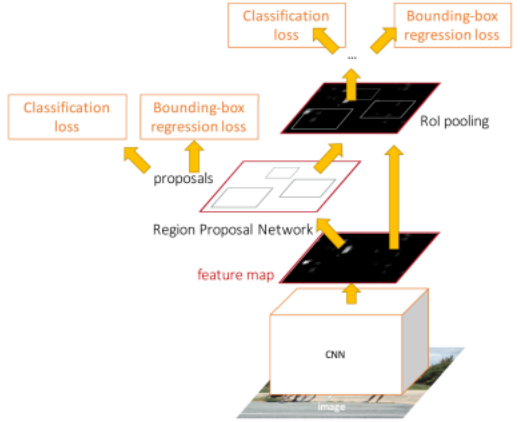
"Slow" R-CNN: Run CNN independently for each region



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



Faster R-CNN: Compute proposals with CNN



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

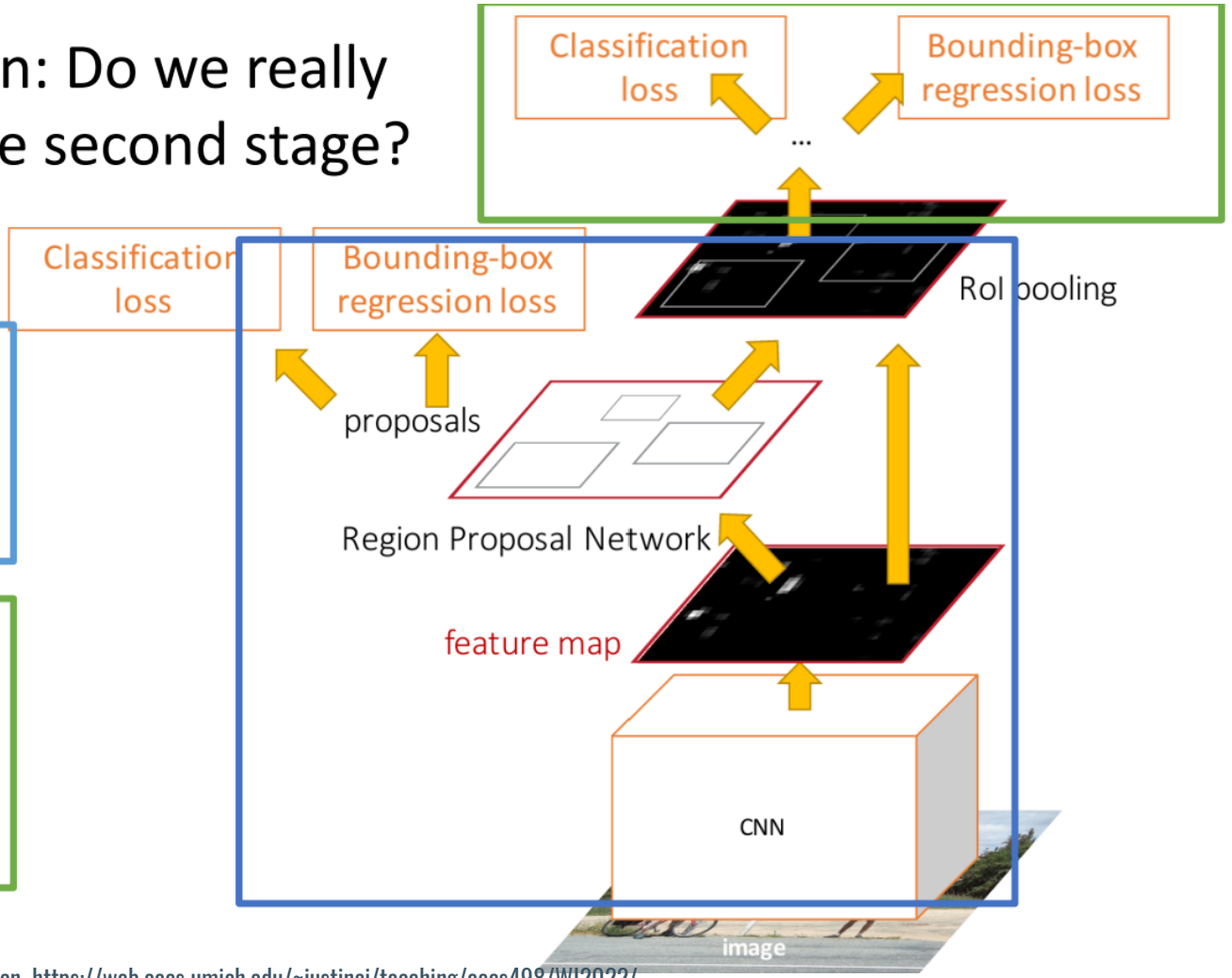
R-CNN Family

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

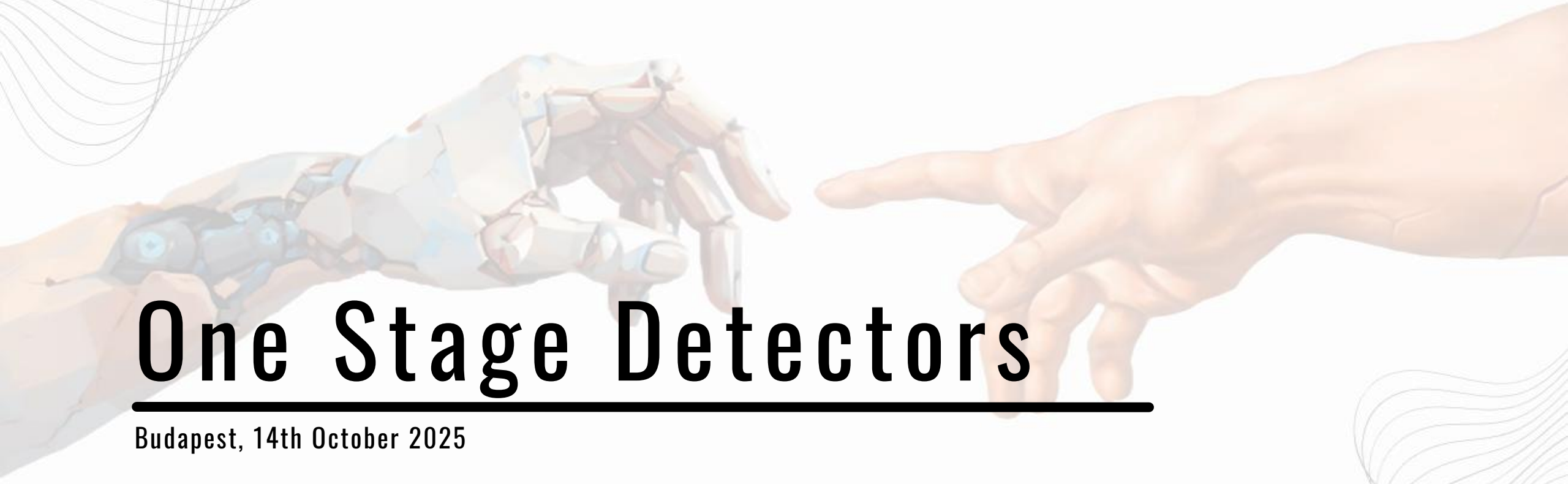
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Deep Network Development

Lecture 8.



One Stage Detectors

Budapest, 14th October 2025

1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

Problems with Faster R-CNN

Run backbone CNN to get features aligned to input image

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

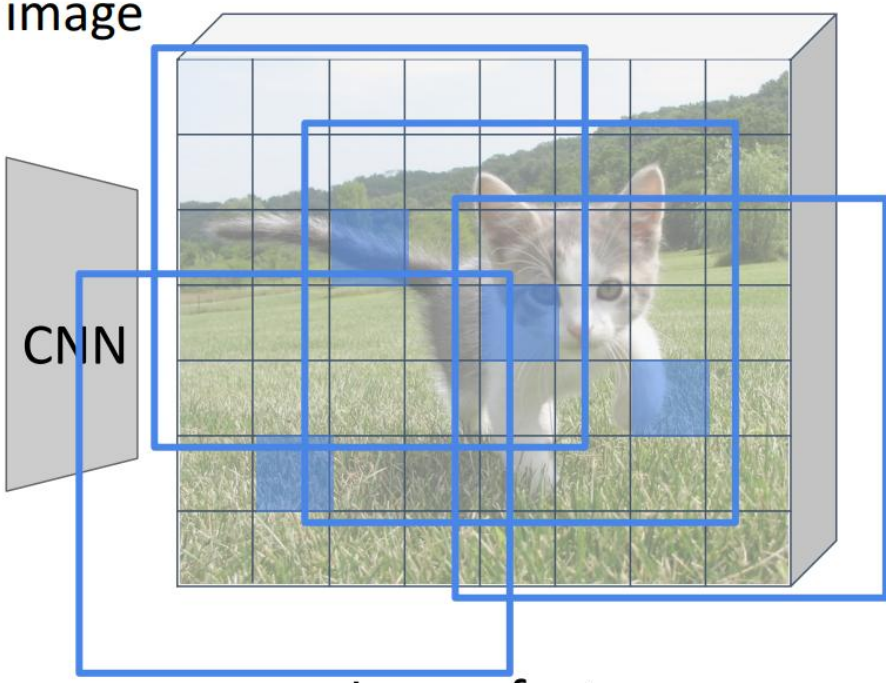


Image features
(e.g. 512 x 20 x 15)

Problem: Anchor box may have the wrong size / shape

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

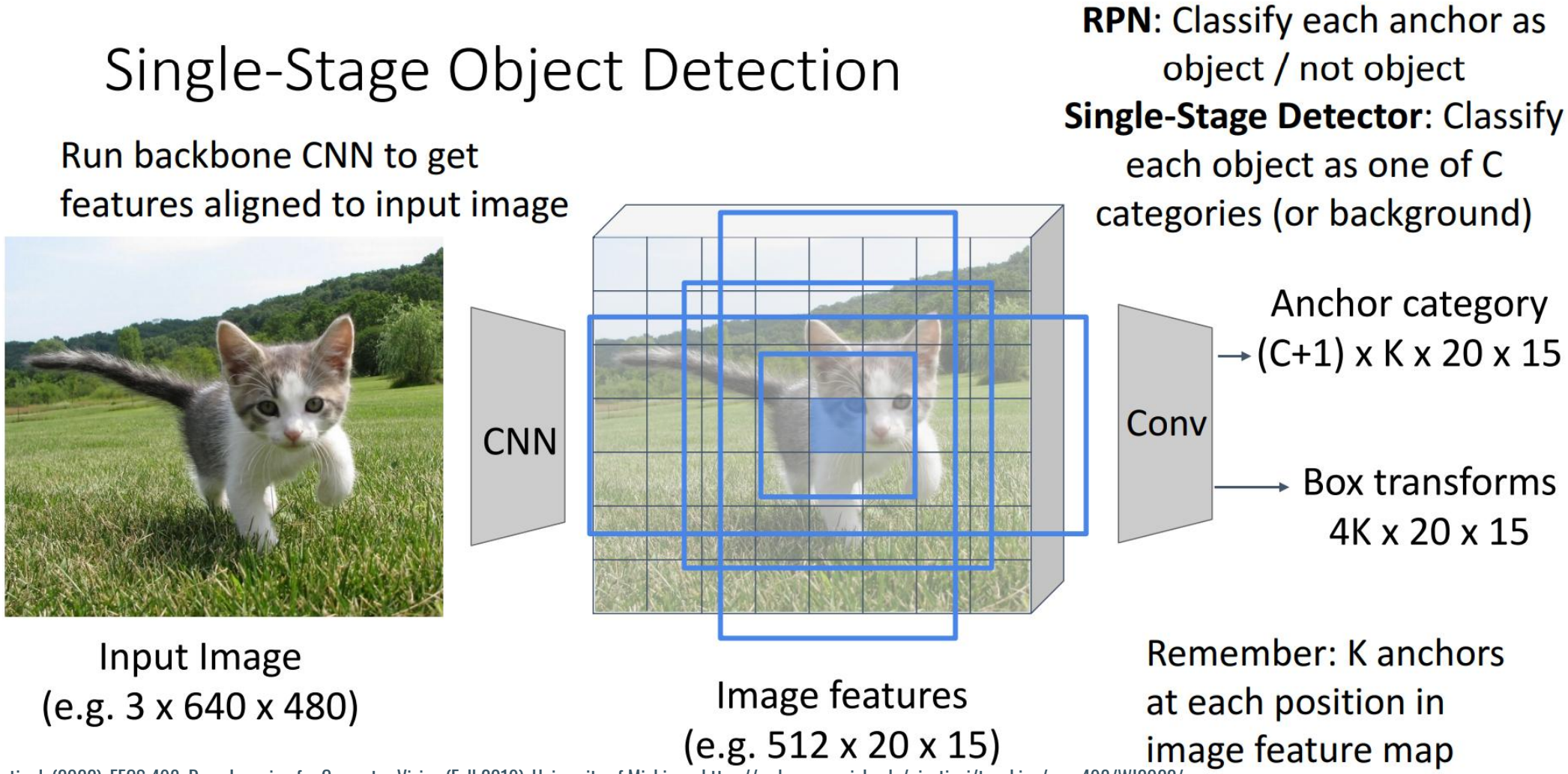
Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Problems with Faster R-CNN

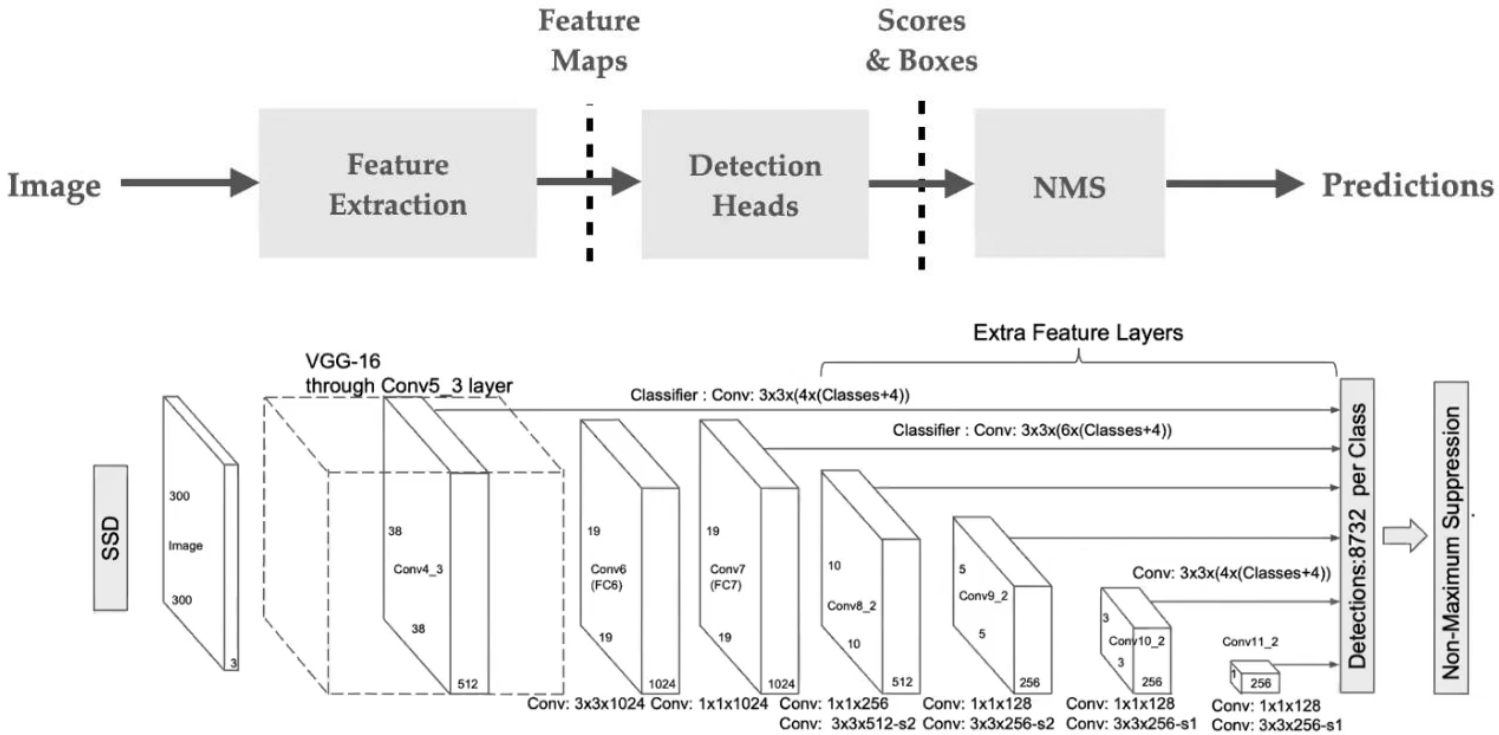


Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Single-Shot Detectors (SSD) [4]

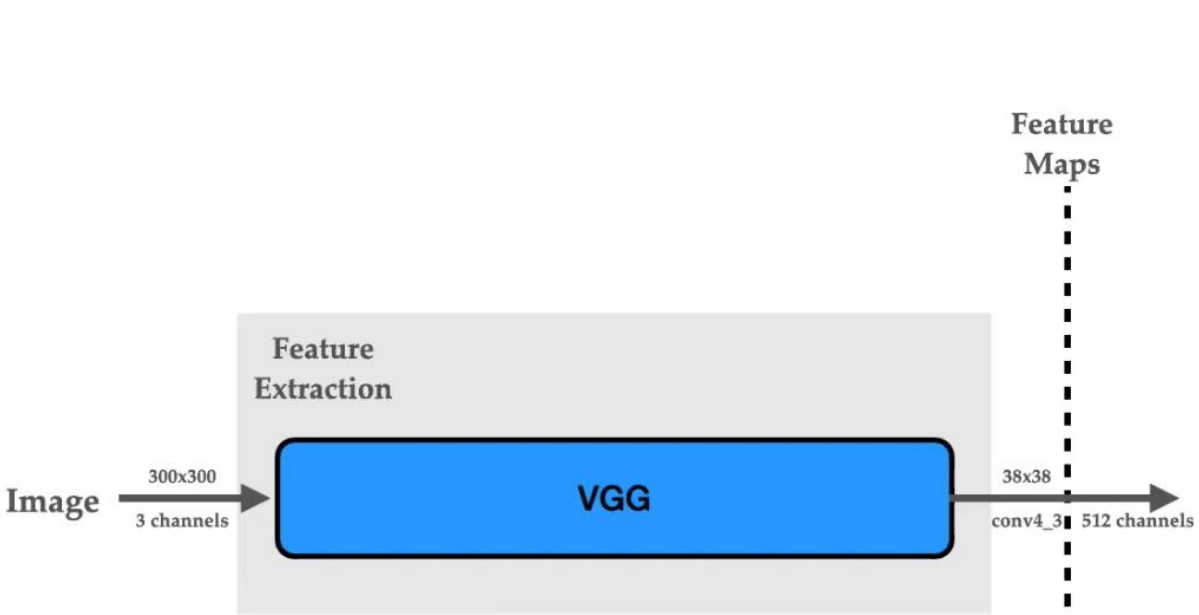
Key Ideas:

1. Multiple Layers → handle different scales
2. Different filters predict boxes of different shapes/sizes



[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD) [4]

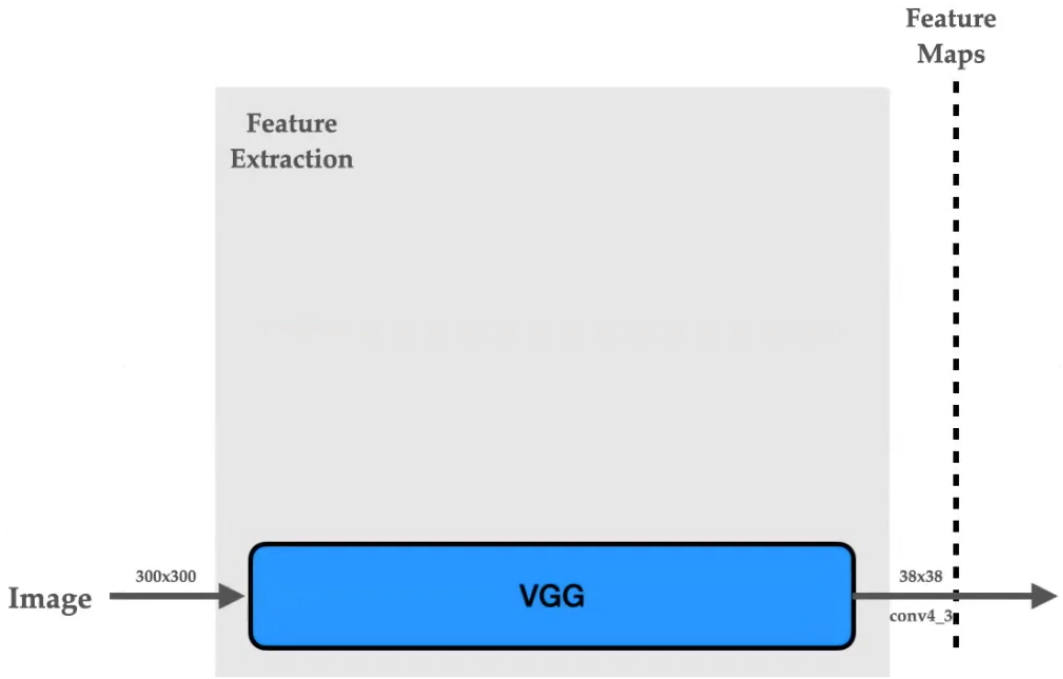


VGG

conv3-64
conv3-64
conv3-128
conv3-128
conv3-256
conv3-256
conv3-256
conv3-256
conv3-512
conv3-512
conv3-512
conv3-512
conv3-512
conv3-512
conv3-512
conv3-512
maxpool
FC-4096
FC-4096
FC-1000
soft-max

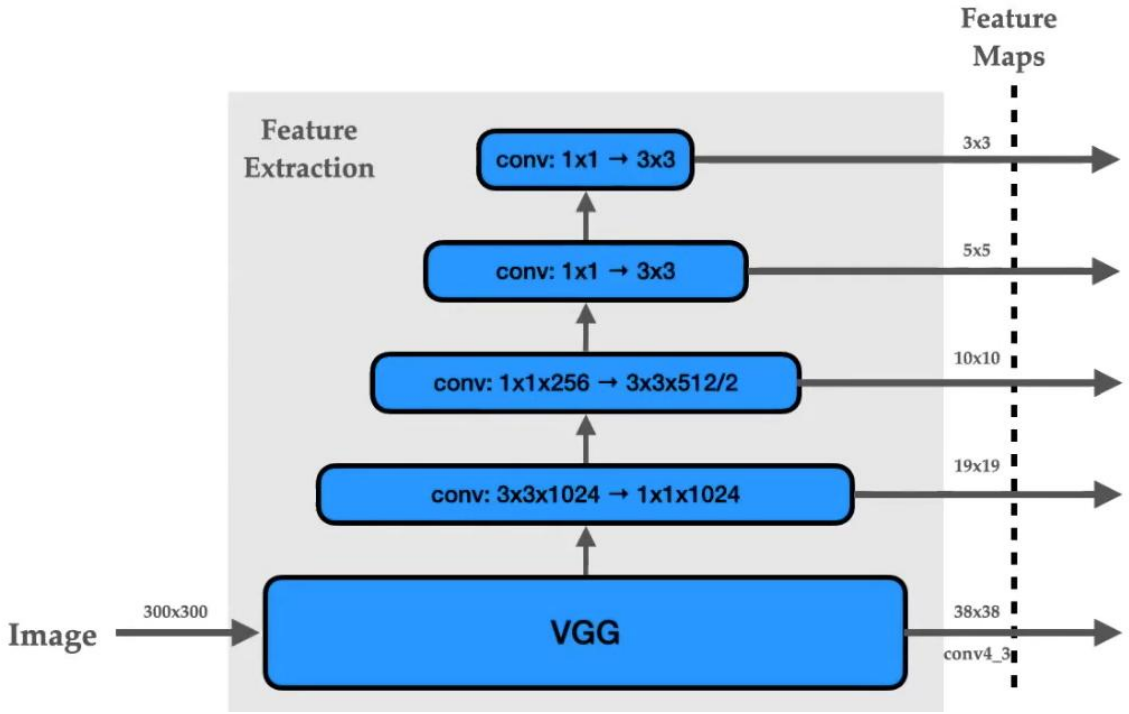
[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD) [4]



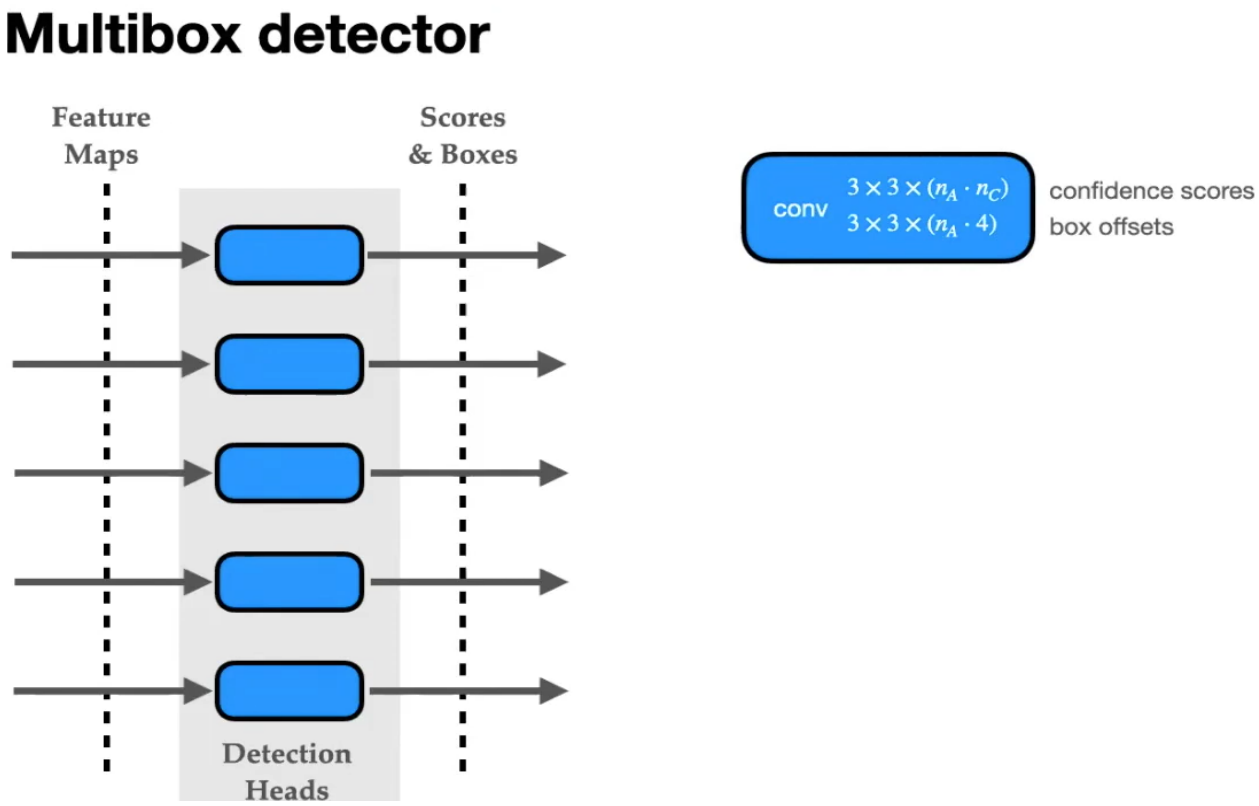
[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD) [4]



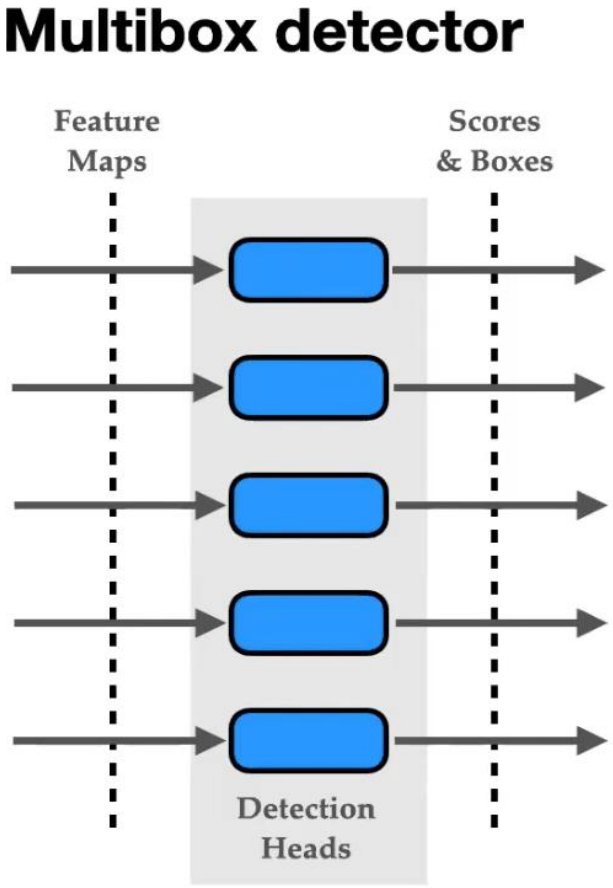
[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

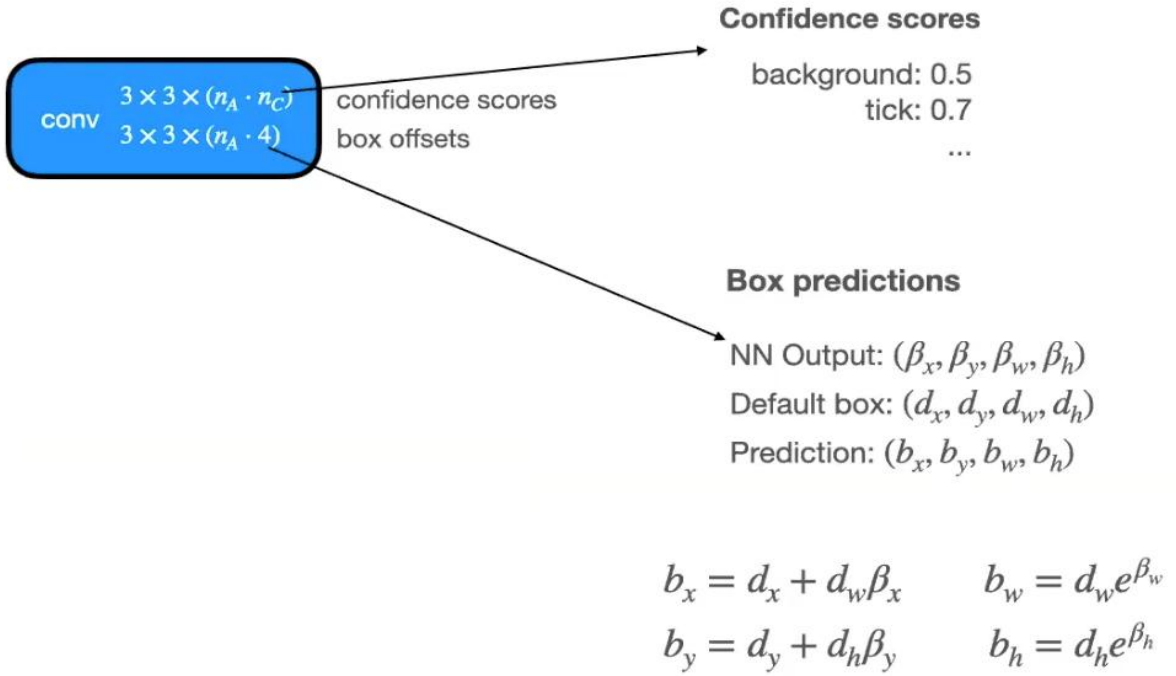
Single-Shot Detectors (SSD) [4]



[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD) [4]

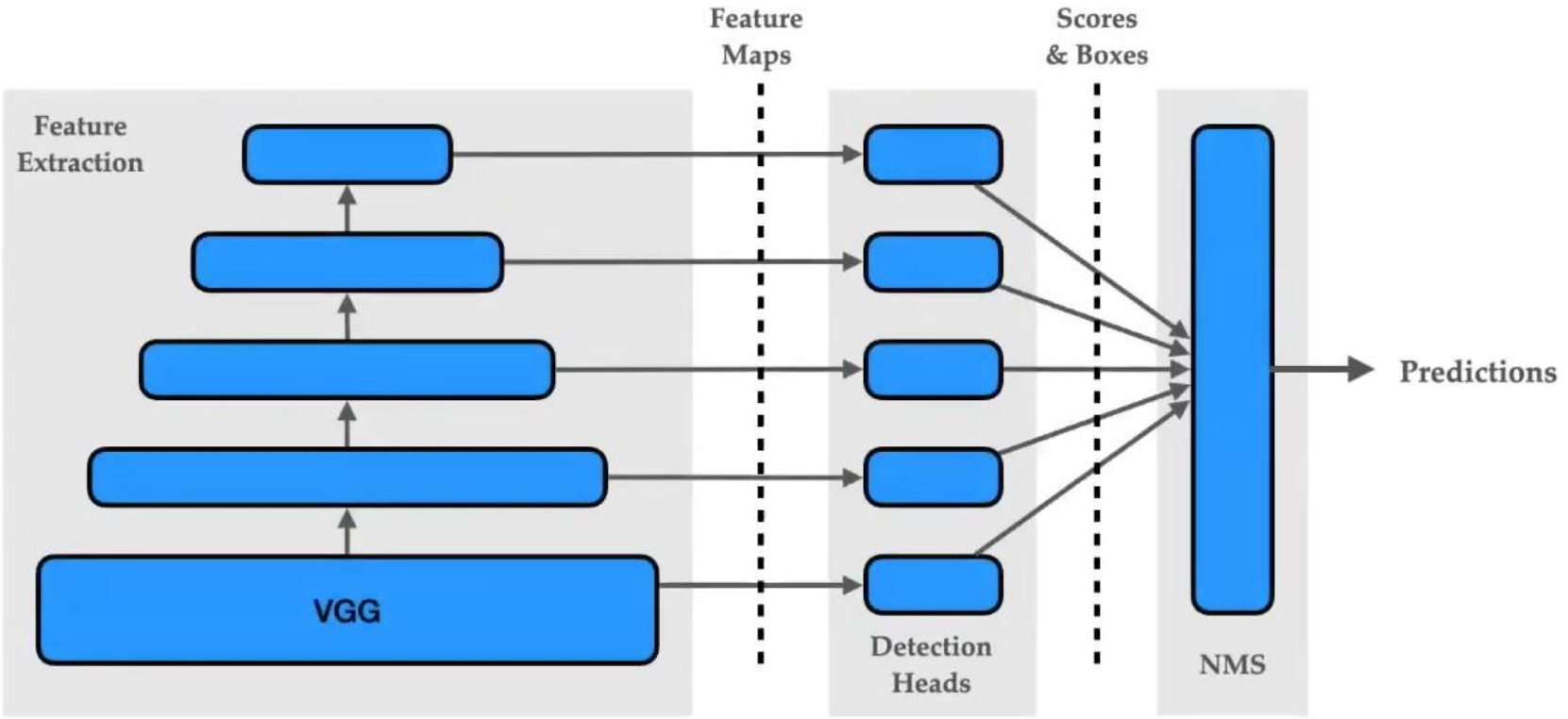




$$b_x = d_x + d_w\beta_x \qquad b_w = d_w e^{\beta_w}$$
$$b_y = d_y + d_h\beta_y \qquad b_h = d_h e^{\beta_h}$$

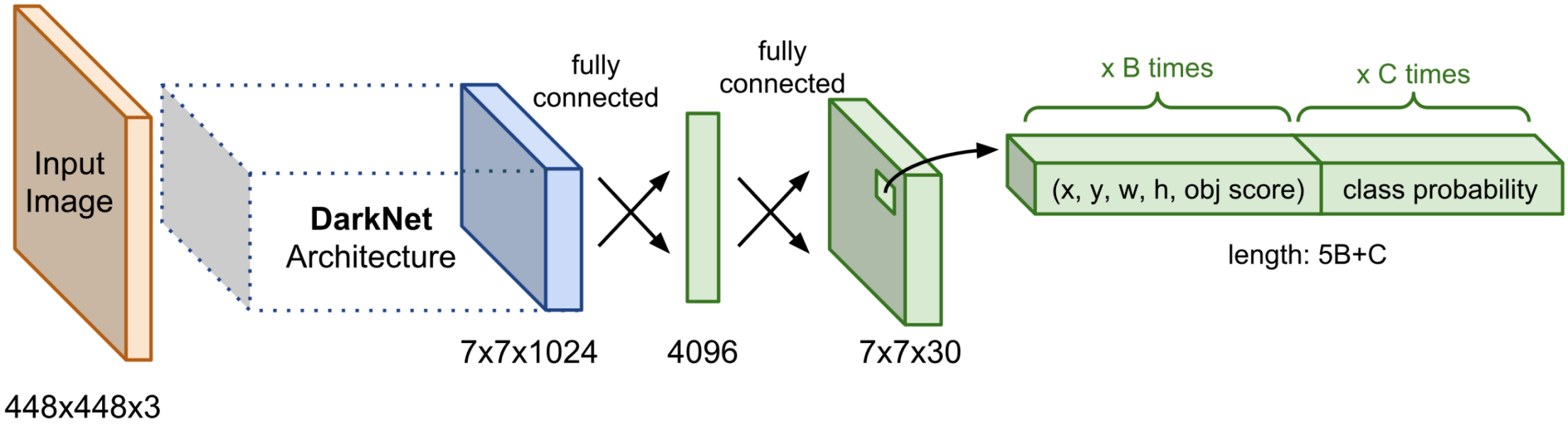
[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD) [4]



[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

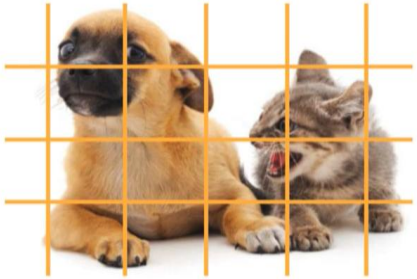
You Only Look Once (YOLO) [5]



[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640

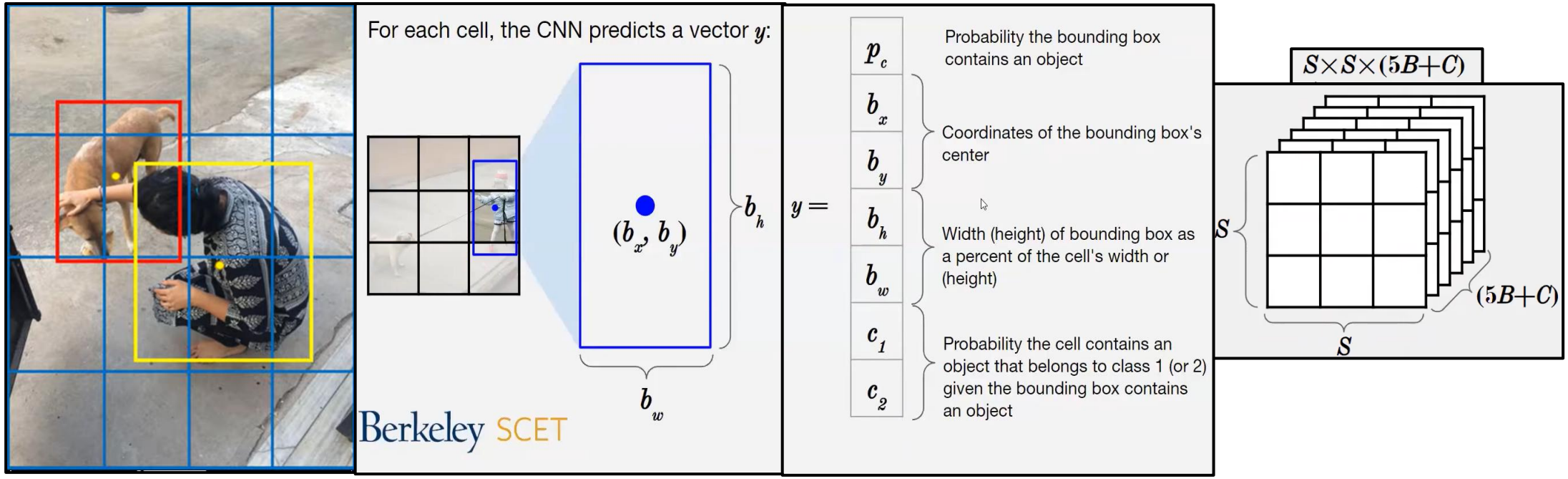
You Only Look Once (YOLO) [5]

- Anchor boxes are highly overlapped in SSD
- YOLO cuts the input image uniformly into S x S anchor boxes
- Each anchor box predicts B bounding boxes
- V2 and V3 add more improvements



[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640

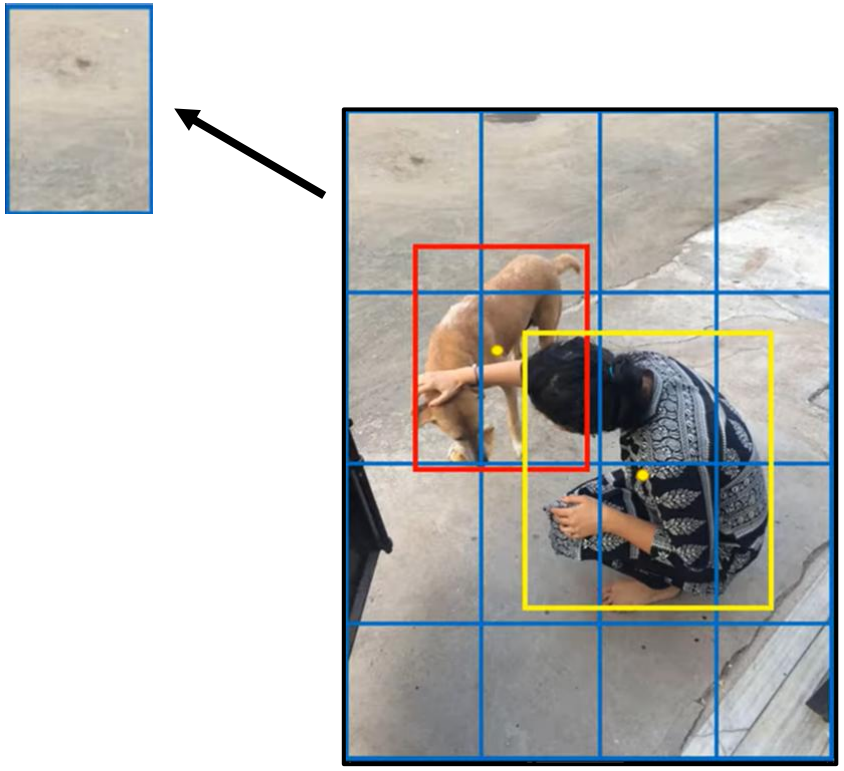
You Only Look Once (YOLO) [5]



[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640

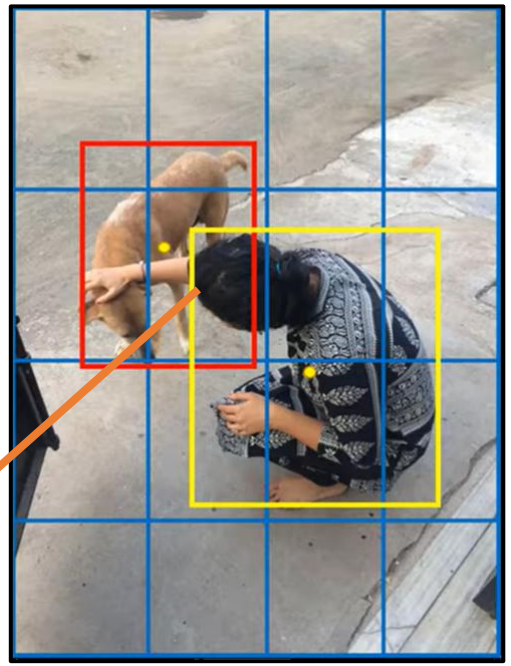
You Only Look Once (YOLO) - Format

$$\begin{pmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{pmatrix} \begin{pmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$$



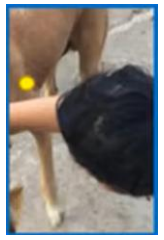
You Only Look Once (YOLO) - Format

$$\begin{pmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{pmatrix} \quad \begin{pmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$$



$$\begin{pmatrix} 1 \\ 0.05 \\ 0.3 \\ 2 \\ 1.3 \\ 1 \\ 0 \end{pmatrix}$$

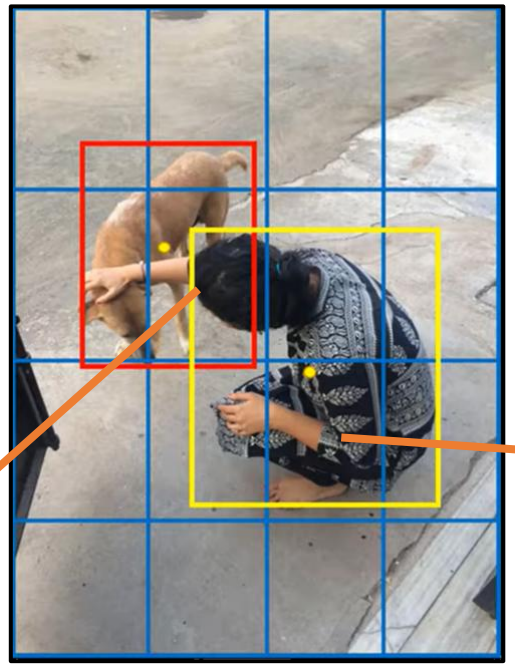
(0,0)



(1,1)

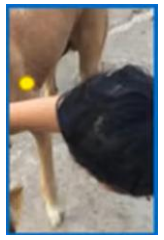
You Only Look Once (YOLO) - Format

$$\begin{pmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{pmatrix} \quad \begin{pmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$$

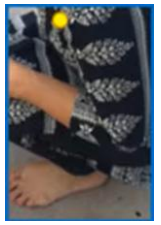


$$\begin{pmatrix} 1 \\ 0.05 \\ 0.3 \\ 2 \\ 1.3 \\ 1 \\ 0 \end{pmatrix}$$

(0,0)

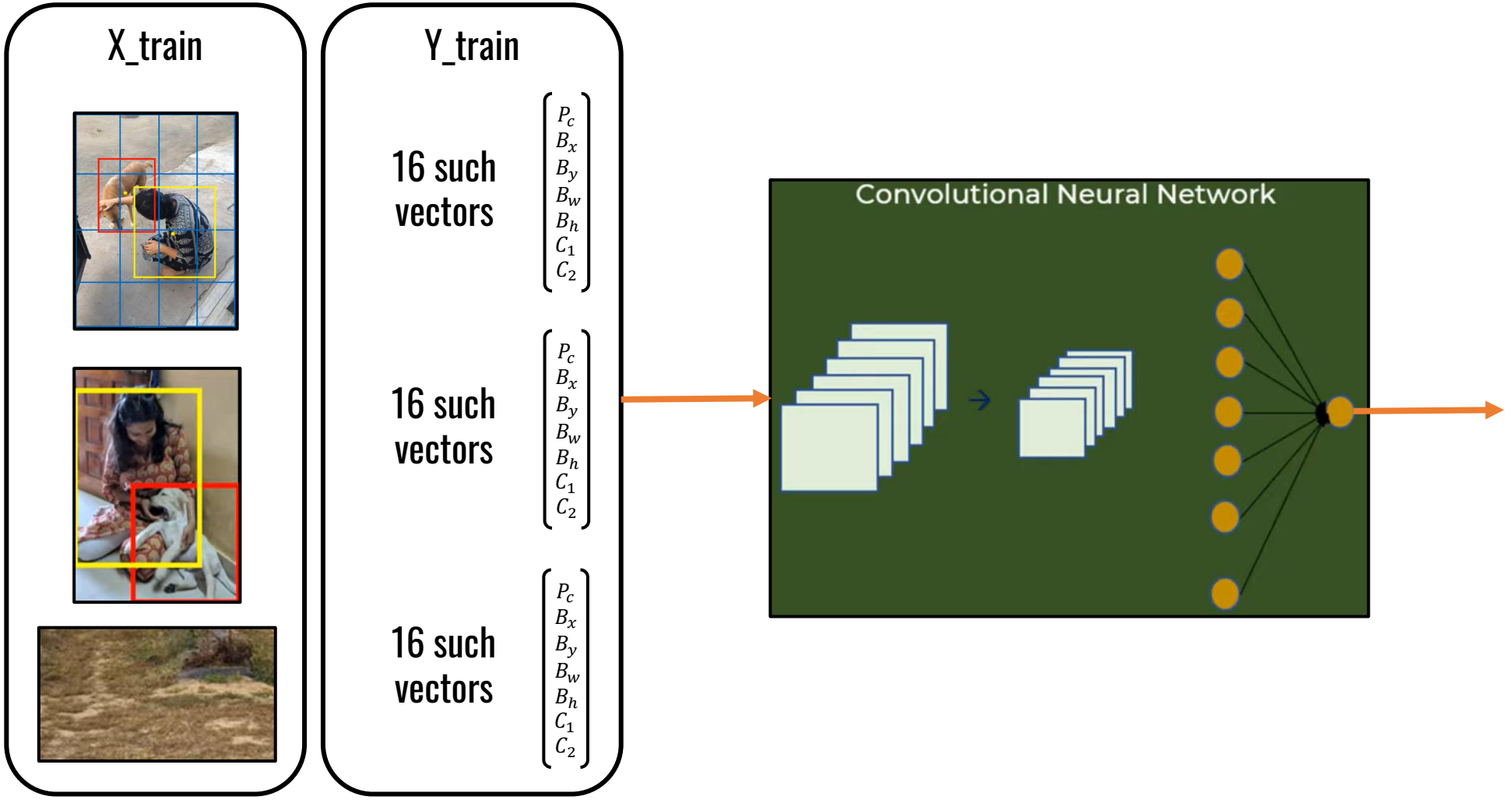


(1,1)



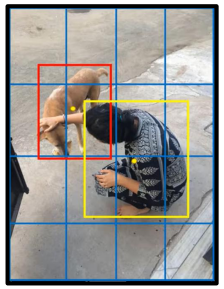
$$\begin{pmatrix} 1 \\ 0.32 \\ 0.02 \\ 3 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

You Only Look Once (YOLO) - Training



You Only Look Once (YOLO) - Prediction

X_train



Y_train

16 such vectors

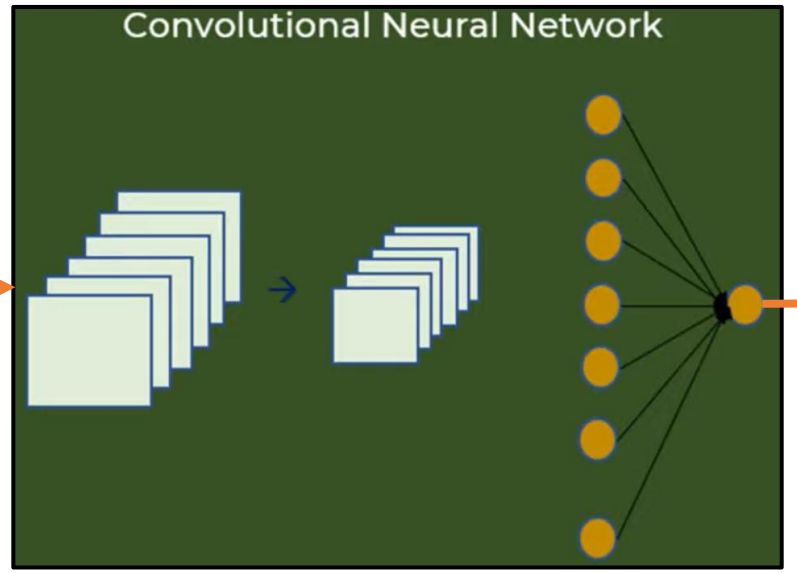
$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$

16 such vectors

$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$

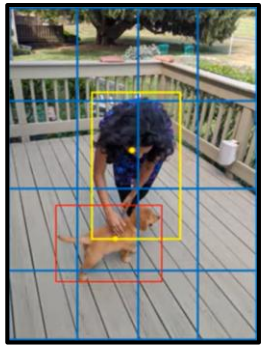
16 such vectors

$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$



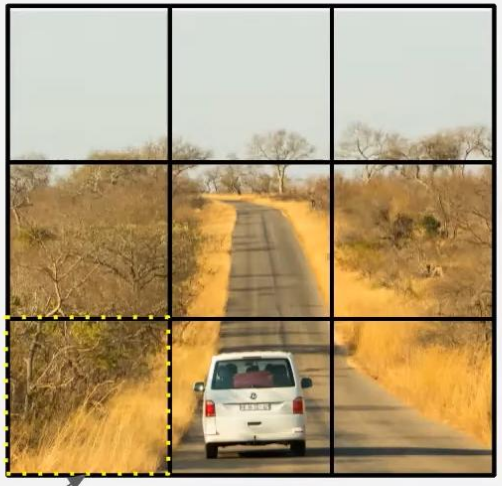
16 such vectors

$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$



You Only Look Once (YOLO)

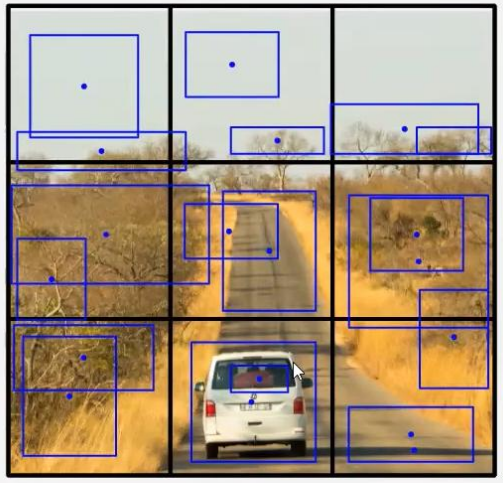
1. **Divide the image into cells with an S x S grid.**



$S = 3$

Cell

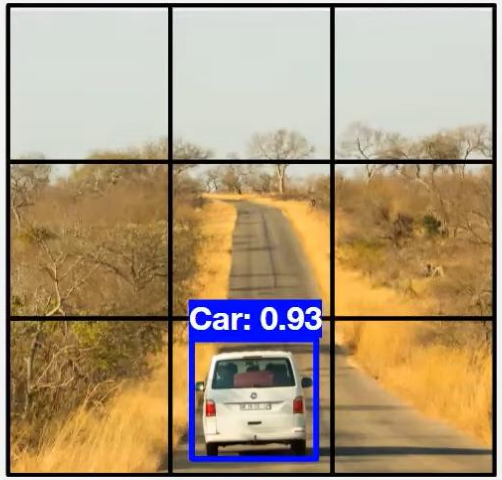
2. **Each cell predicts B bounding boxes.**



$B = 2$

A cell is responsible for detecting an object if the object's bounding box falls within the cell. (Notice that each cell has 2 blue dots.)

3. **Return bounding boxes above confidence threshold.**



All other bounding boxes have a confidence probability less than the threshold (say 0.90) so they are suppressed.

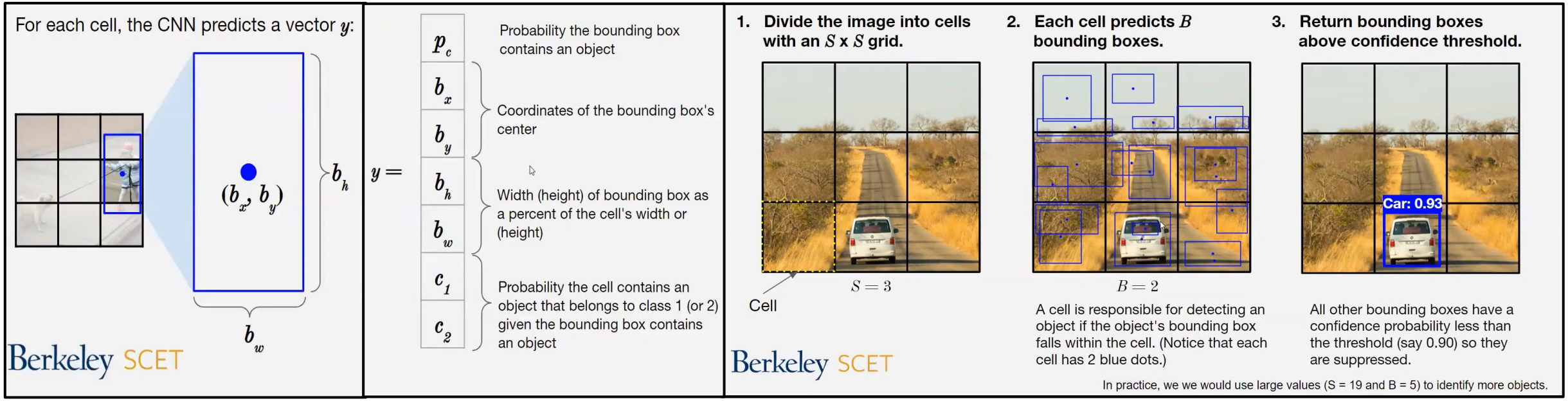
In practice, we we would use large values (S = 19 and B = 5) to identify more objects.

Berkeley SCET

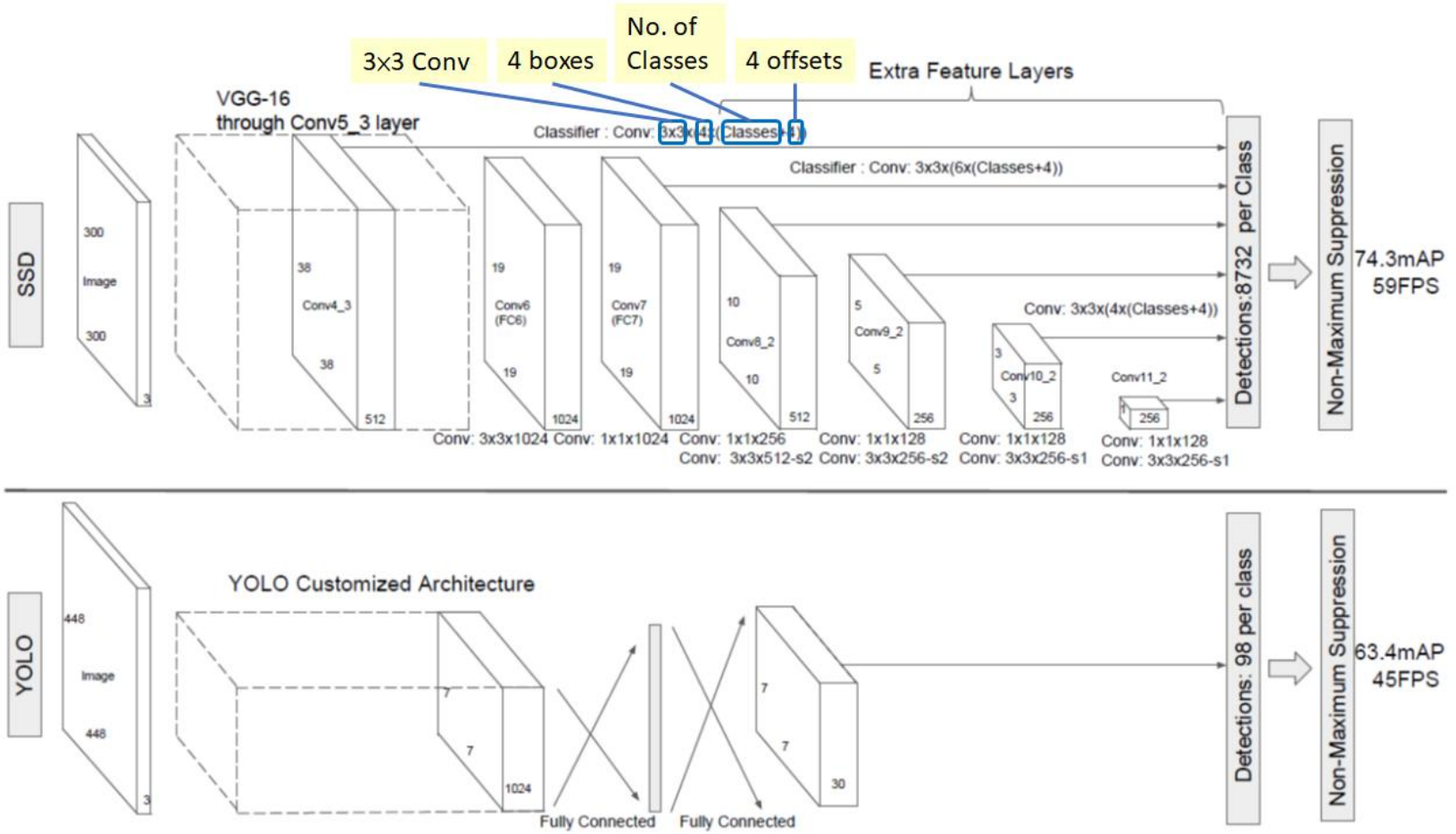
You Only Look Once (YOLO)

Please check the following links for a detailed explanation of YOLO versions:

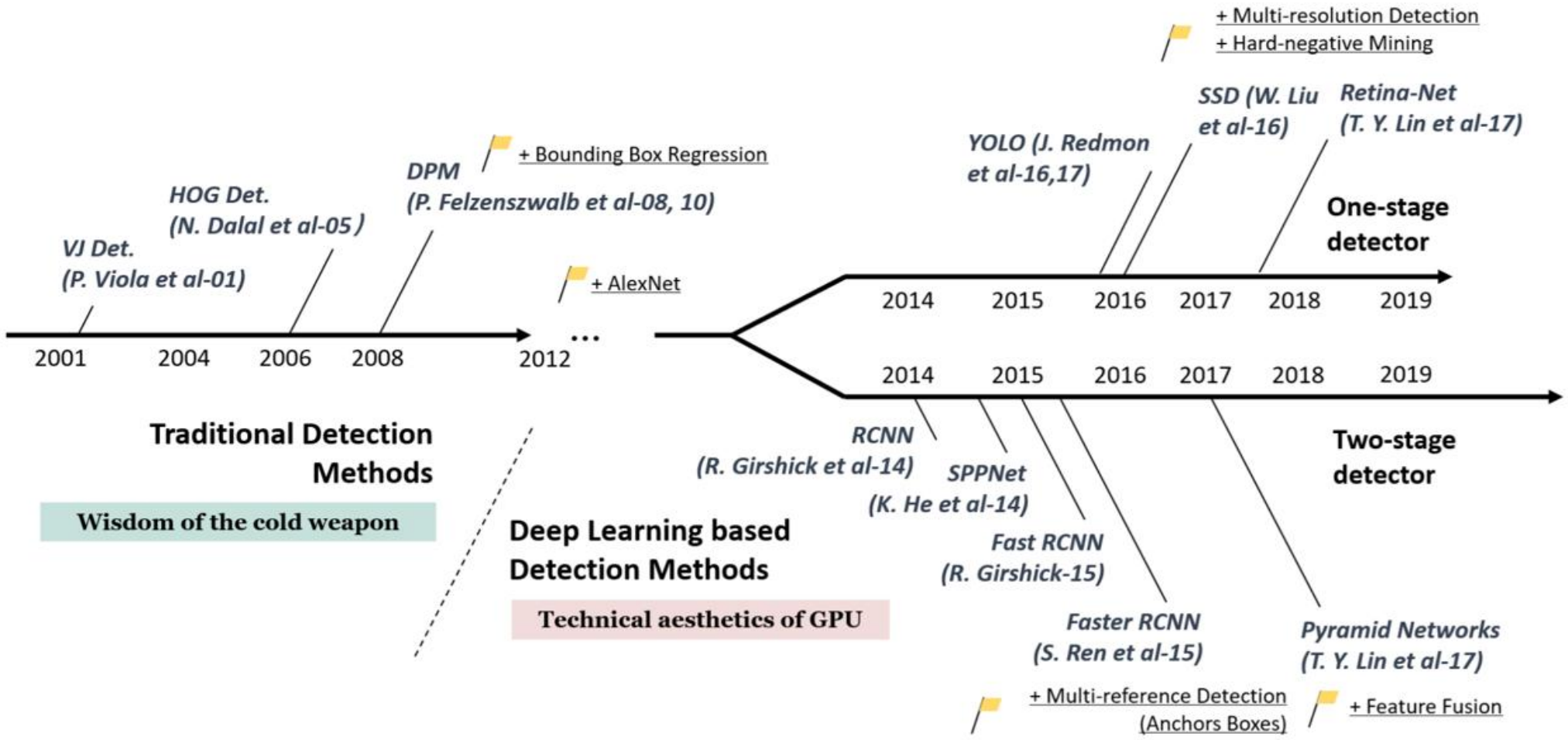
- https://www.v7labs.com/blog/yolo-object-detection
- https://www.datacamp.com/blog/yolo-object-detection-explained
- Video comparison



SSD vs YOLO

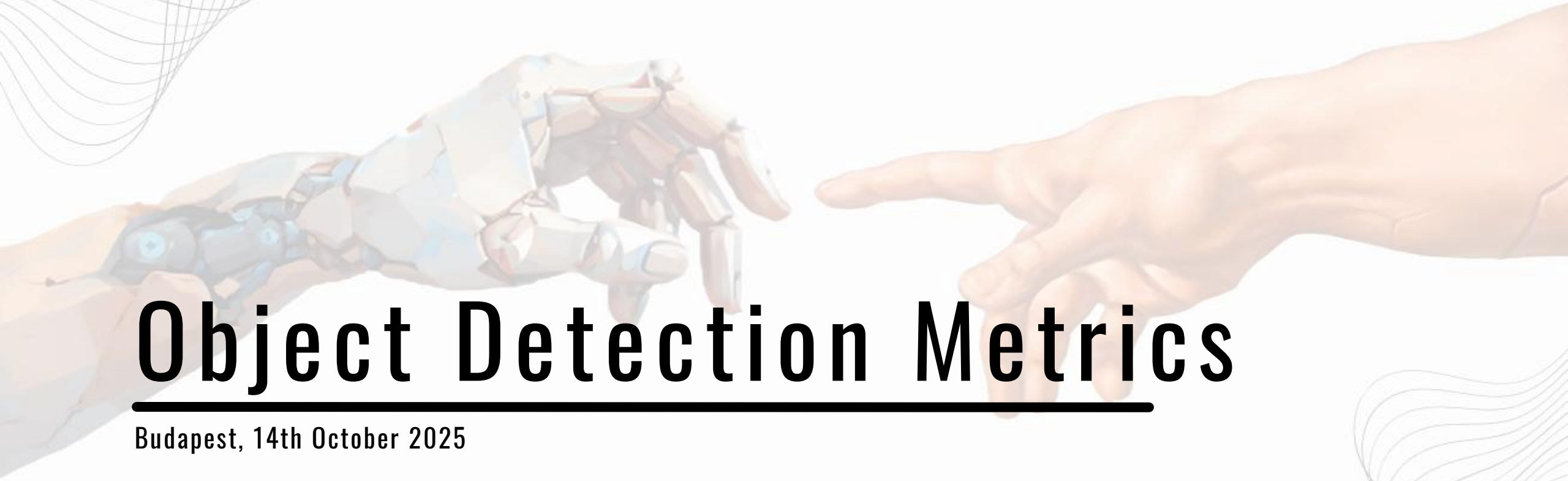

3x3 Conv
4 boxes
No. of Classes
4 offsets
Extra Feature Layers
VGG-16
through Conv5_3 layer
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
SSD
300
Image
300
3
38
Conv4_3
38
512
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
Conv11_2
1
256
Conv: 3x3x(4x(Classes+4))
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
YOLO
448
Image
448
3
YOLO Customized Architecture
7
7
1024
Fully Connected
Fully Connected
7
7
30
Detections: 98 per class
Non-Maximum Suppression
63.4mAP
45FPS

Other Methods



Deep Network Development

Lecture 8.



Object Detection Metrics

Budapest, 14th October 2025

1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

Comparing predictions: Accuracy

In case of binary classification, the binary accuracy is:

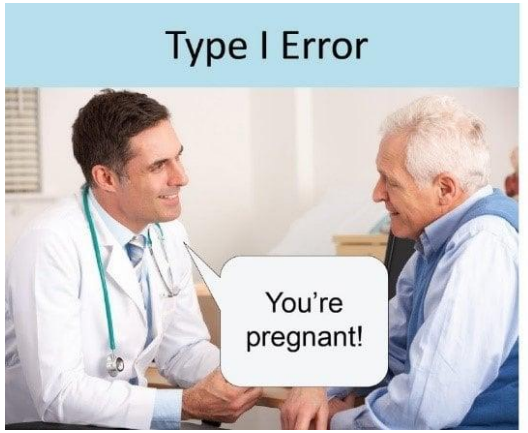
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

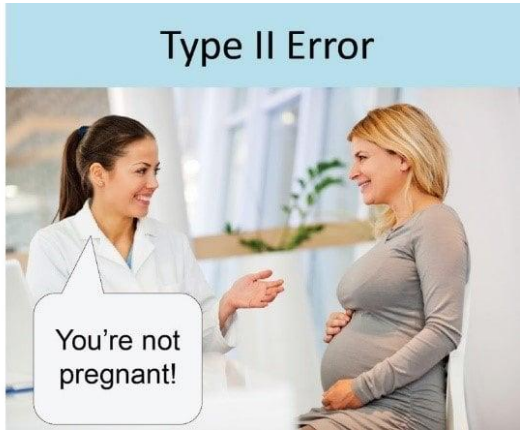
- TP = True positive,
- FP = False Positive,
- TN = True negative
- FN = False negative

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

False positive



False negative



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Comparing predictions: Accuracy

In case of multi class classification, the accuracy is:

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{all classifications}}$$

- This is usually expressed as a percentage, i.e. 90%
- To get a better insight we usually visualize it in a **confusion matrix**

Confusion matrix

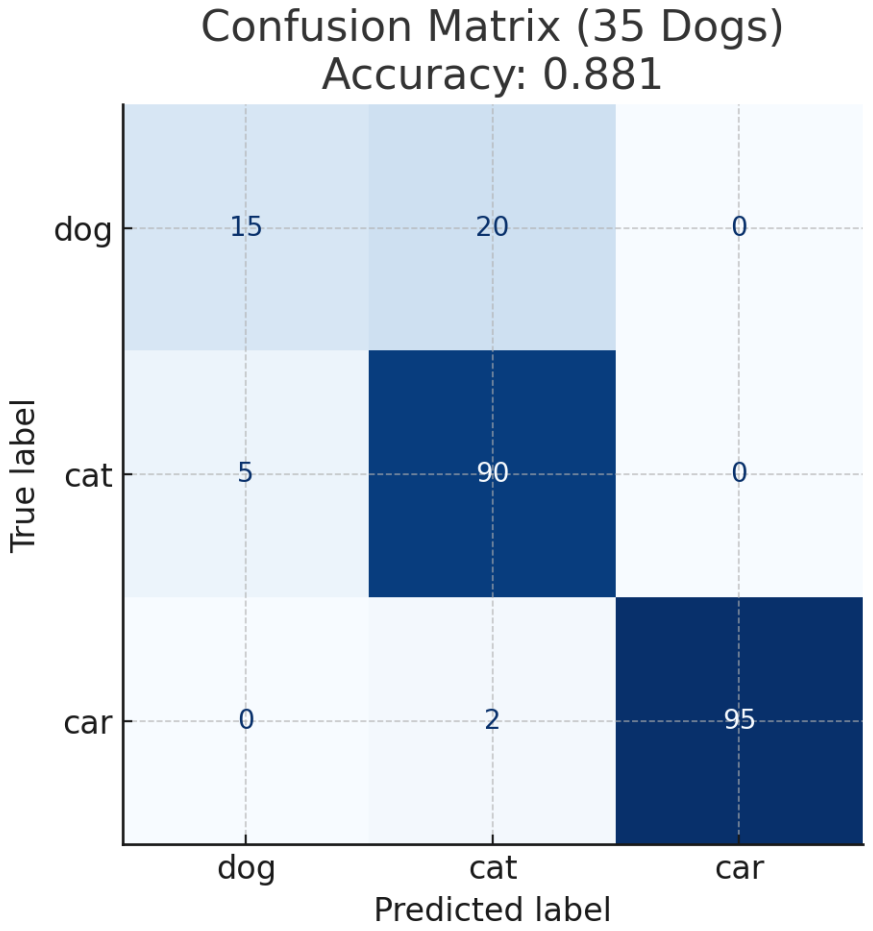
V7

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Comparing predictions: Accuracy

Would you consider this model good?

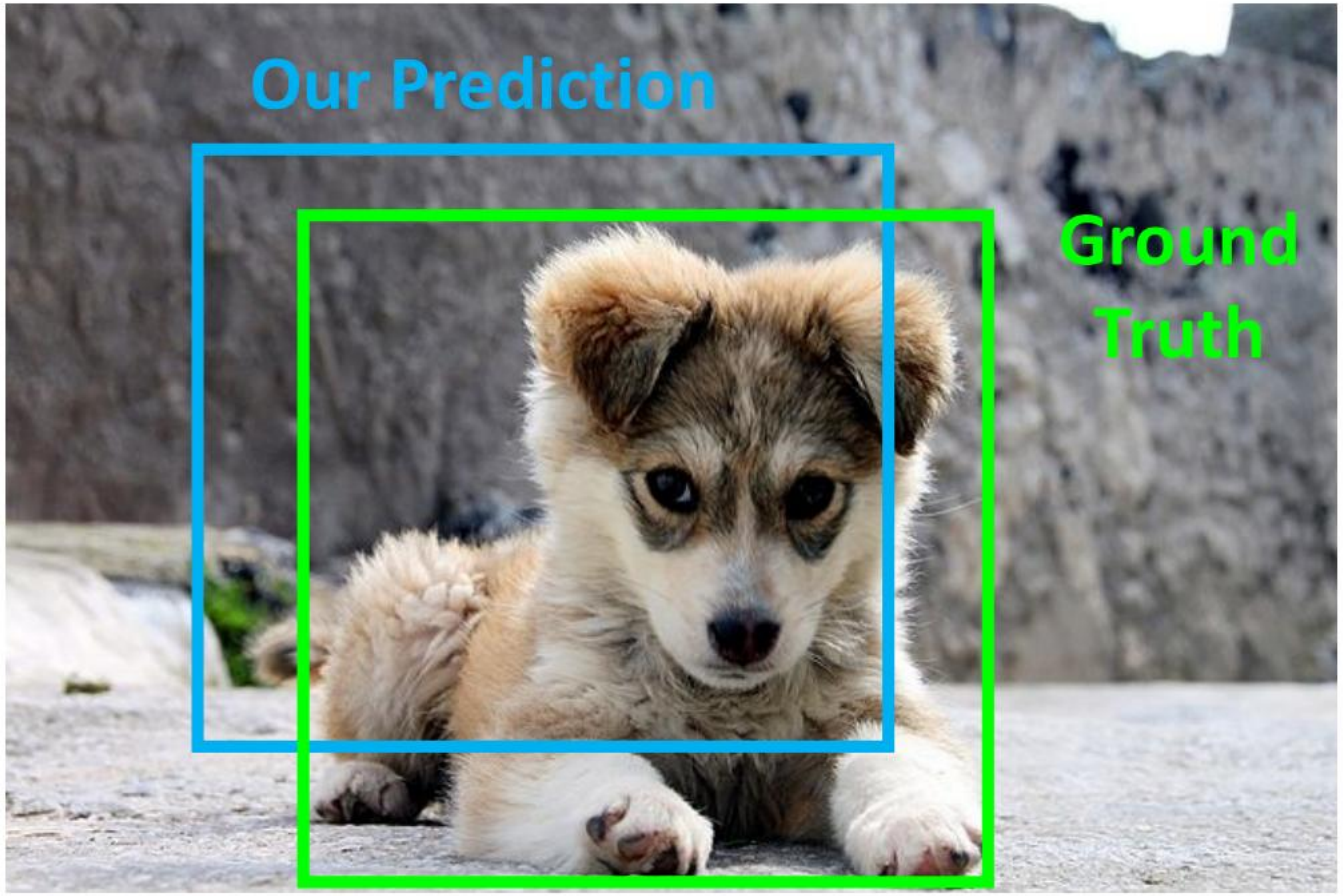
Accuracy: 88%



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Comparing boxes: Intersection over Union (IoU)

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



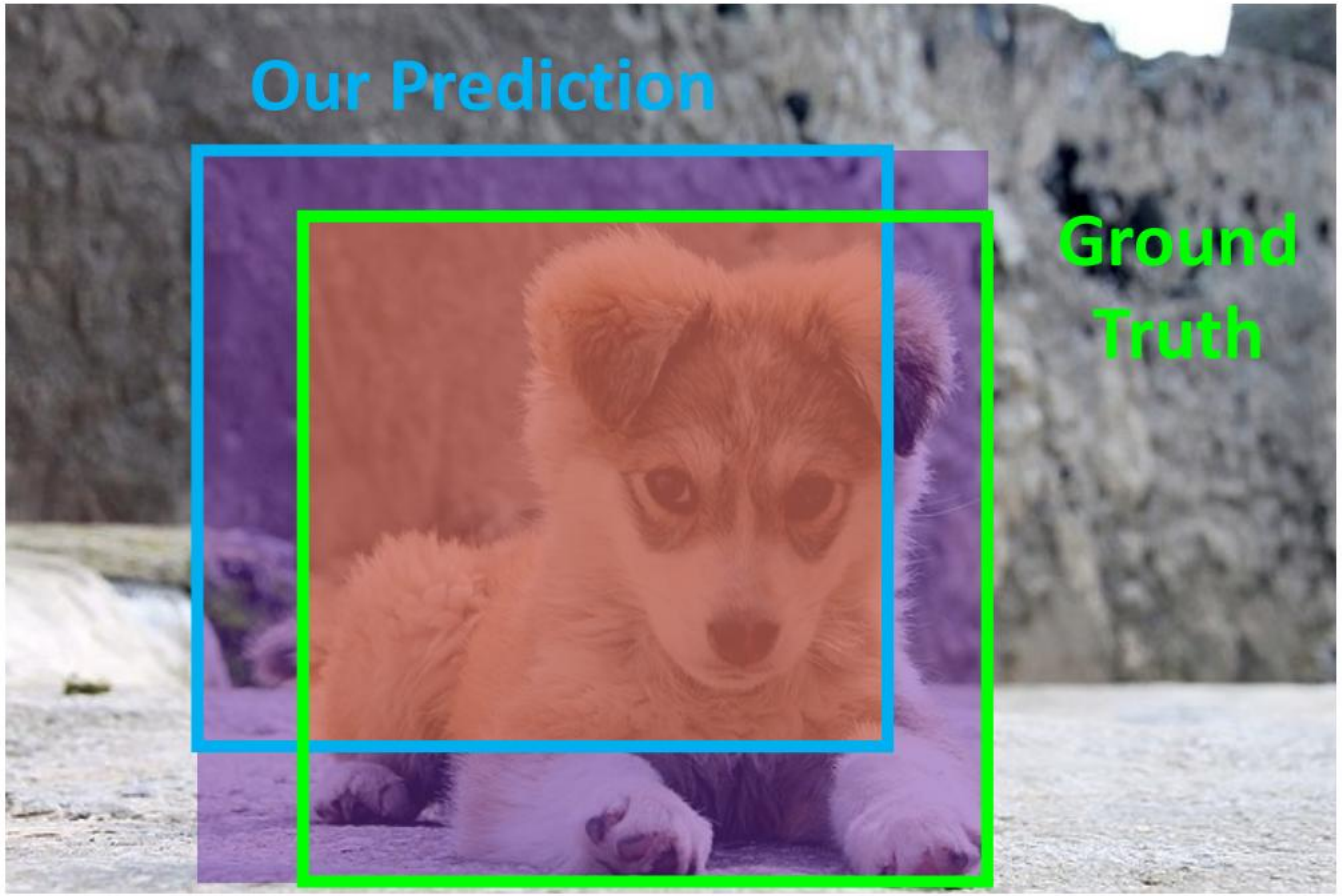
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Comparing boxes: Intersection over Union (IoU)

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

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

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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

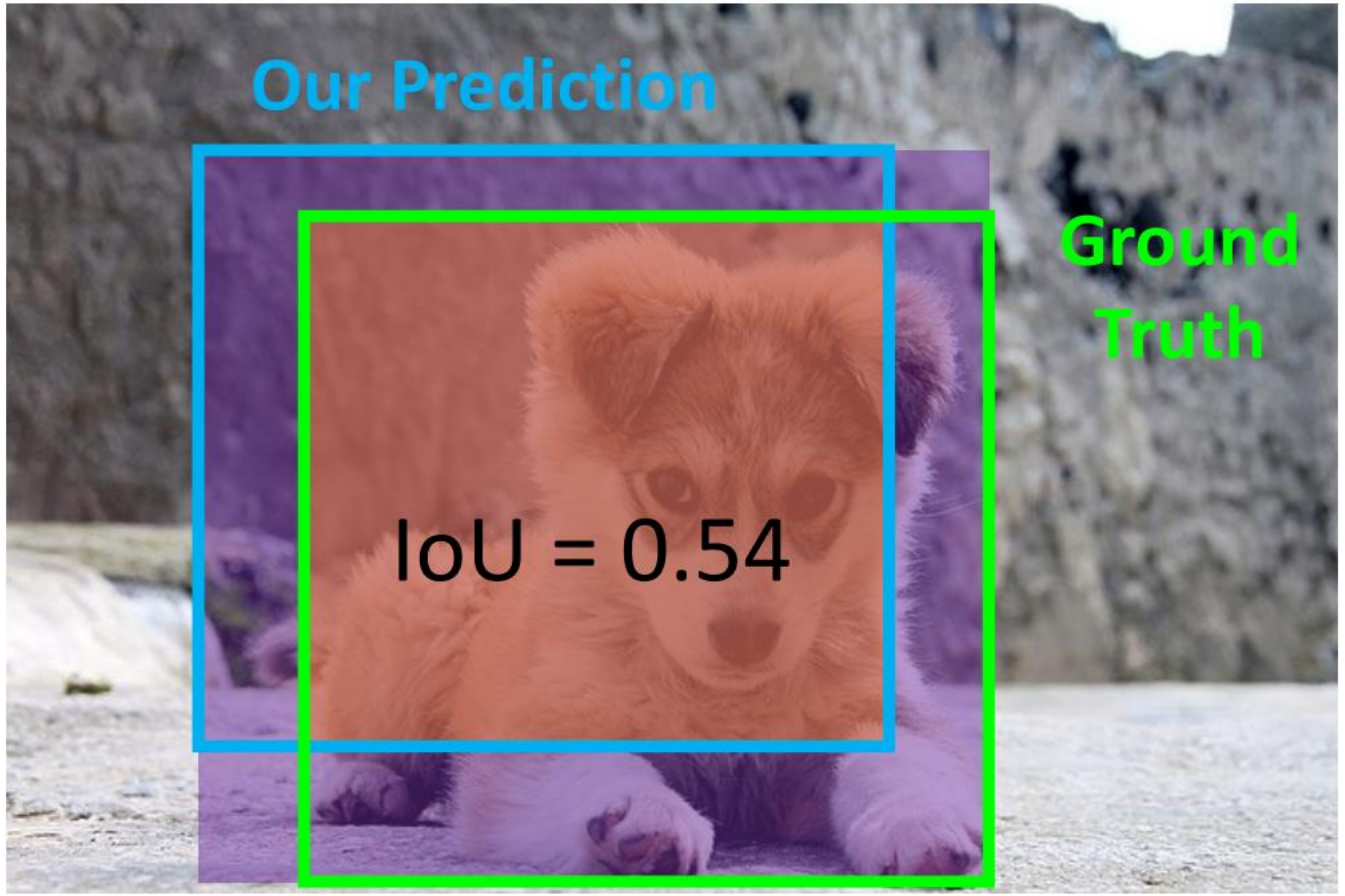
Comparing boxes: Intersection over Union (IoU)

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

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

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

IoU > 0.5 is “decent”



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Comparing boxes: Intersection over Union (IoU)

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

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

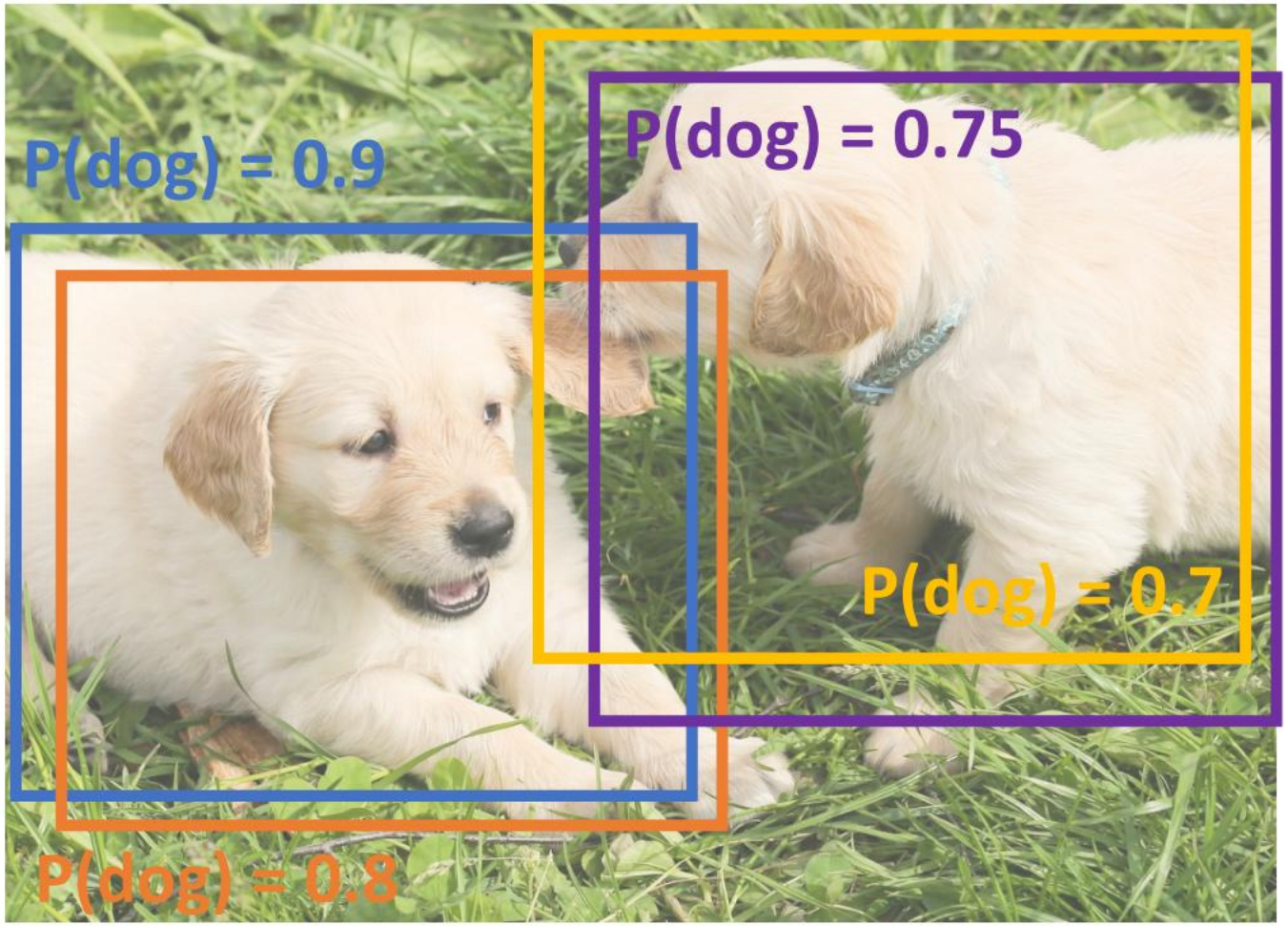
$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

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

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Overlapping boxes

Problem: Object detectors often output many overlapping detections:



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

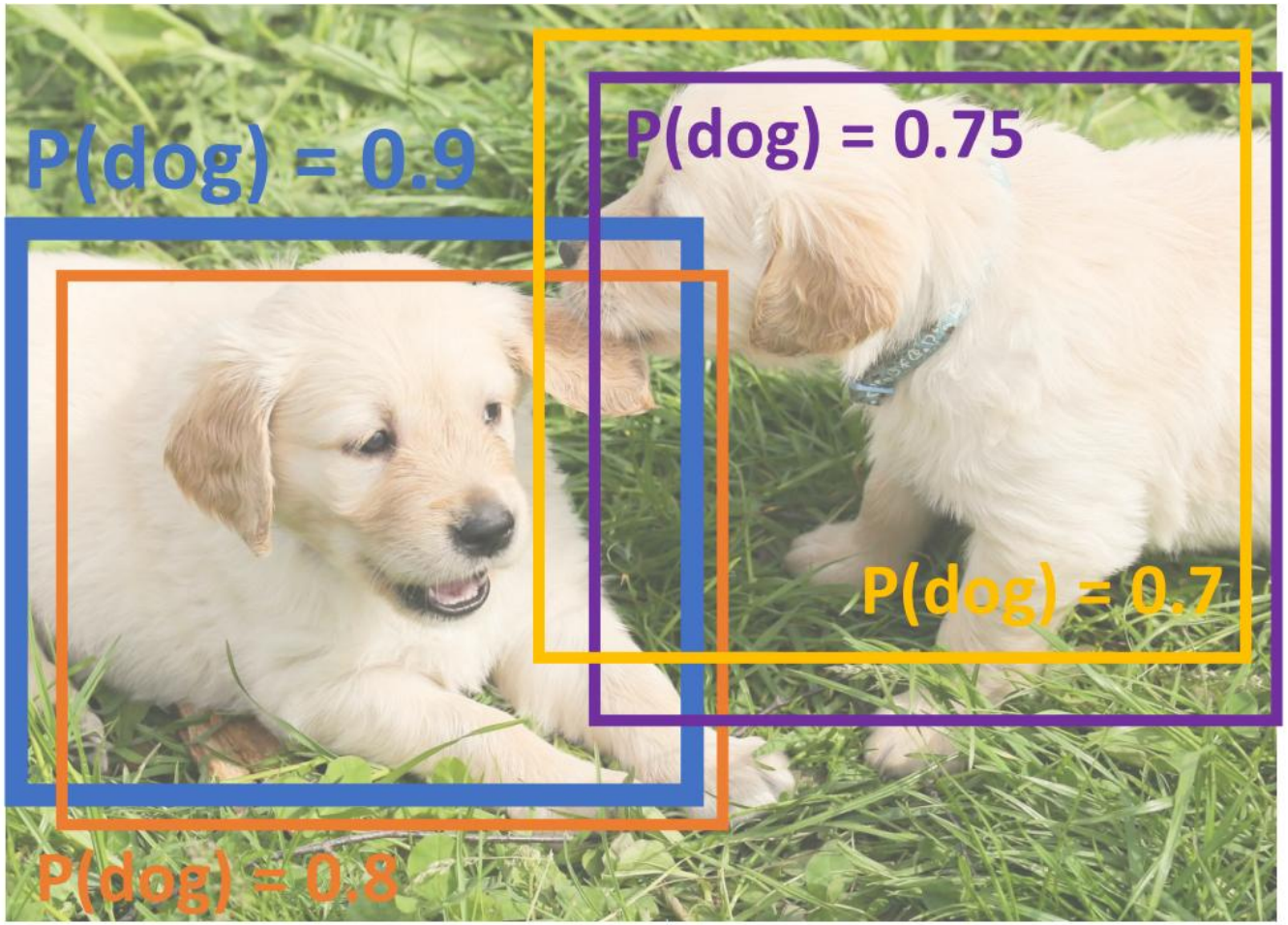
Overlapping boxes

Problem: Object detectors often output many overlapping detections:

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

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

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



Puppy image is CC0 Public Domain

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

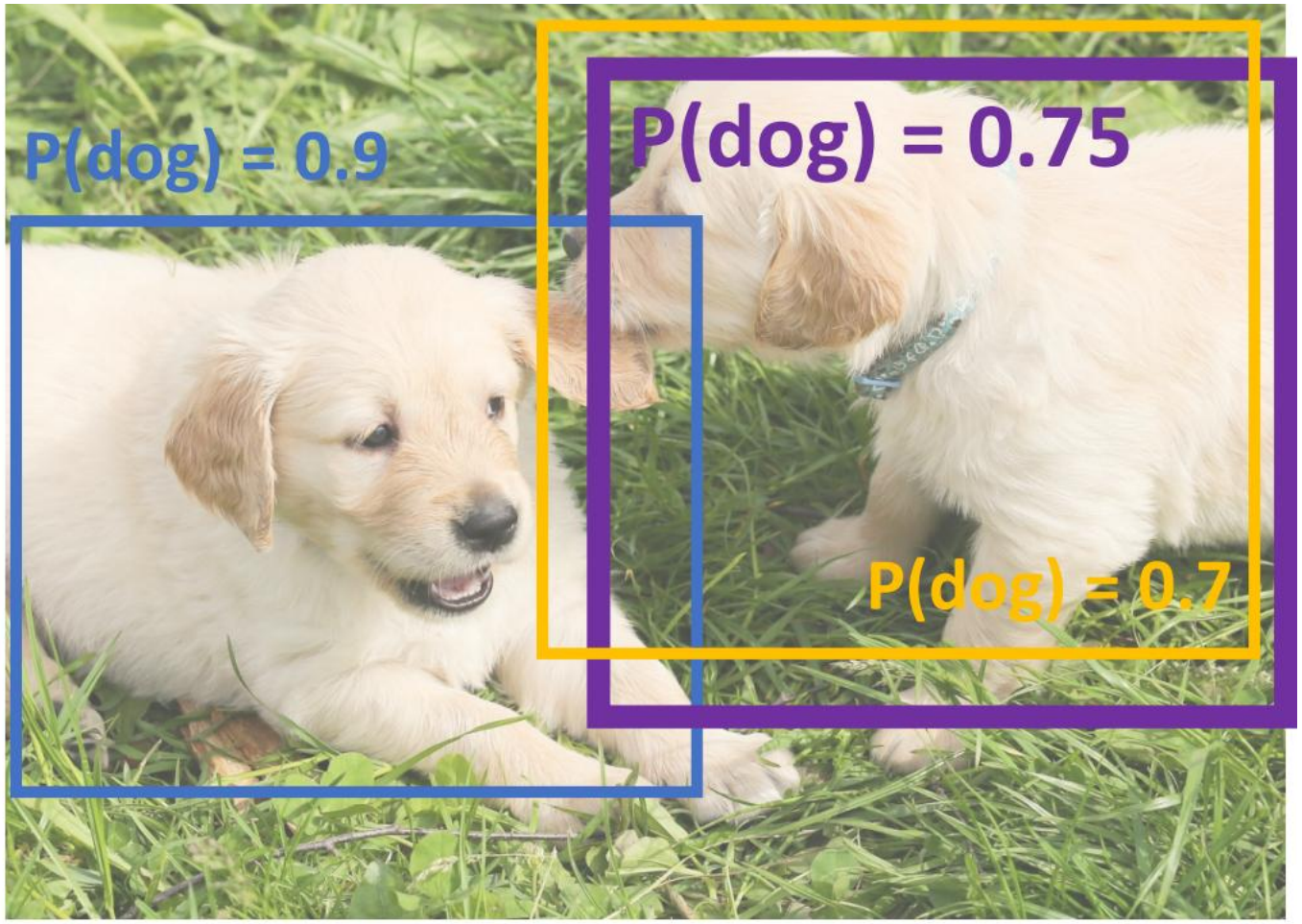
Overlapping boxes

Problem: Object detectors often output many overlapping detections:

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

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

IoU(,) = **0.74**



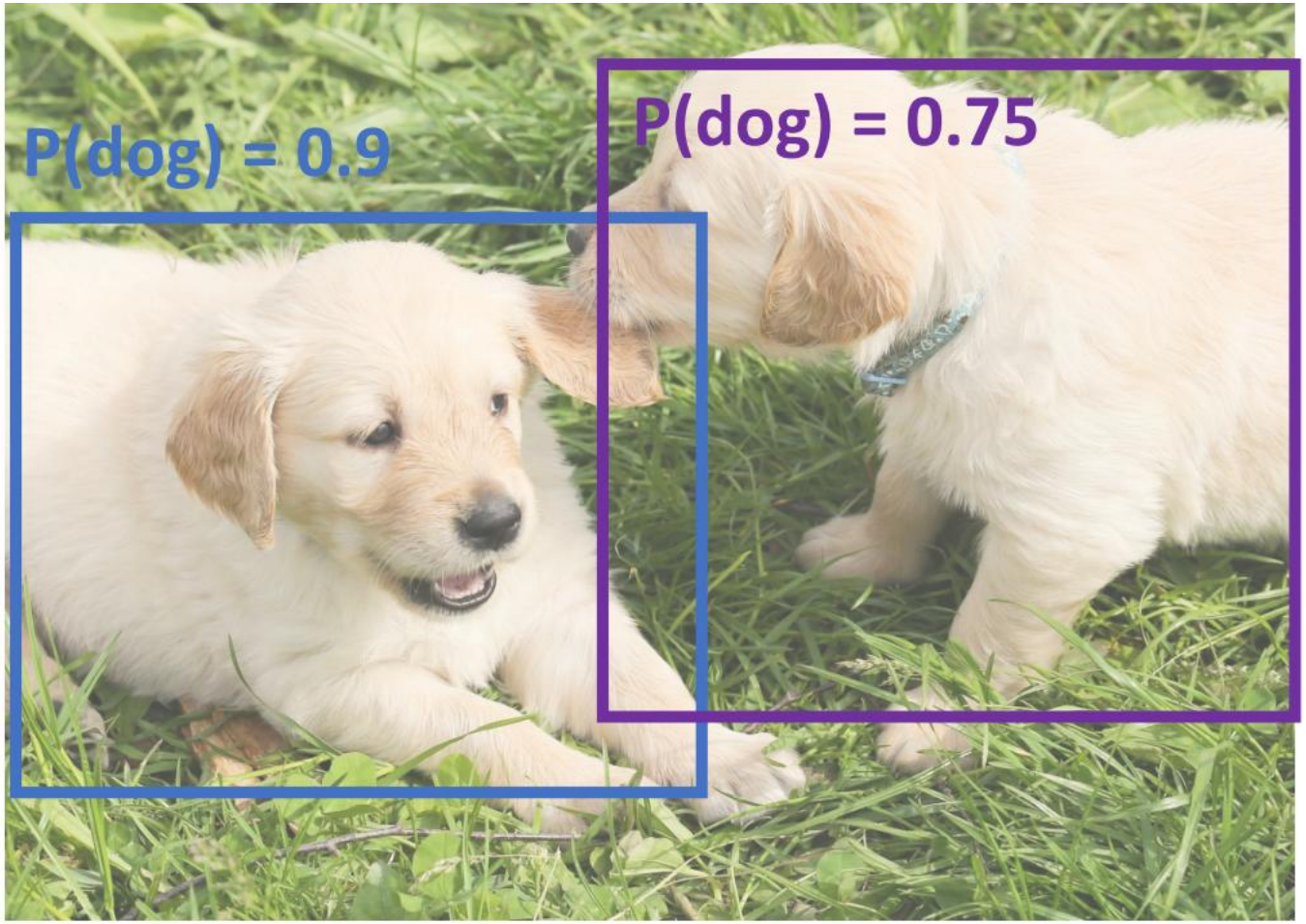
Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Overlapping boxes

Problem: Object detectors often output many overlapping detections:

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

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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

NMS Limitations

Problem: Object detectors often output many overlapping detections:

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

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

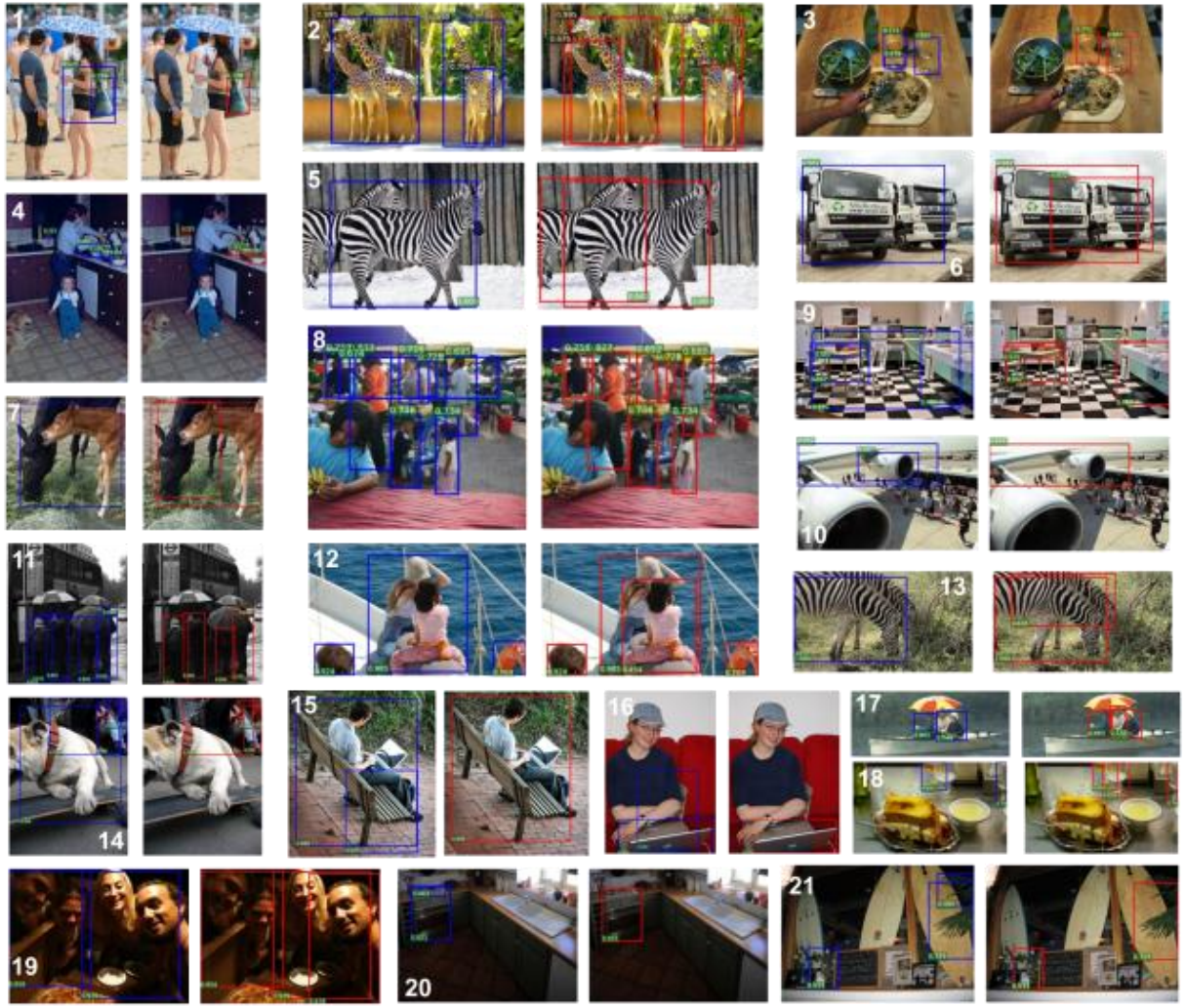
Problem: NMS may eliminate ”good” boxes when objects are highly overlapping... no good solution =(

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

NMS Limitations

There are a few other methods that try to improve detections:

- **Soft-NMS:** Traditional NMS is a binary operation that discards all but the highest-scoring bounding box. Soft-NMS, on the other hand, assigns lower scores to overlapping boxes rather than completely removing them. This results in smoother score degradation for close-by objects, reducing the chance of removing valid detections.

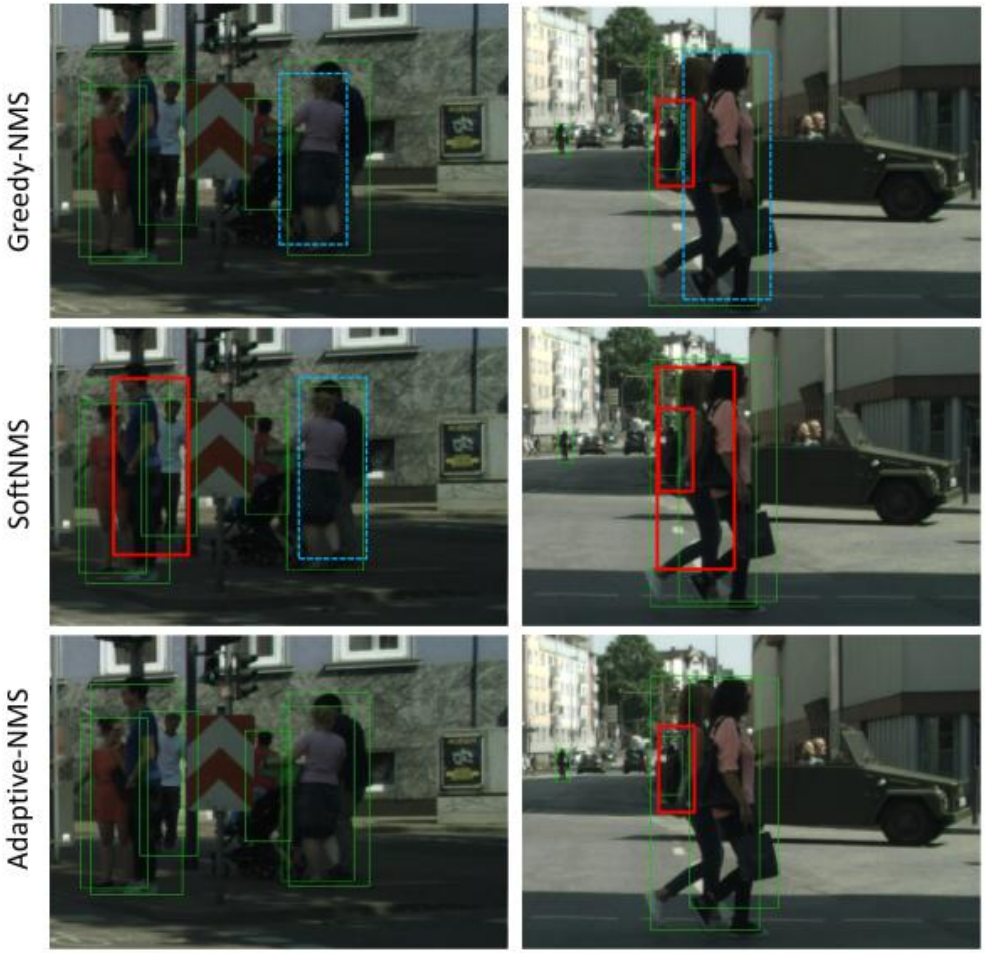


Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

NMS Limitations

There are a few other methods that try to improve detections:

- **IoU Threshold Adaptation:** Instead of using a fixed IoU (Intersection over Union) threshold for NMS, you can dynamically adjust the threshold based on the object's characteristics. For instance, you might use a higher IoU threshold for large objects and a lower IoU threshold for smaller objects.

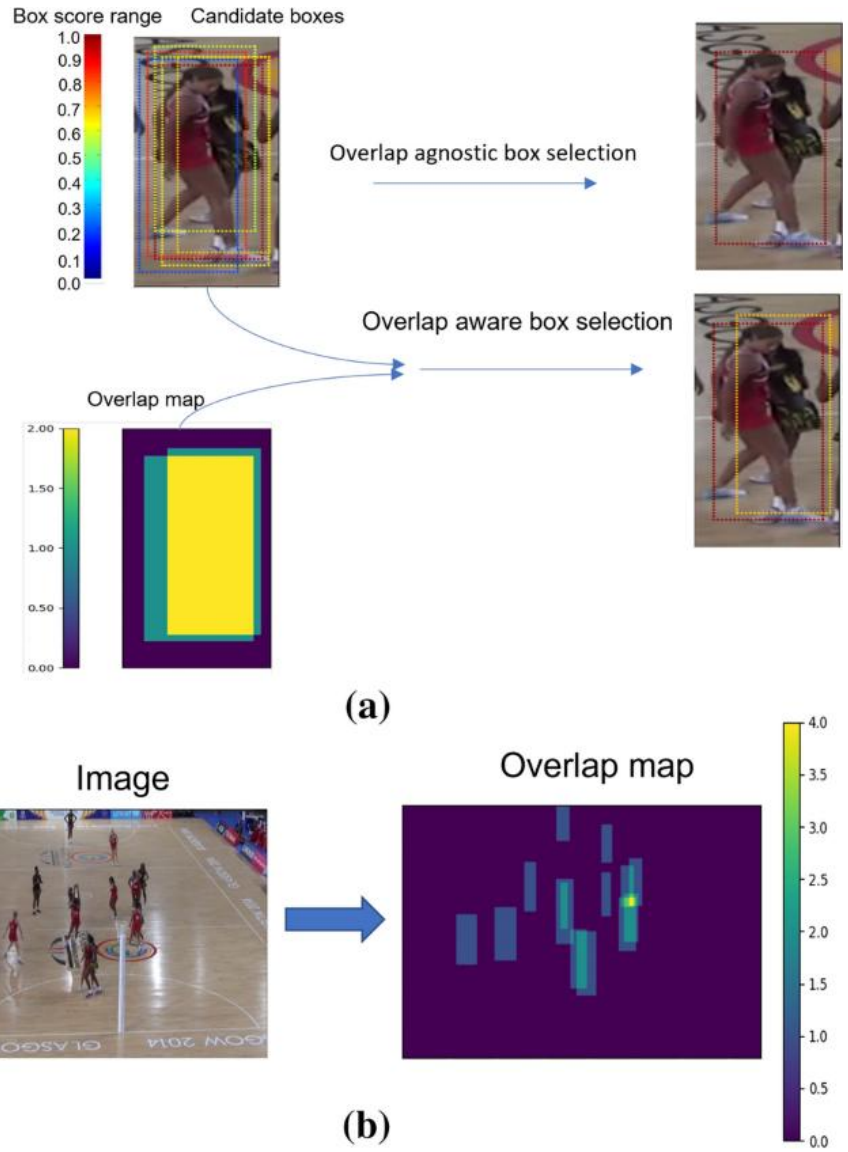


Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

NMS Limitations

More methods:

- Selection of object detections using overlap map predictions (https://link.springer.com/article/10.1007/s00521-022-07469-x)
- https://www.sciencedirect.com/science/article/pii/S2214914721001914
- https://arxiv.org/pdf/2207.00865.pdf
- And more...
- Open research field (make your contribution)



Evaluating Object Detectors: Mean Average Precision (mAP)

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

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Precision = $\frac{\text{True Positive}}{\text{Actual Results}}$ or $\frac{\text{True Positive}}{\text{True Positive + False Positive}}$

Recall = $\frac{\text{True Positive}}{\text{Predicted Results}}$ or $\frac{\text{True Positive}}{\text{True Positive + False Negative}}$

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Example with detecting Traffic lights
Positive: Traffic light
Negative: Background (non traffic light)

TRUE POSITIVE

FALSE POSITIVE

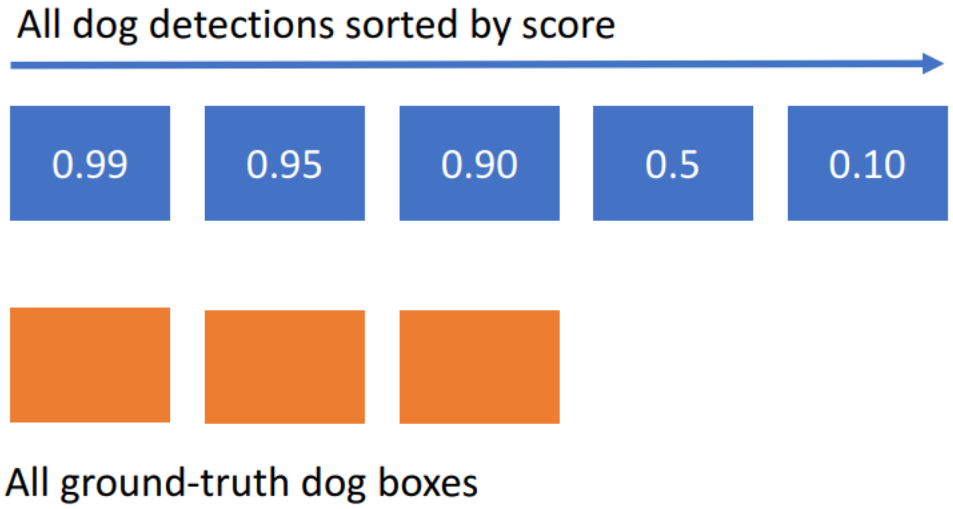
FALSE NEGATIVE

TRUE NEGATIVE

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

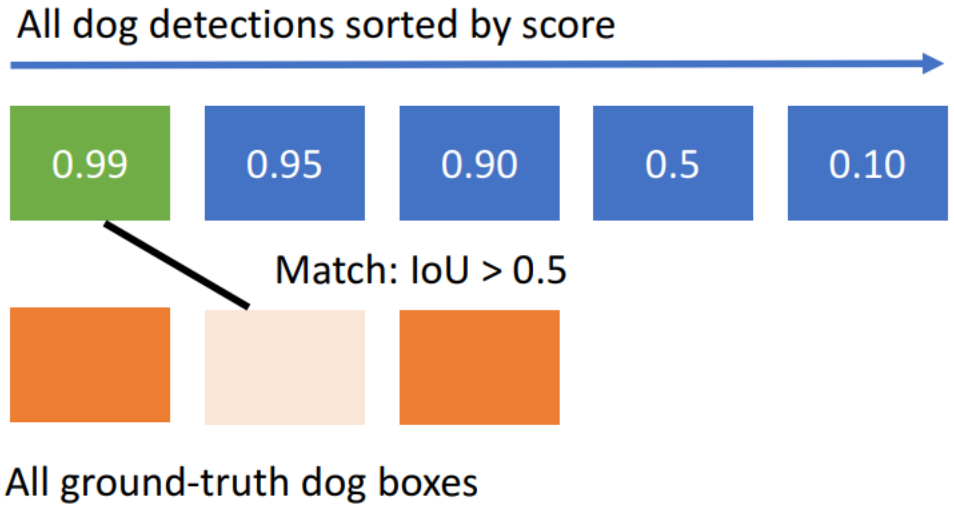
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

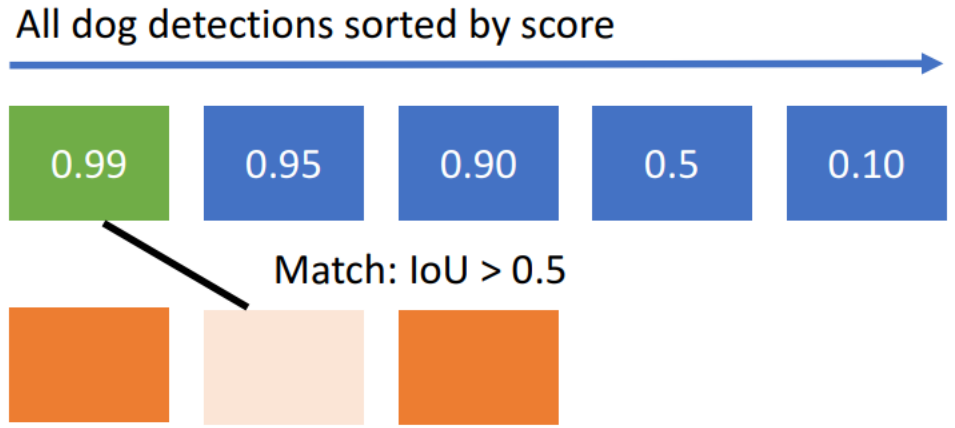
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

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



All ground-truth dog boxes

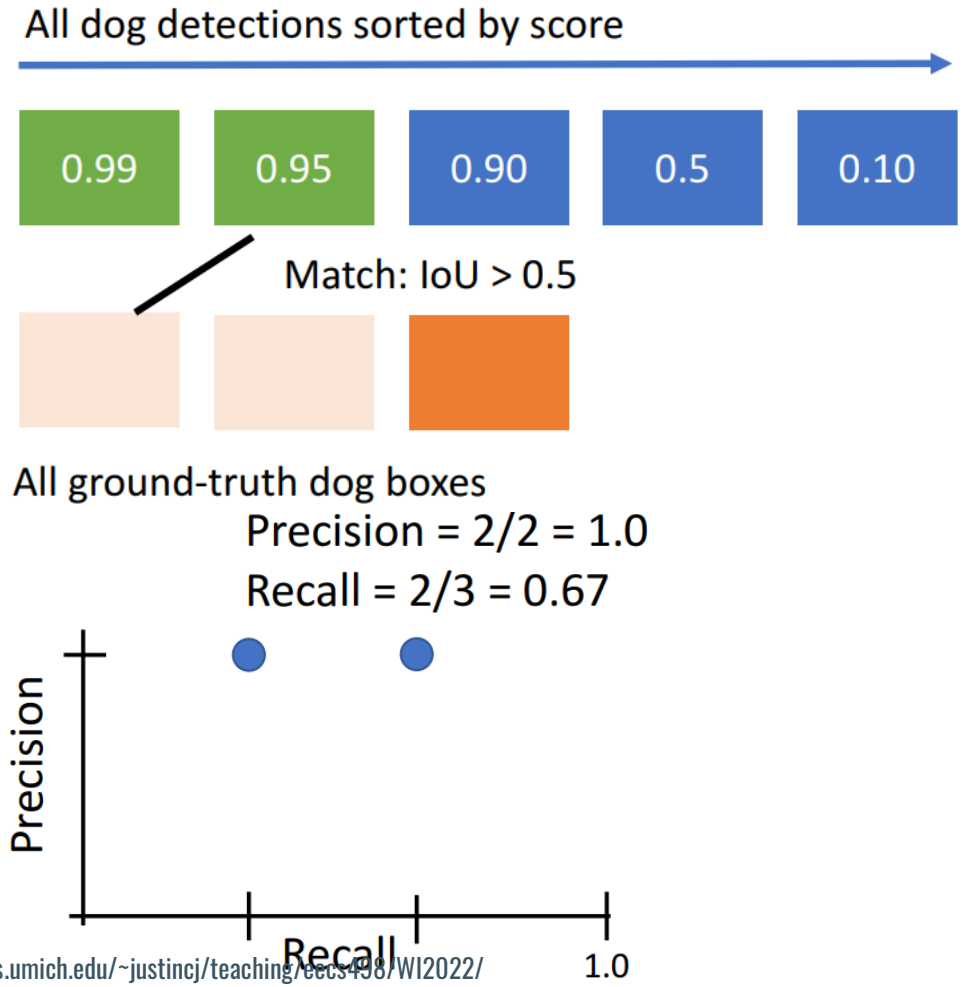
Precision = 1/1 = 1.0

Recall = 1/3 = 0.33

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

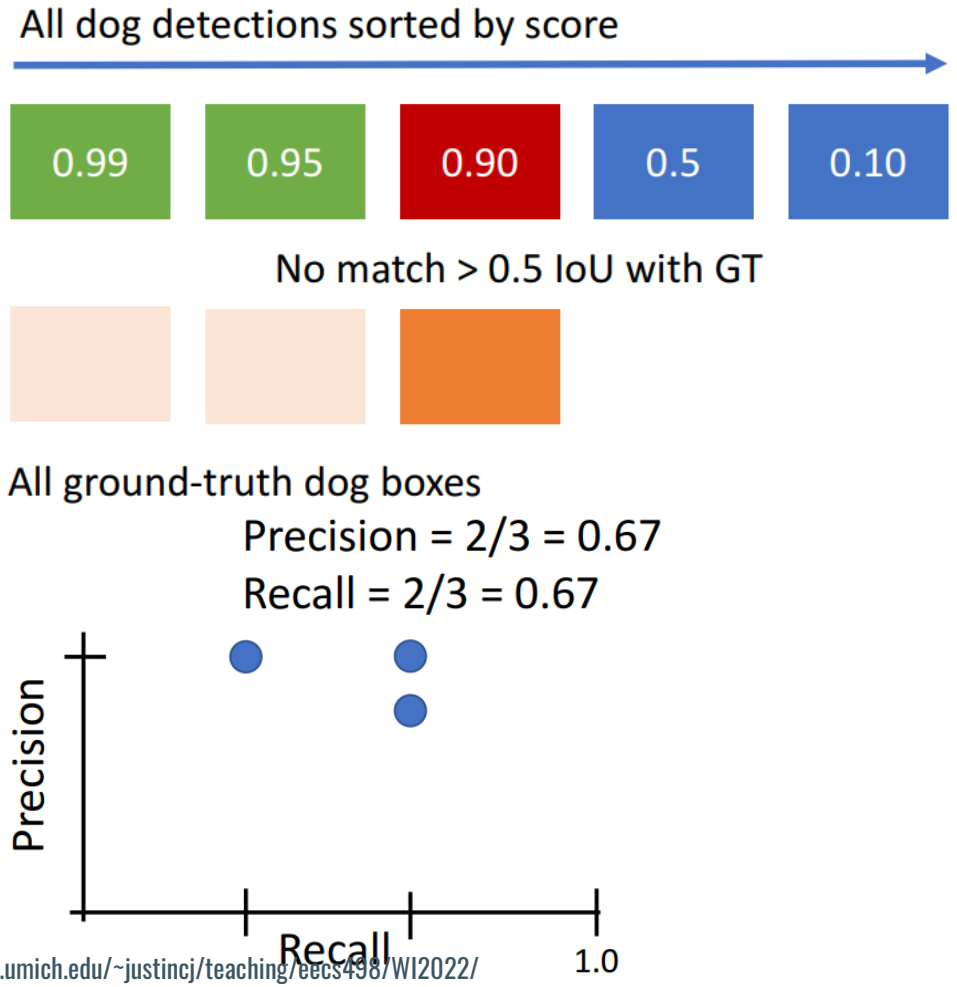
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

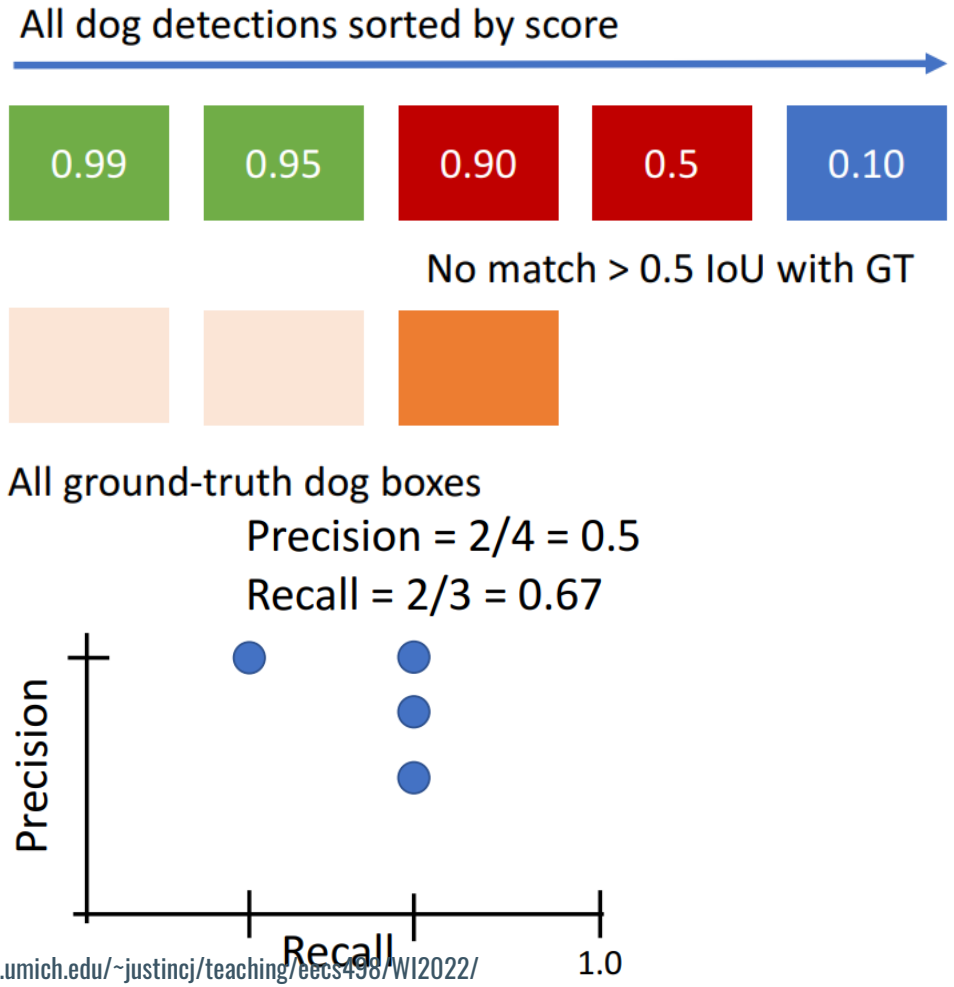
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

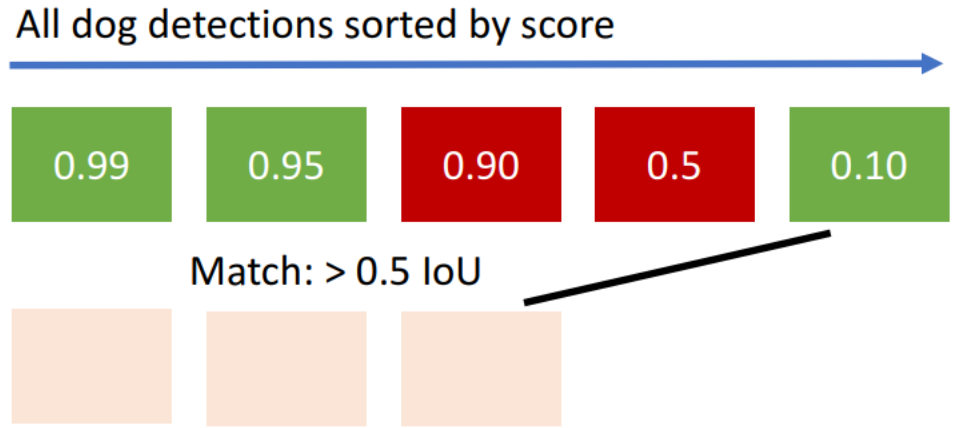
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

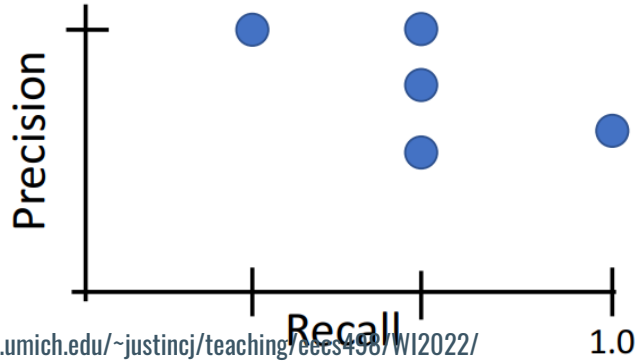
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



All ground-truth dog boxes

Precision = 3/5 = 0.6

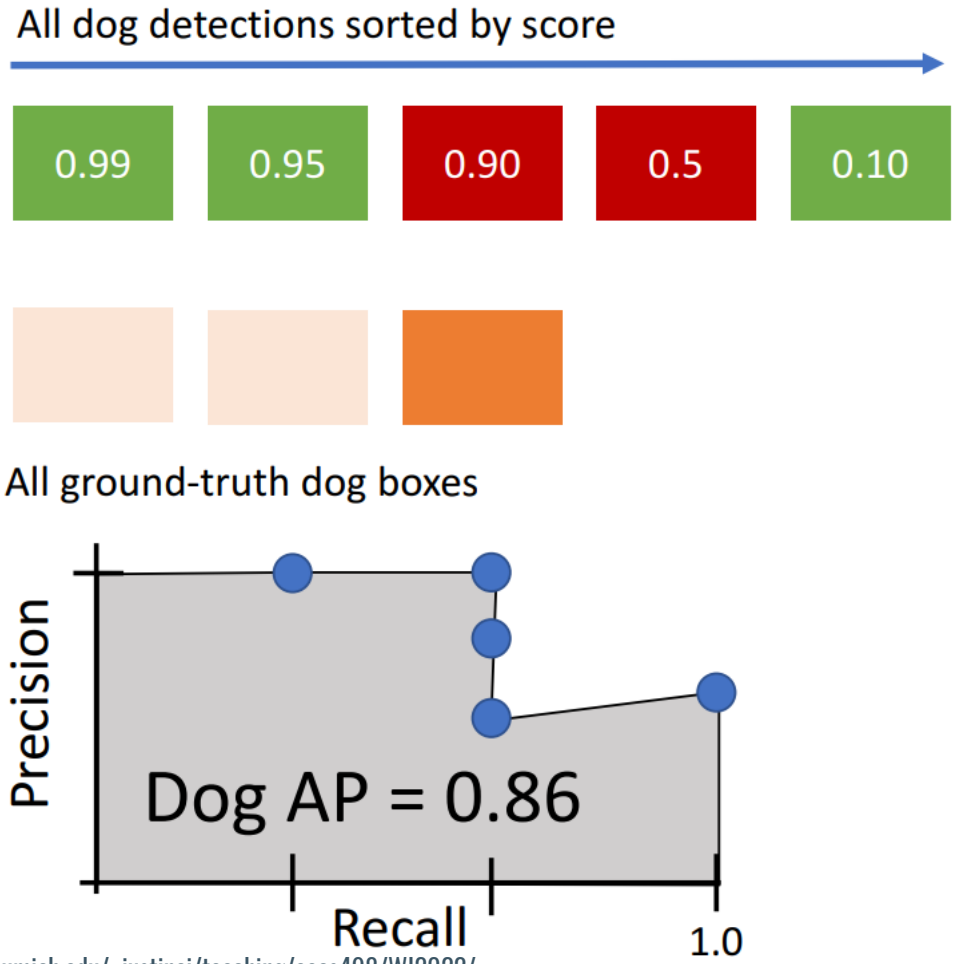
Recall = 3/3 = 1.0



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

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



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

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

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

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

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.2

COCO mAP = 0.4

Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

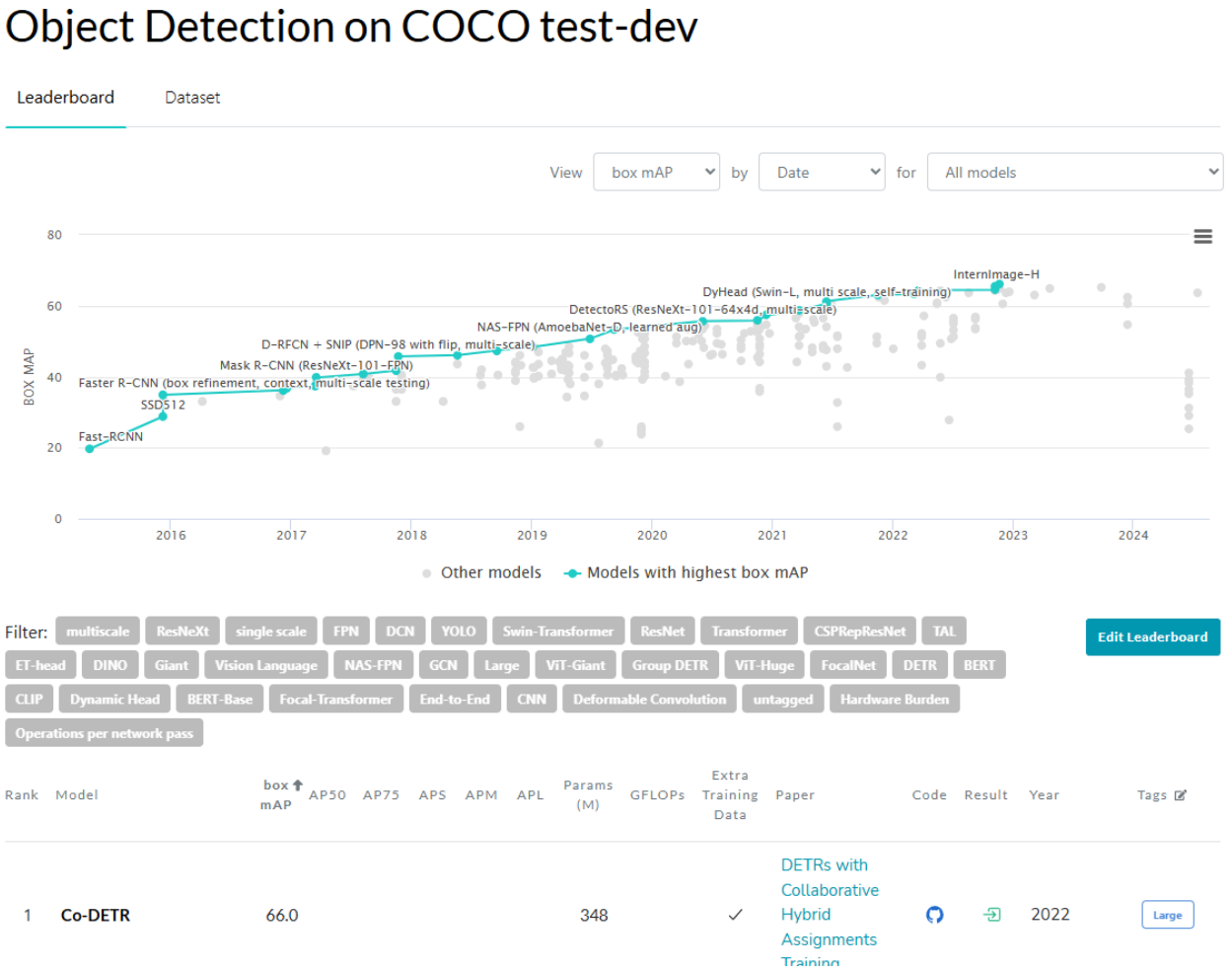
Understanding object detection paper results

Hardware independent metrics:

- **mAP (Mean Average Precision):** Overall average precision across classes and IoU thresholds, indicating detection quality.
- **AP50:** Average precision at an IoU threshold of 0.50, a lenient measure of detection accuracy.
- **AP75:** Average precision at an IoU threshold of 0.75, a stricter measure of accuracy.
- **APS/APM/APL:** Average precision for small, medium, and large objects, assessing model performance across object sizes.
- **Params (M):** Number of model parameters, in millions, indicating model size.
- **GFLOPs:** Computational complexity, showing the number of floating-point operations needed for a forward pass.

Hardware dependent metrics:

- Latency, Inference speed

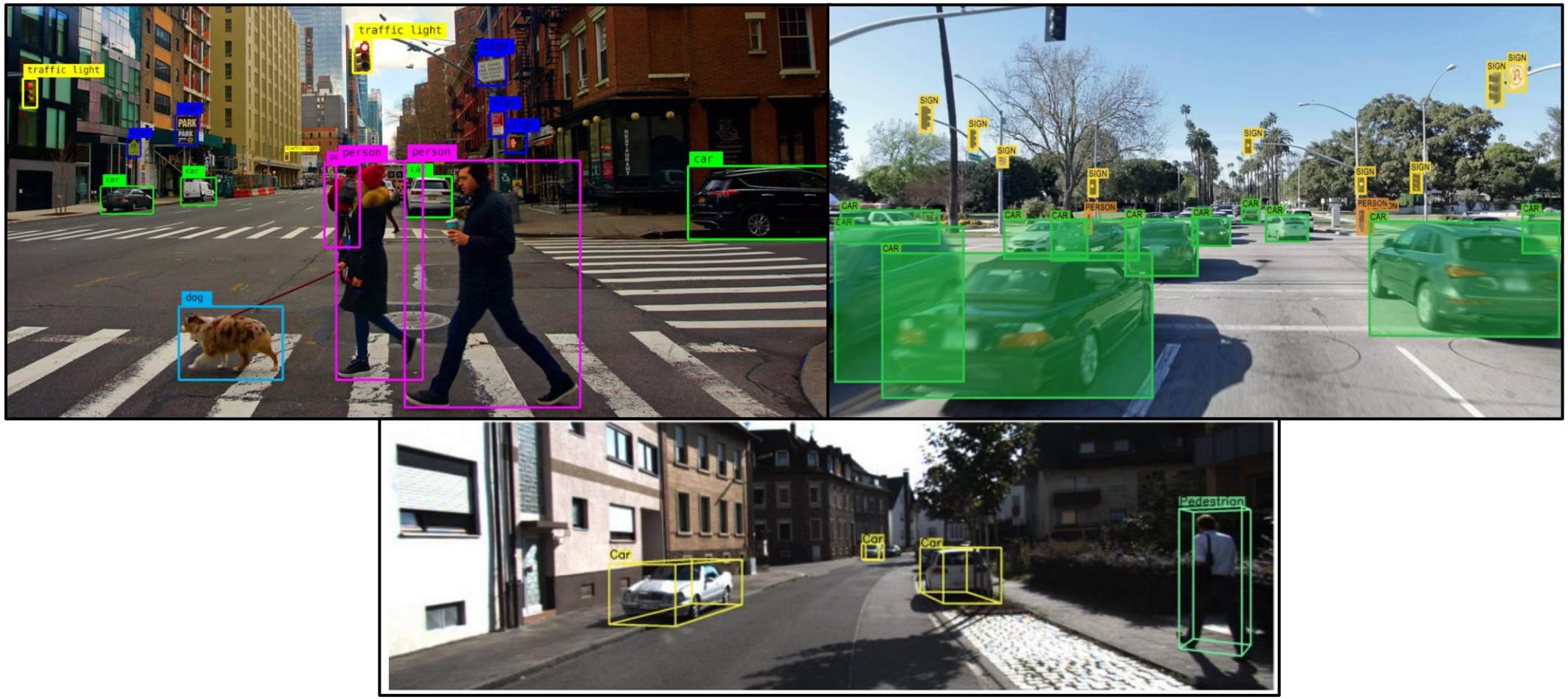


COCO test-dev Benchmark (Object Detection) | Papers With Code

Understanding object detection paper results

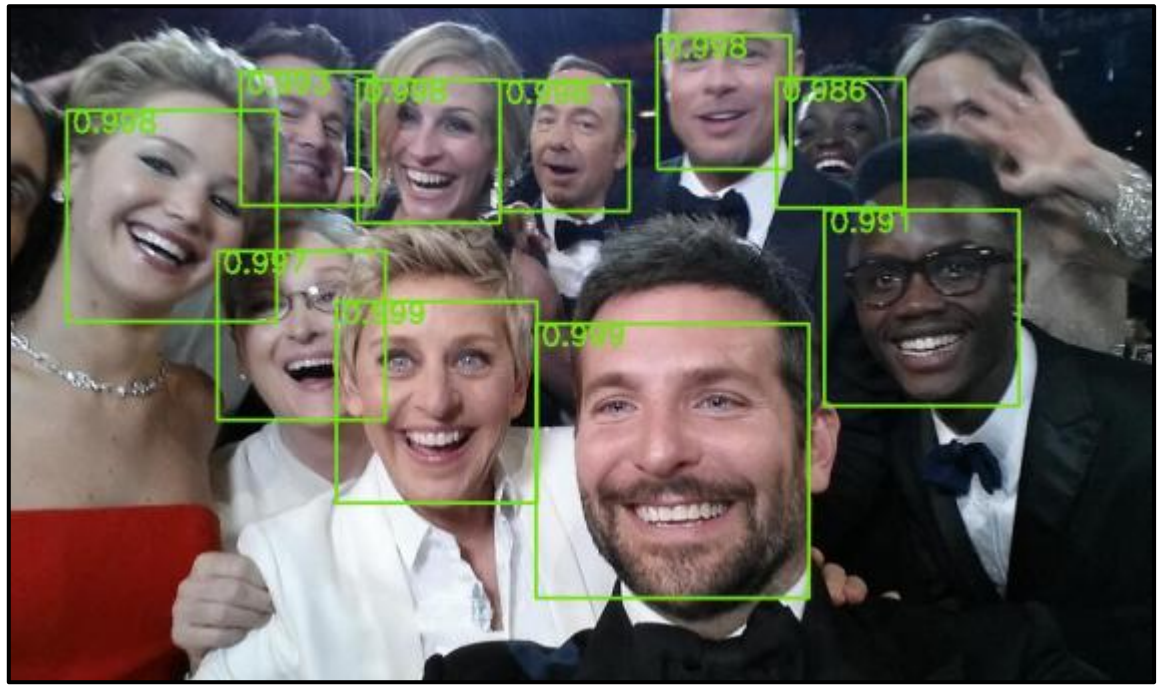
- **mAP (Mean Average Precision):** This is the average precision across different classes and IoU (Intersection over Union) thresholds. It's a comprehensive metric for measuring the performance of object detection models. The higher the mAP, the better the model's precision and recall performance over all categories and thresholds.
- **AP50 (Average Precision at IoU=0.50):** This metric evaluates the average precision when the IoU threshold is set to 0.50. This means that, for a detection to be considered correct, the predicted bounding box must overlap the ground truth by at least 50%. It is often considered a relatively lenient metric.
- **AP75 (Average Precision at IoU=0.75):** Like AP50.
- **APS (Average Precision for Small Objects):** This metric calculates the average precision for detecting small objects. Smaller objects can be more difficult to detect accurately, so this metric specifically tracks how well the model handles smaller object sizes.
- **APM/APL:** Average precision for medium and large objects, assessing model performance across object sizes.
- **Params (M) (Model Parameters in Millions):** This metric shows the number of parameters in the model, typically in millions (M). More parameters often mean a larger, more complex model, but not necessarily better performance.
- **GFLOPs (Giga Floating Point Operations):** This refers to the number of floating-point operations required to make a single forward pass through the model, usually measured in gigaflops (GFLOPs). It is an indicator of the model's computational complexity and can be used to gauge the efficiency of the model.

Autonomous Driving



Inventory Scan

Face Detection



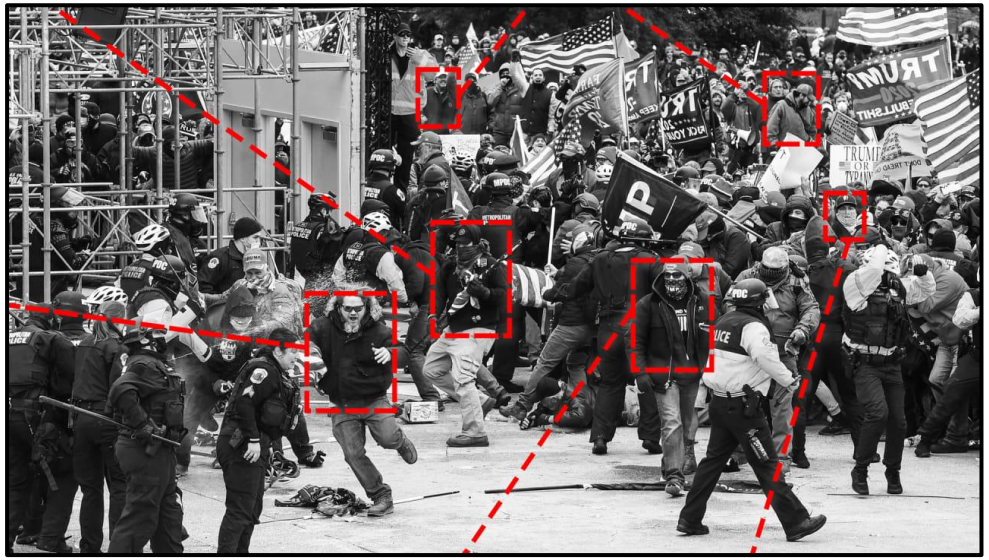
+ Face Recognition (endless applications)

Attendance check

Entry access to specific place

Security

...



Deep Network Development

Lecture 8.

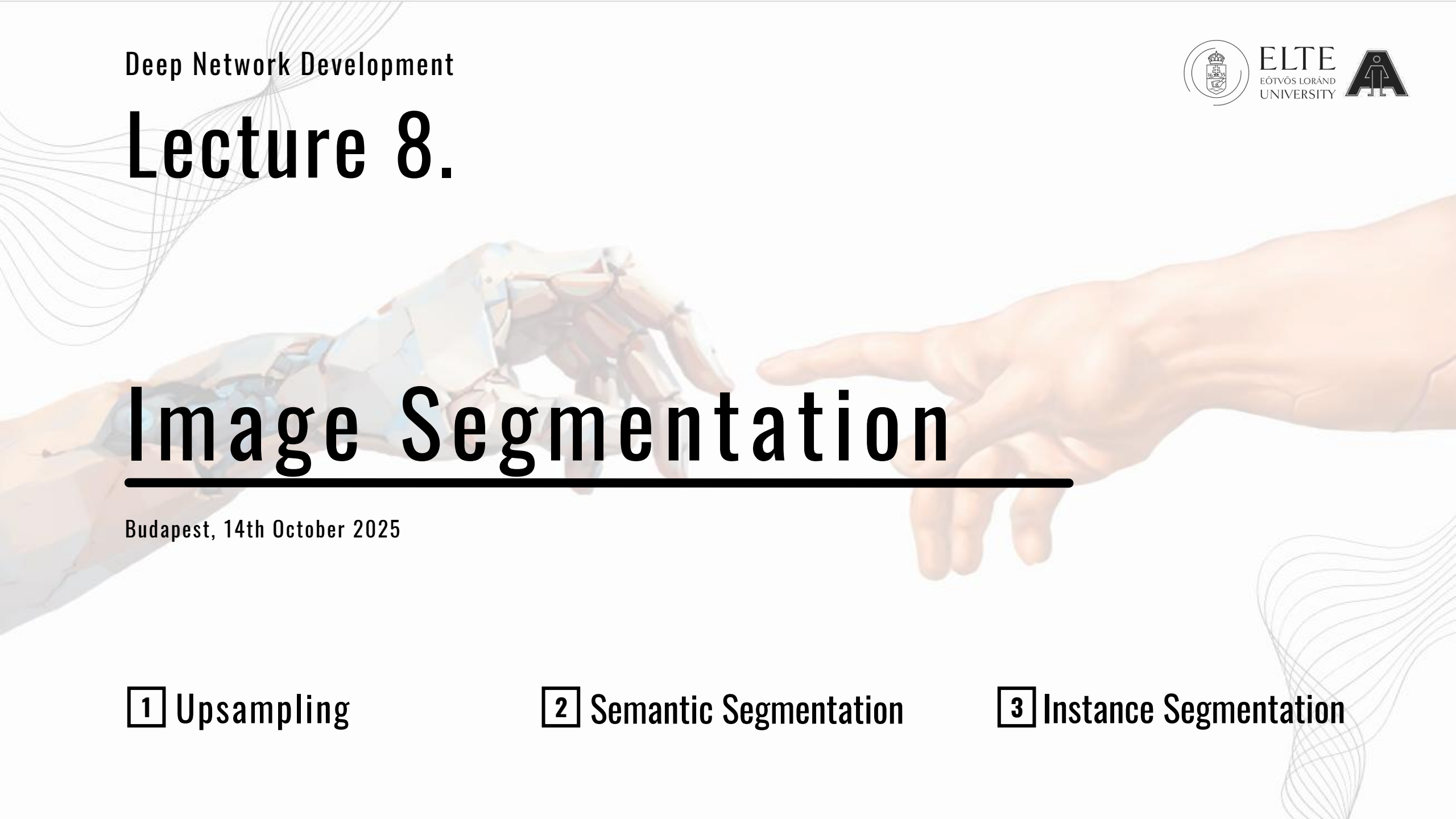


Image Segmentation

Budapest, 14th October 2025

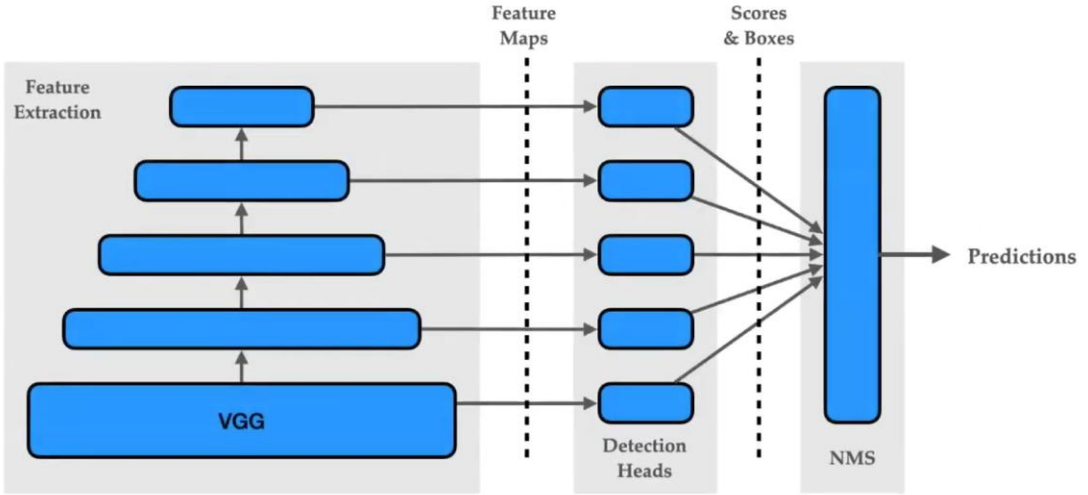
1 Upsampling

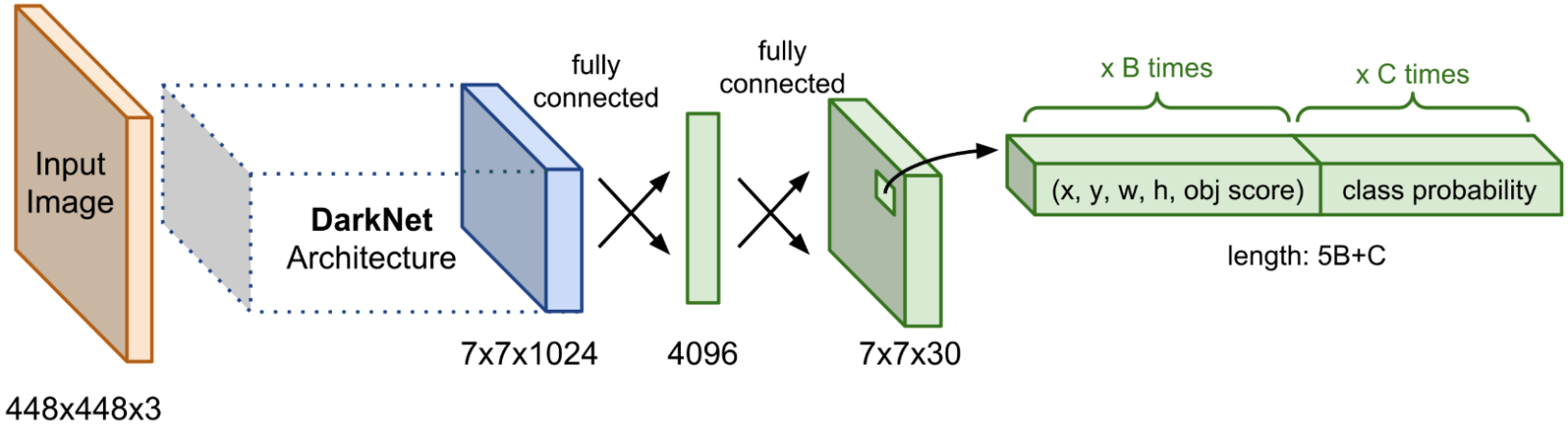
2 Semantic Segmentation

3 Instance Segmentation

Previously on Lecture 5

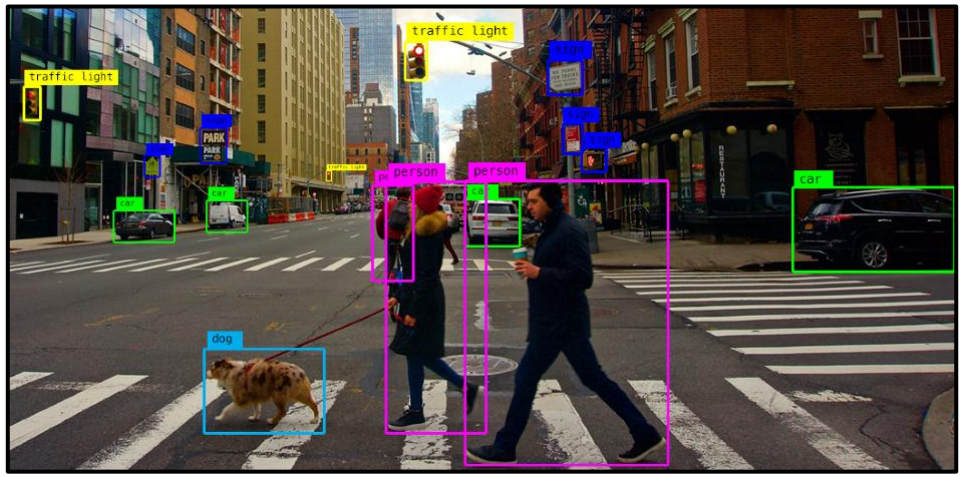
- Single-Shot Detectors: Naïve SSD, YOLO
- Multi-box Detection
- Non-Max Suppression (NMS)





Previously on Lecture 5

- Object Detection Metrics: IoU, mAP, etc.
- Applications of Object Detection



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

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

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

Supervised Learning tasks

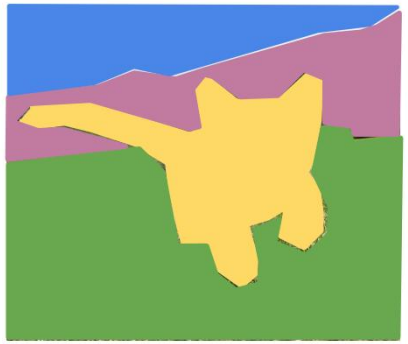
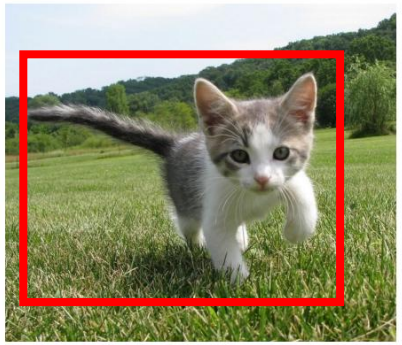
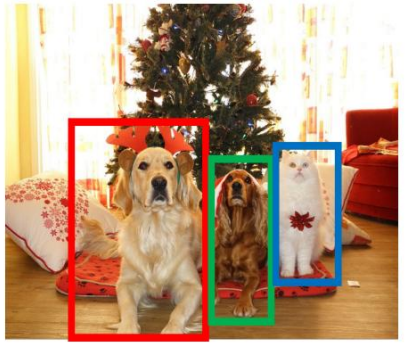
Classification	Semantic Segmentation	Classification + Localization	Object Detection	Instance Segmentation
				
CAT	GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT
Single Object	No objects, just pixels	Single Object	Multiple Objects	Multiple Objects
✓	⟳	✓	✓	⟳

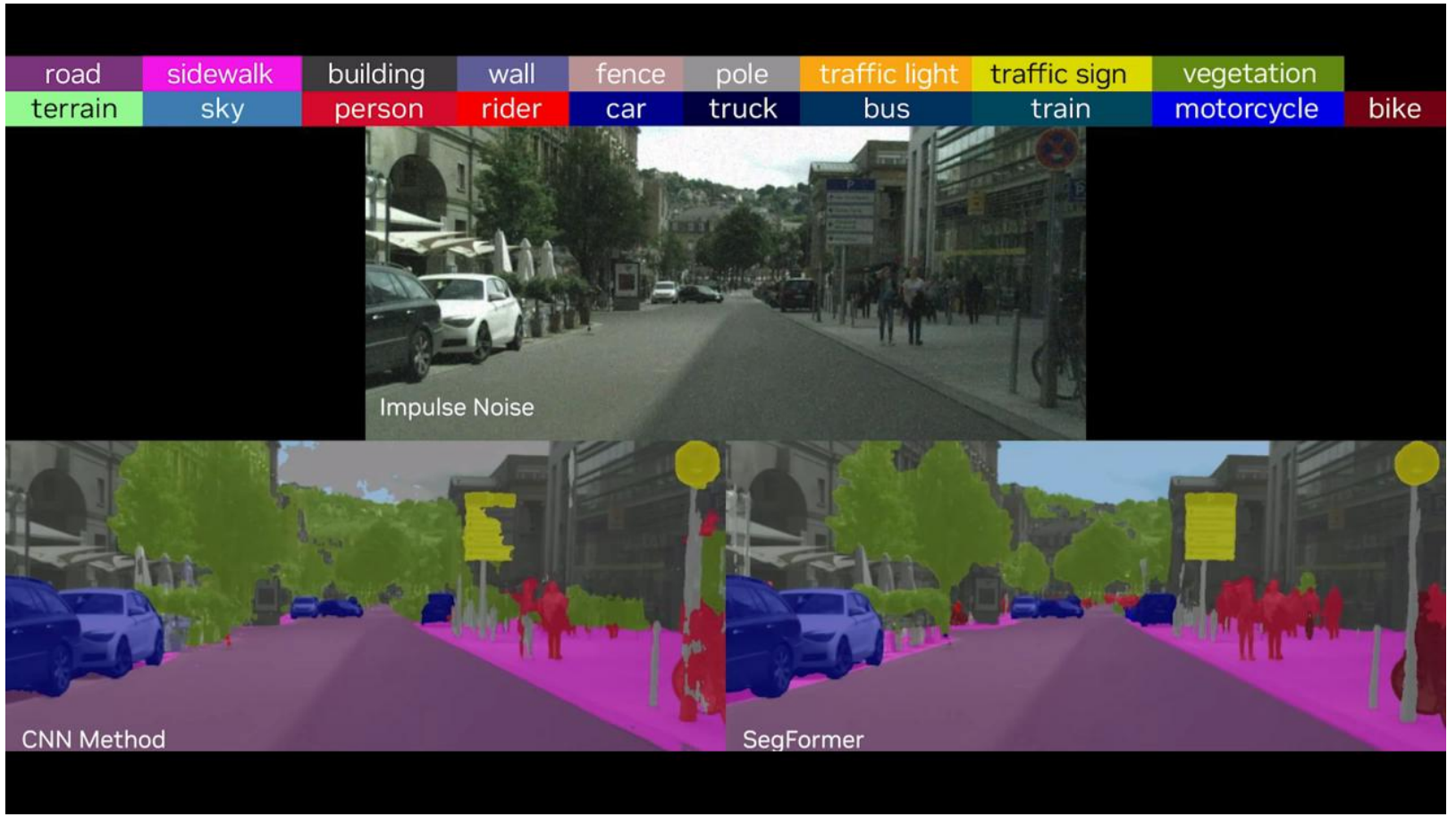
Image Segmentation

Semantic Segmentation

Instance Segmentation

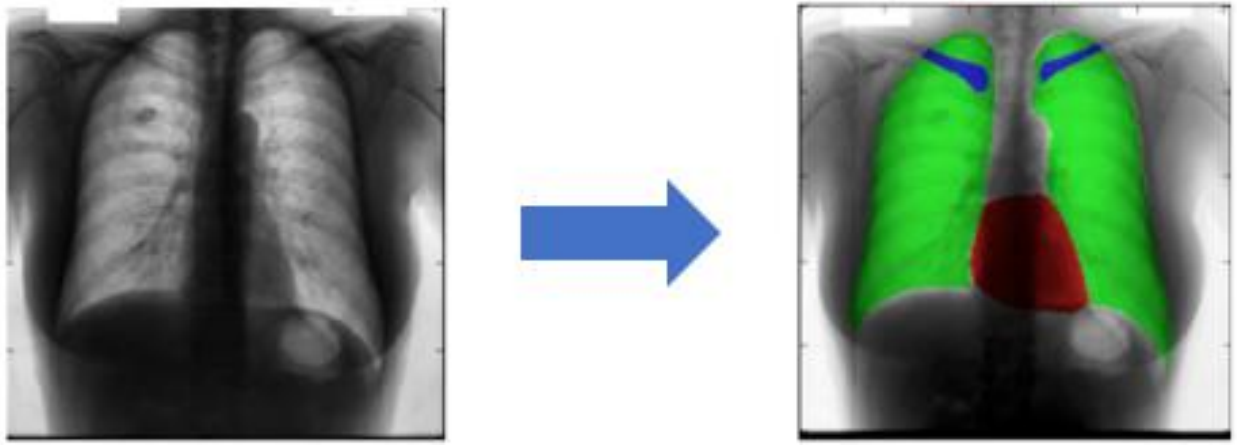
Video Segmentation

Autonomous Driving example - NVIDIA DRIVE (2024)



Applications

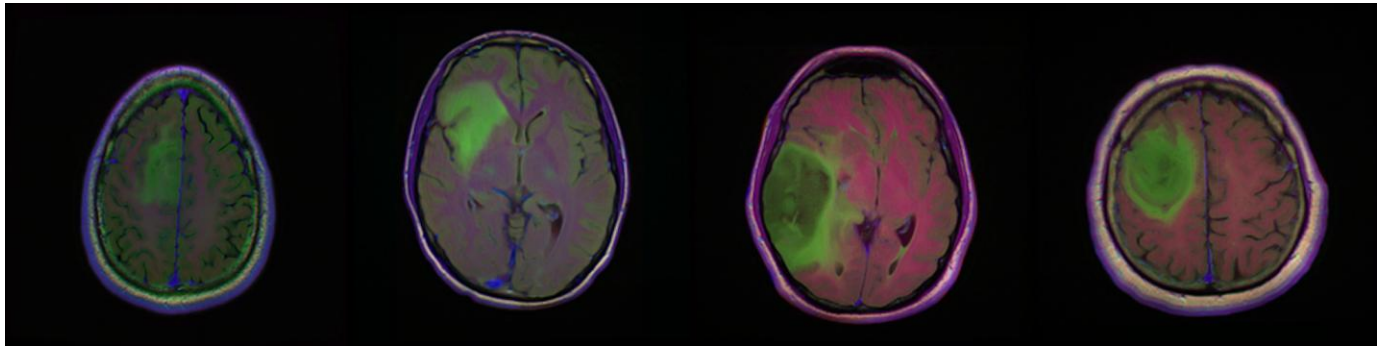
- Medical image diagnosis



Input Image

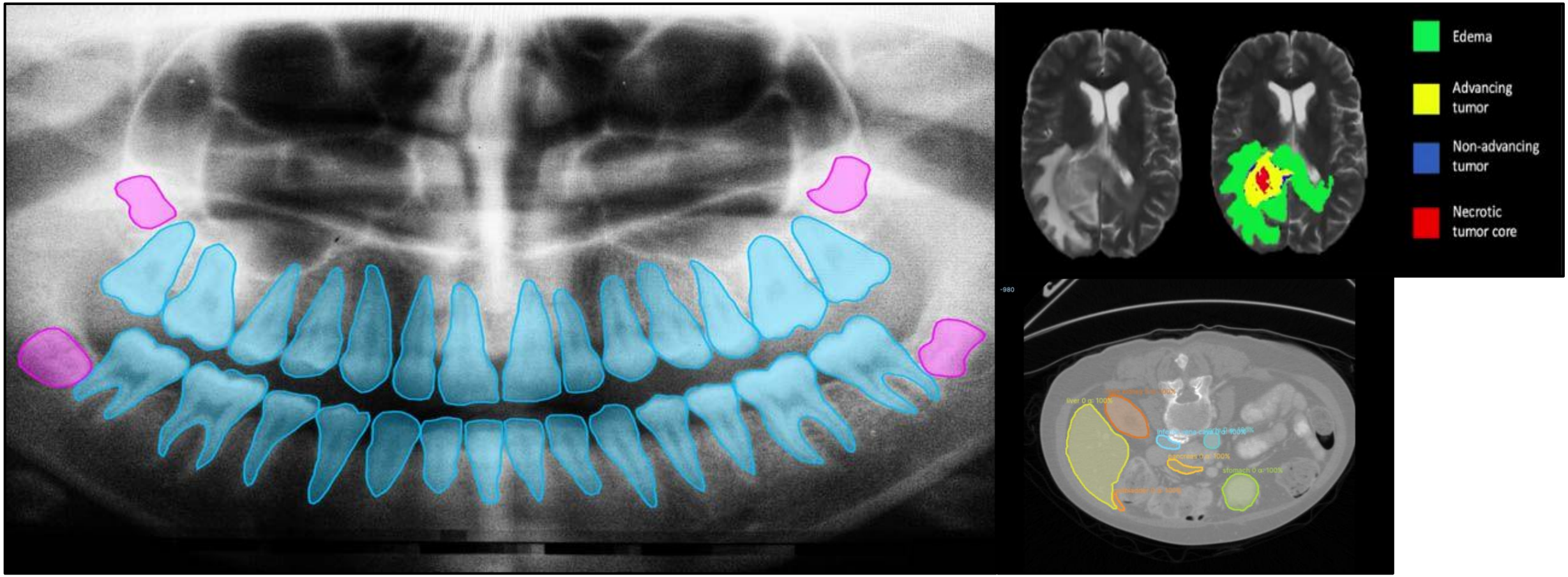
Segmented Image

https://github.com/mateuszbuda/brain-segmentation-pytorch
https://github.com/Saswatm123/3D-Brain-Tumor-Segmentation-PyTorch
https://github.com/hahnicity/pytorch-lung-segmentation



Applications

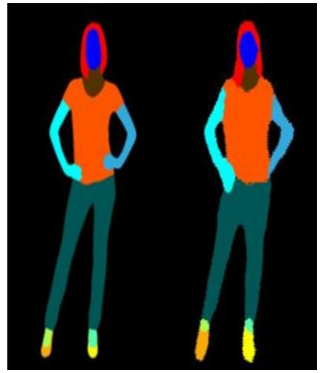
- Medical image diagnosis



Applications

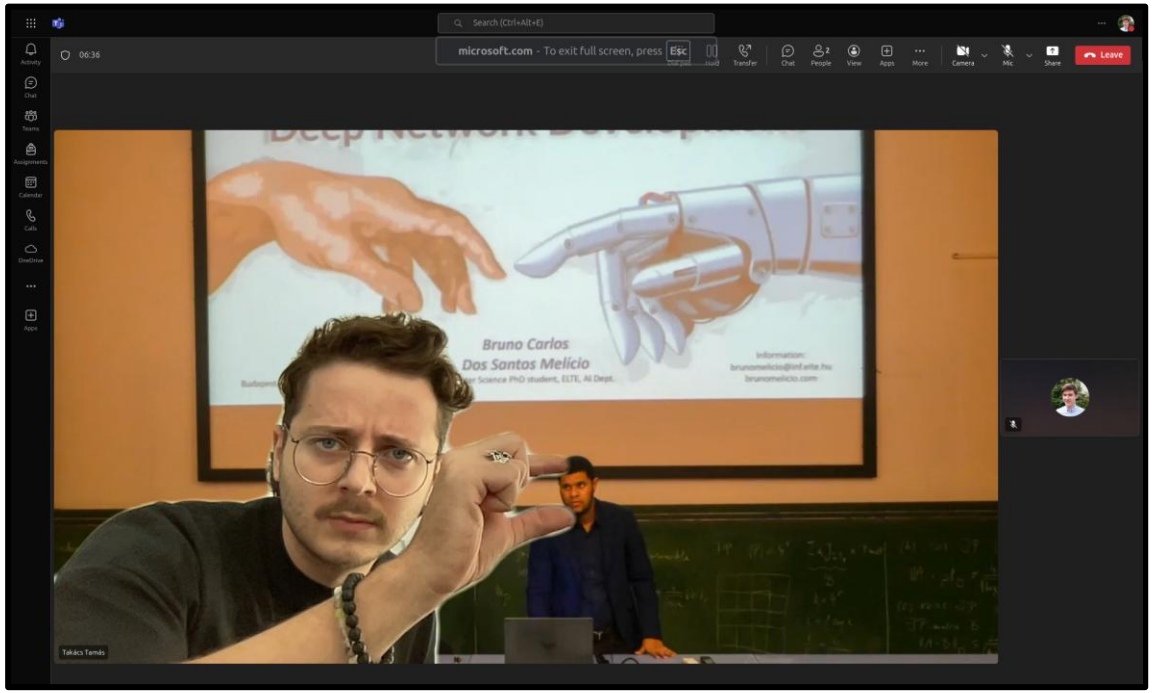
- Entertainment
  - Photo effect
  - Virtual try on

https://towardsdatascience.com/semantic-image-segmentation-with-deeplabv3-pytorch-989319a9a4fb
https://github.com/thuyngch/Human-Segmentation-PyTorch
https://github.com/kishorkuttan/Deep-Virtual-Try-On
https://github.com/shadow2496/VITON-HD
https://github.com/JDAI-CV/Down-to-the-Last-Detail-Virtual-Try-on-with-Detail-Carving



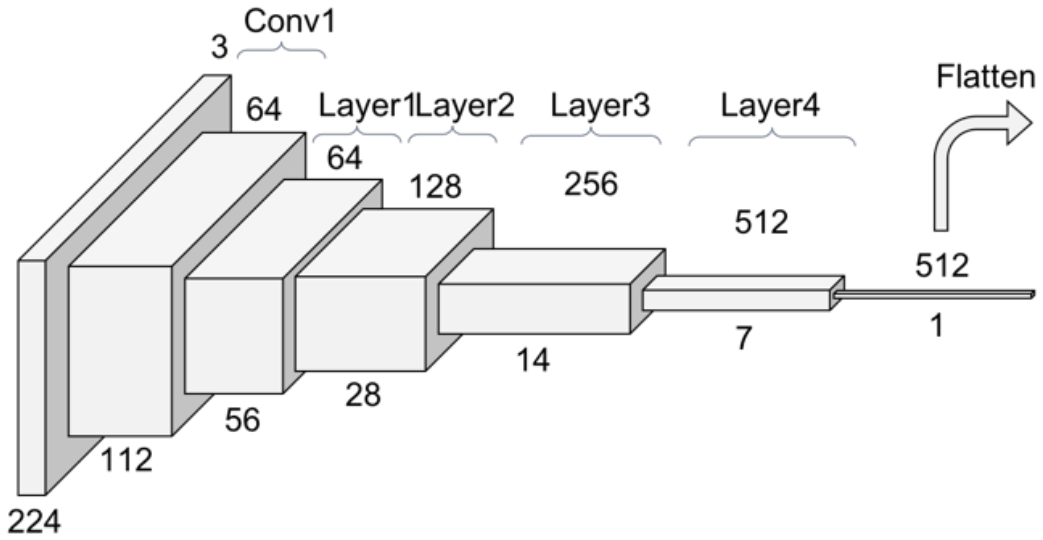
Video Segmentation

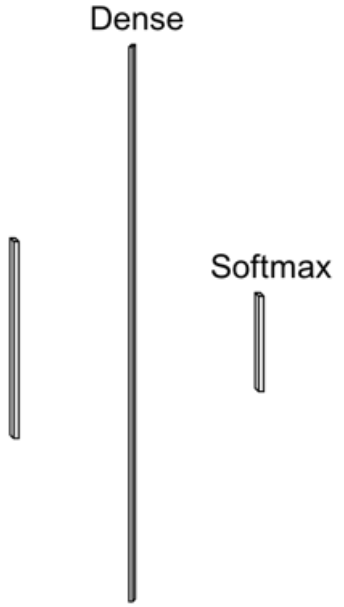
- Microsoft Teams segmentation



Classification Task

We map images (x) to labels (y)



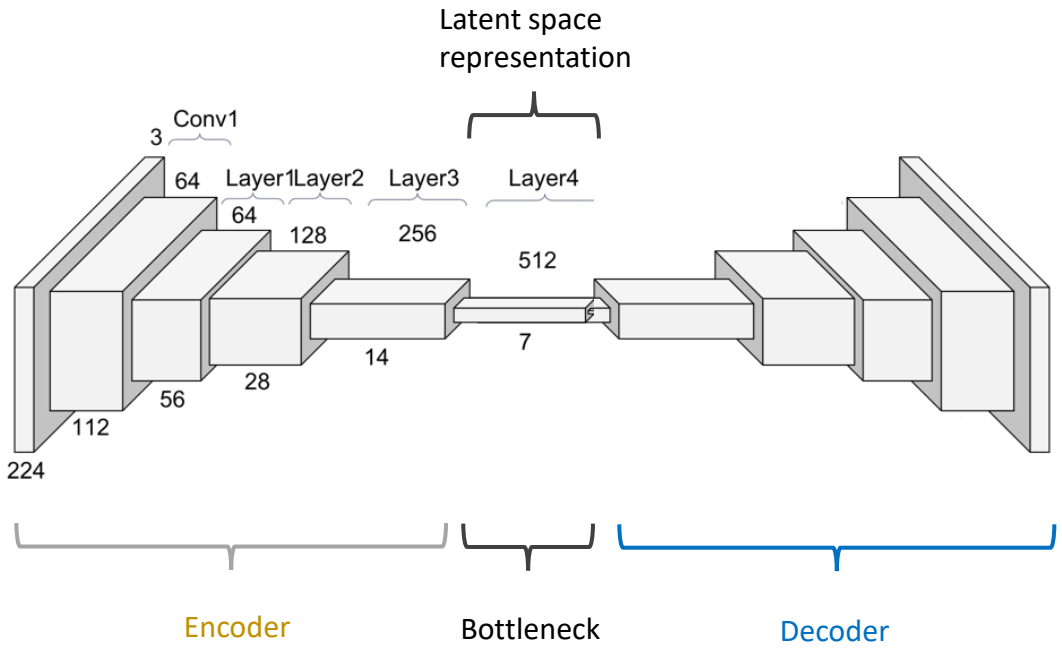


0.01	Dog
0.01	Cat
0.91	Racoon*
…	…
0.01	Flower

Reconstruction task

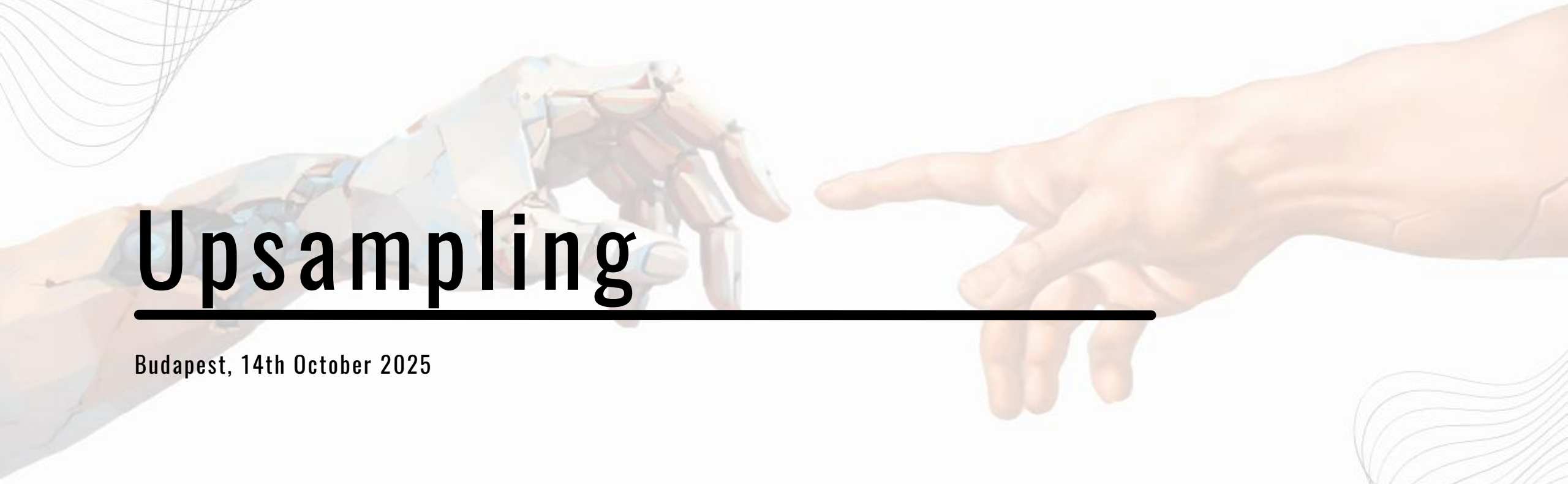
We try to get back the original image while constraining the network to only learn meaningful information

- Can be used for denoising images
- We lose information during the constraining
- How can we upsample from the latent representation?



Deep Network Development

Lecture 8.



Upsampling

Budapest, 14th October 2025

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

Upsampling

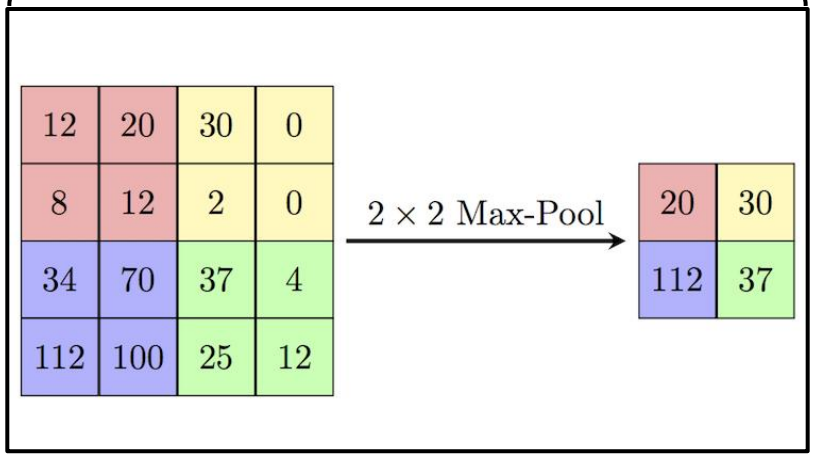
How to upsample?

1. Unpooling
2. Transposed Convolution

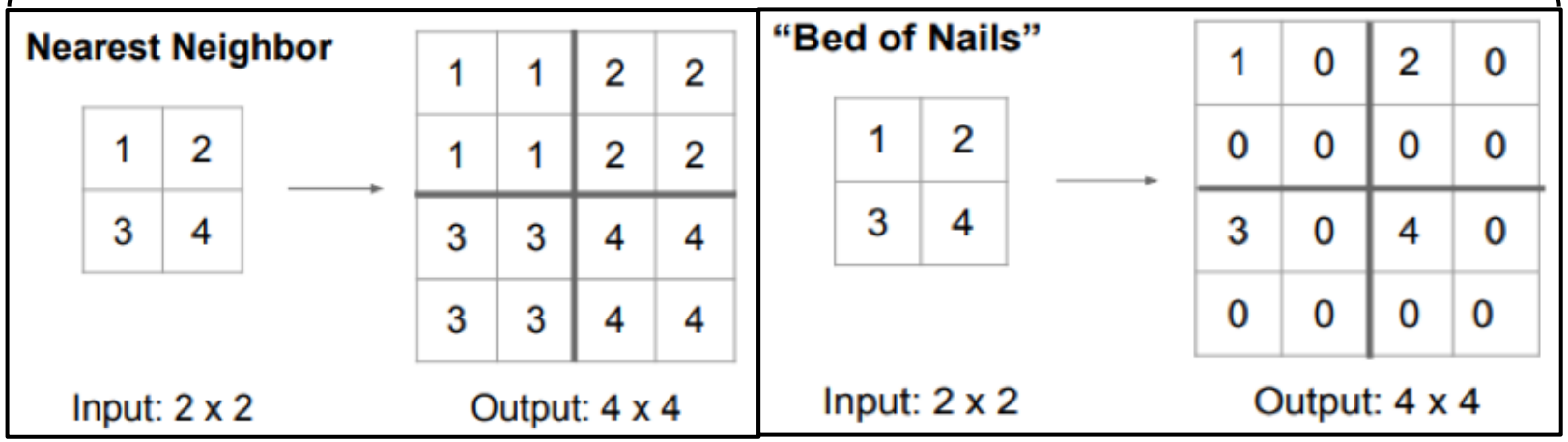
Unpooling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (ie. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.

Pooling



Unpooling

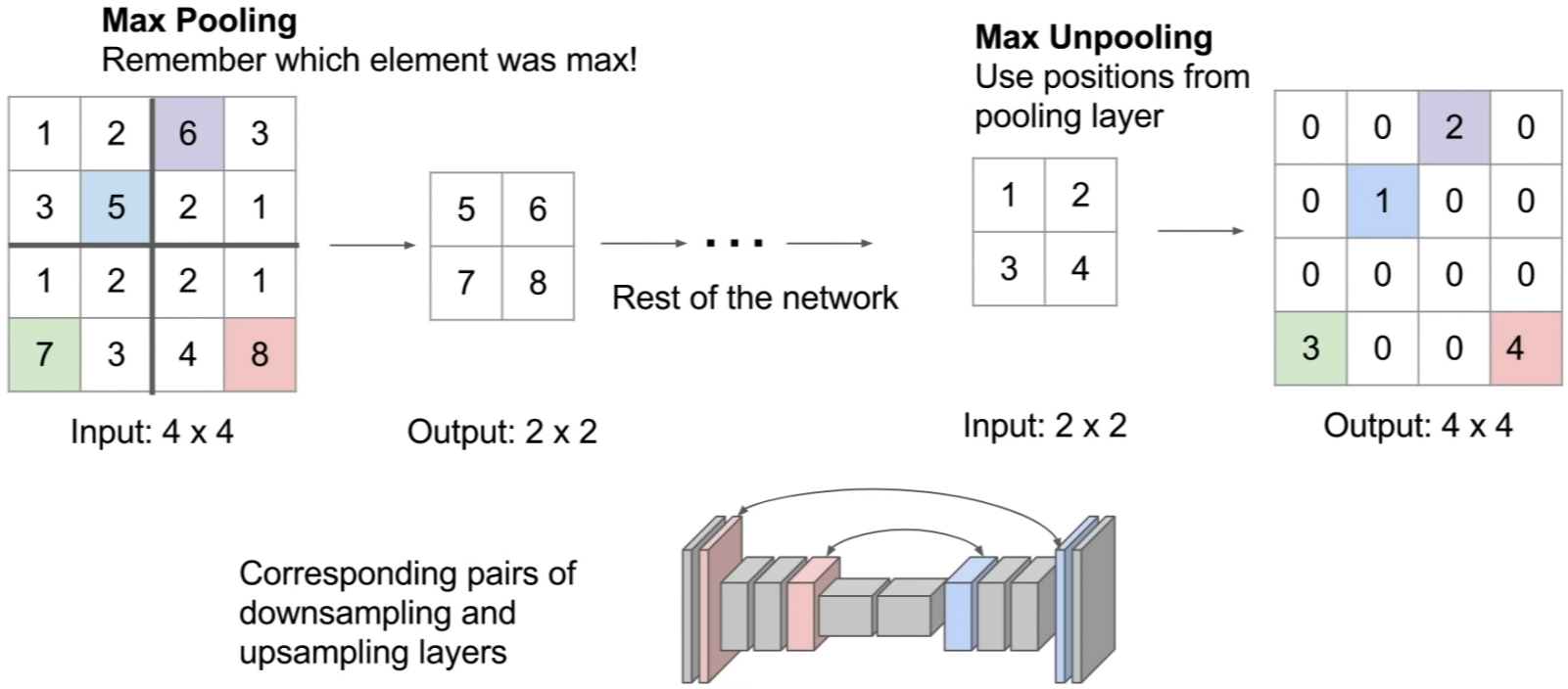


Bilinear
Linear Shifted

Unpooling

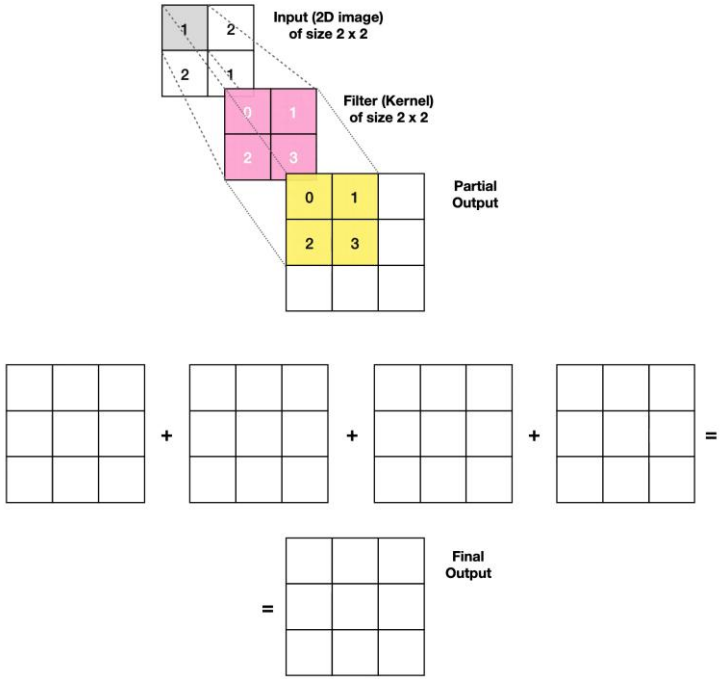
- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (I.e. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.
- **No weights, nothing to learn here!**

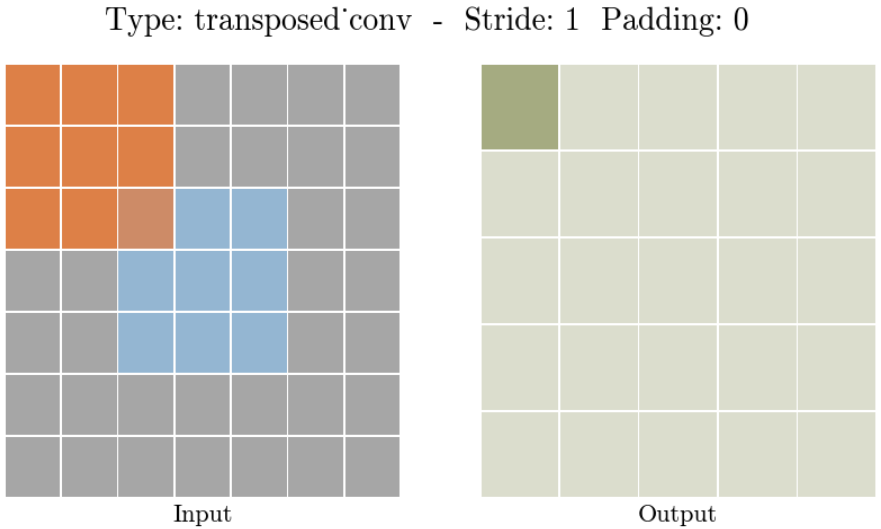
In-Network upsampling: “Max Unpooling”



Transposed Convolution

- Most popular approach
- Whereas a typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position, a **transpose convolution** essentially does the opposite. For a transpose convolution, we take a single value from the low-resolution feature map and multiply all the weights in our filter by this value, projecting those weighted values into the output feature map.

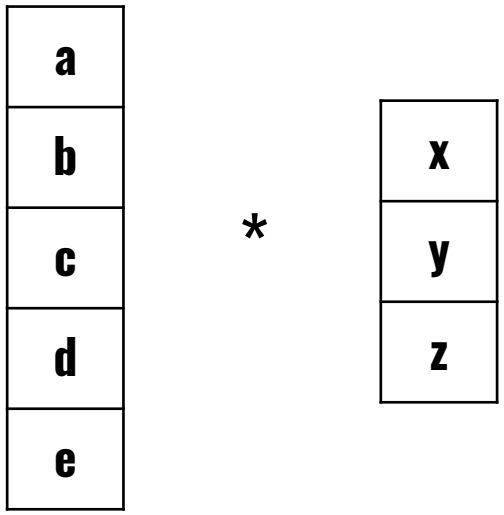




Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3
Stride: 2

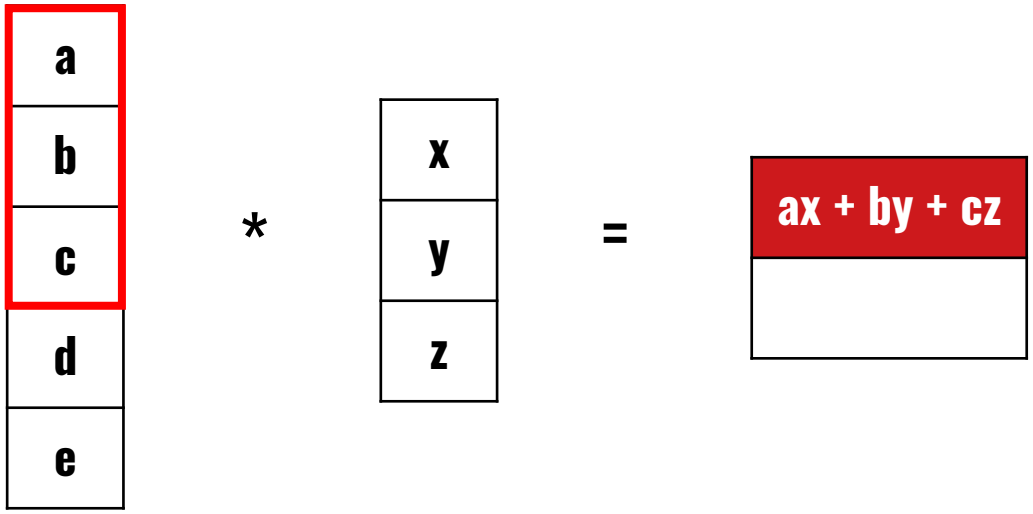


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3 **Output size:** 2
Stride: 2

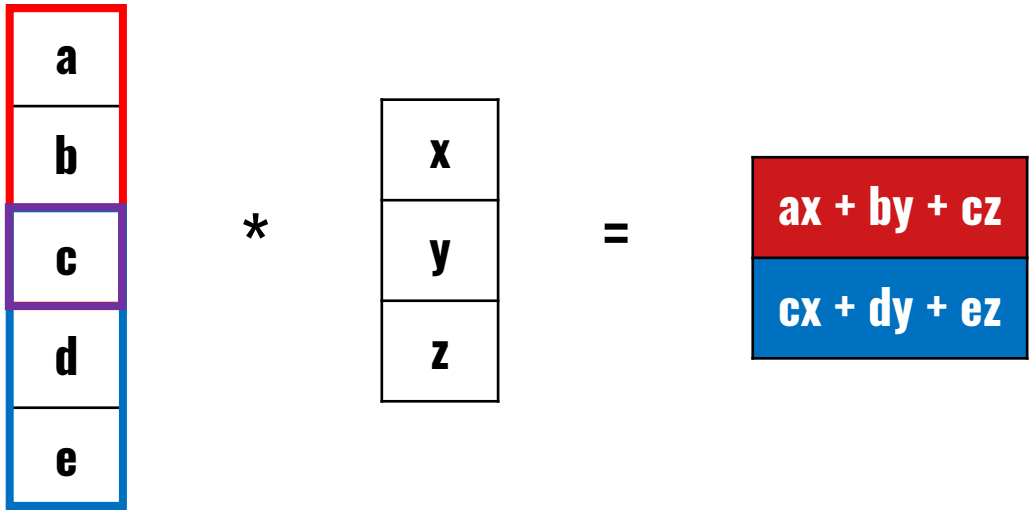


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3 **Output size:** 2
Stride: 2



$n \times n$ *image* $f \times f$ *filter*
padding p *stride* s

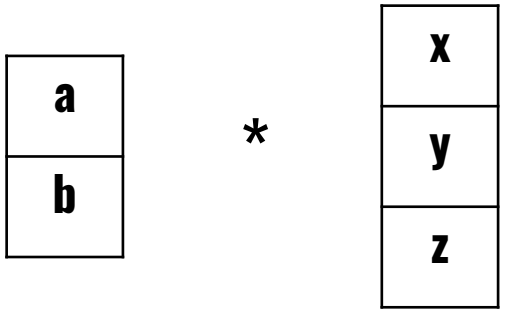
$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3
Stride: 2

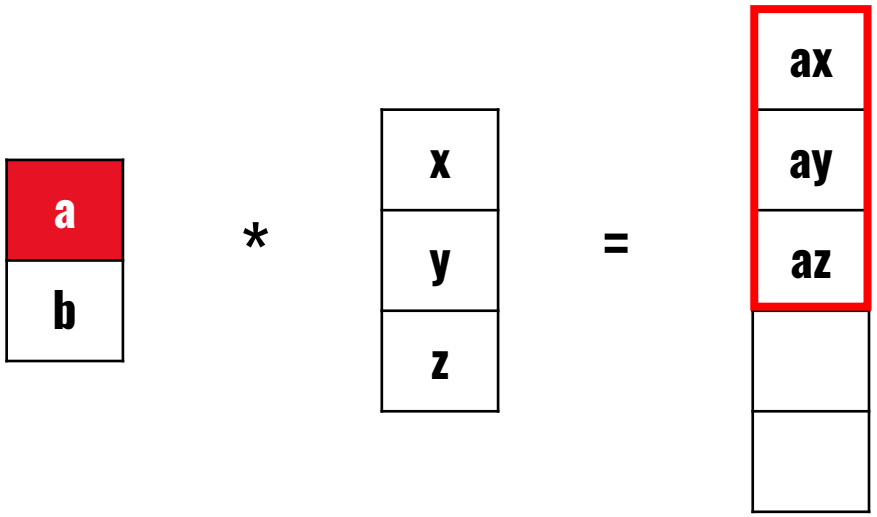


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3 **Output size:** 5
Stride: 2

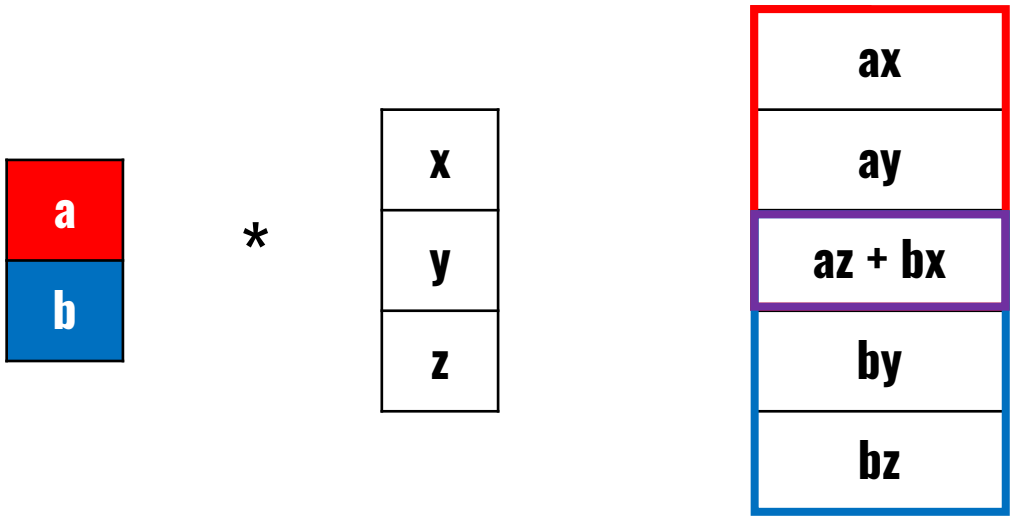


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3 **Stride:** 2 **Output size:** 5

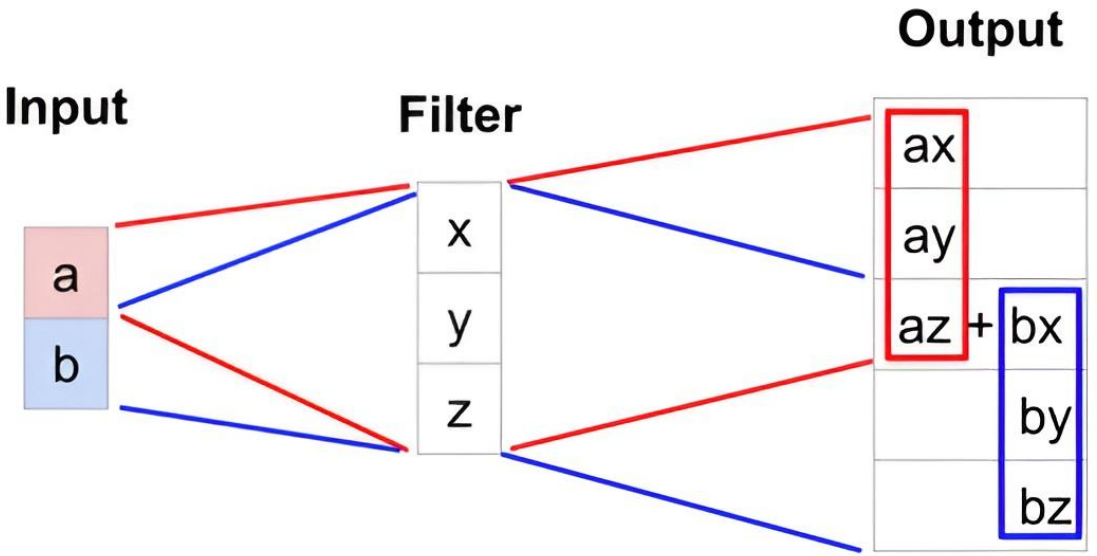


$$\textbf{output size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example
- For filter sizes which produce an overlap in the output feature map, the overlapping values are simply added together.
- Less common names: **Deconvolution**, **Fractionally strided convolution**, **Up convolution**, ...



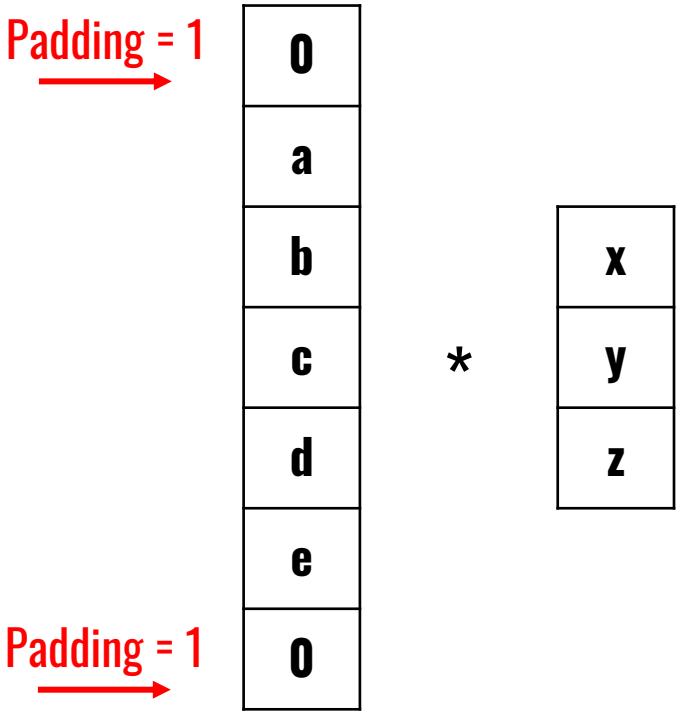
Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

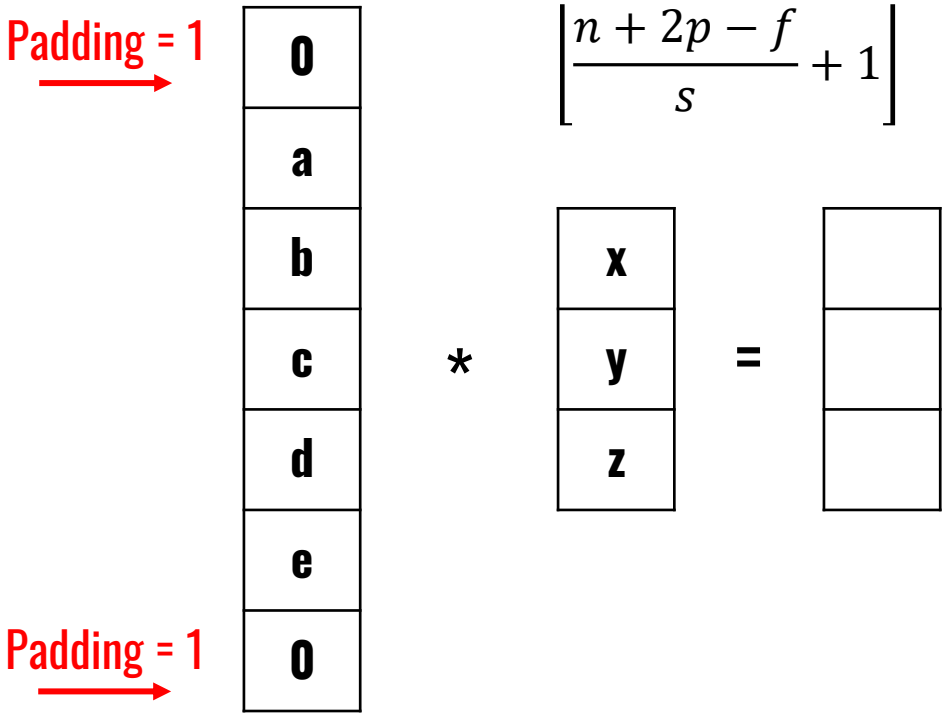
Normal Convolution

$n \times n$ *image* $f \times f$ *filter*

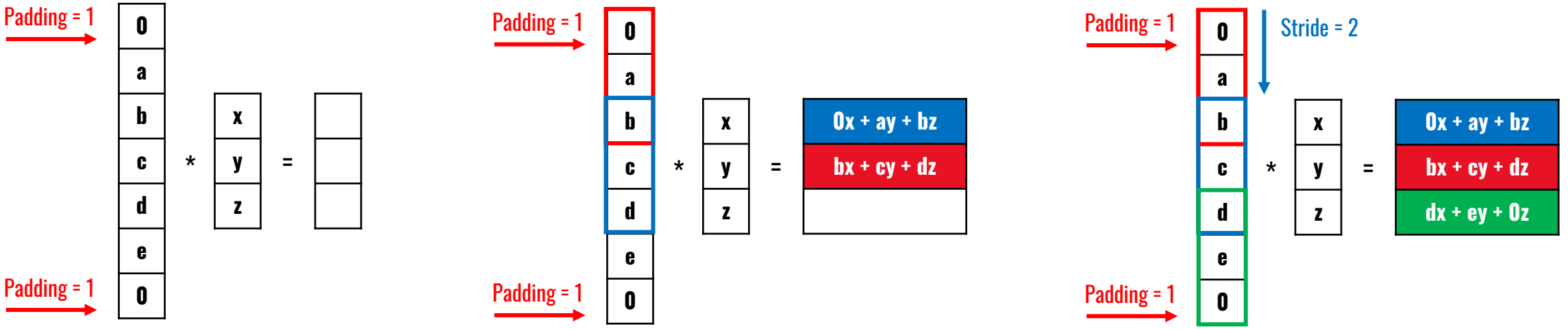
padding p *stride* s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$



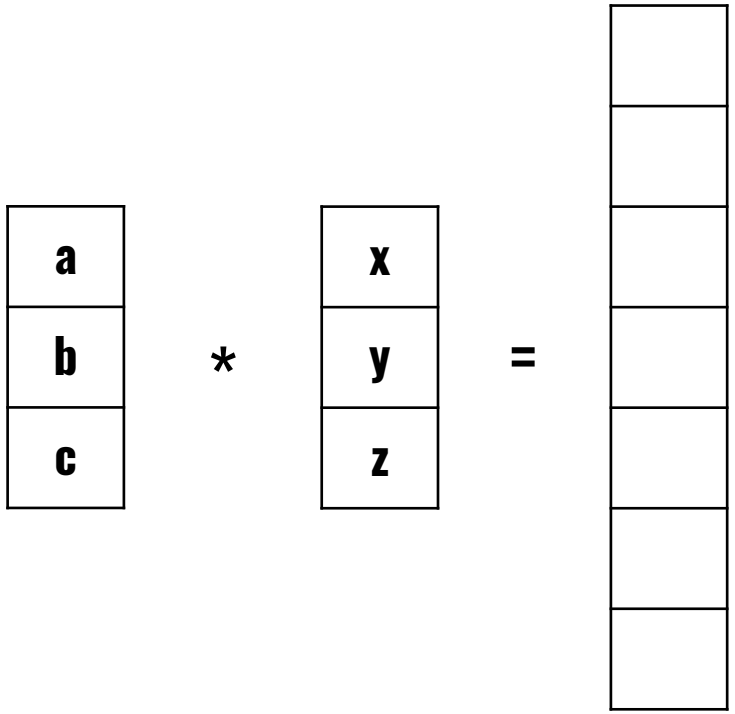


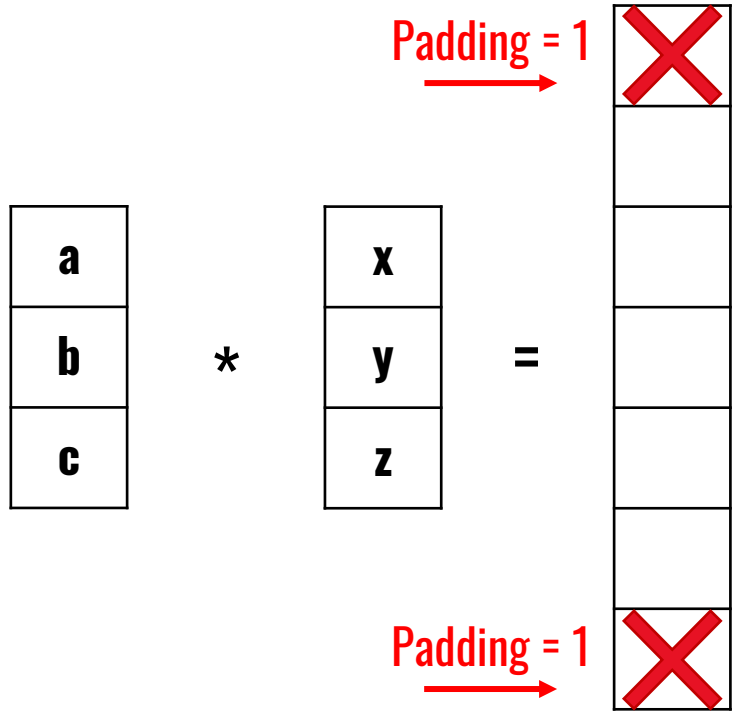
Normal Convolution



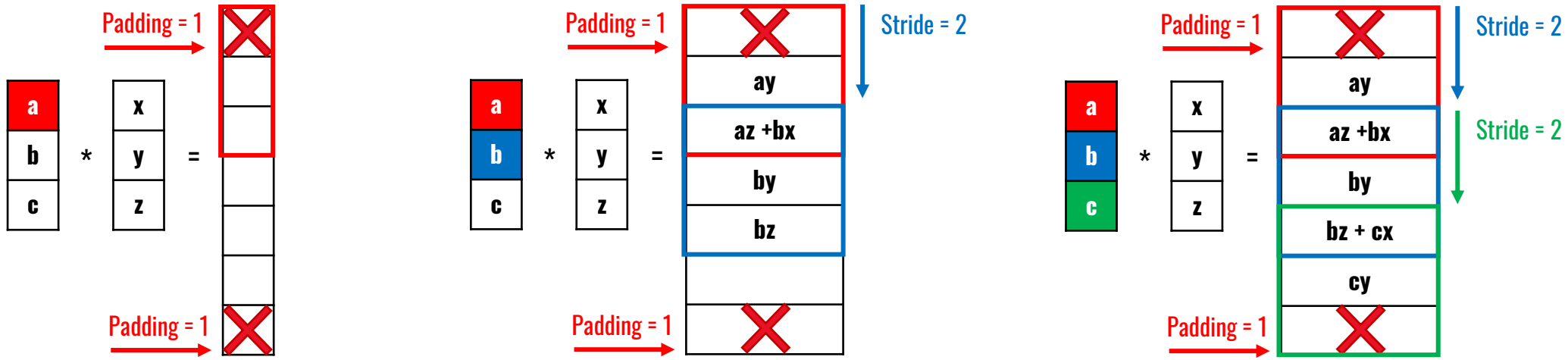
Transposed Convolution

$$\textbf{output size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

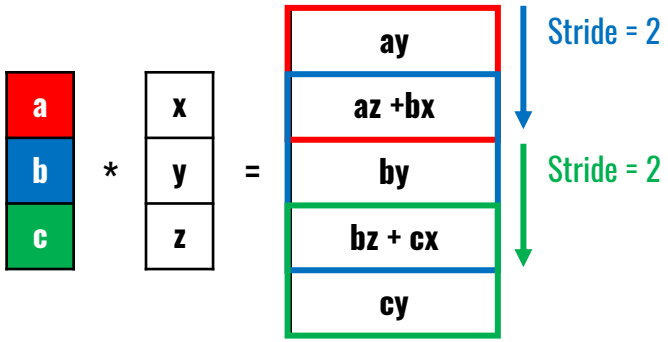


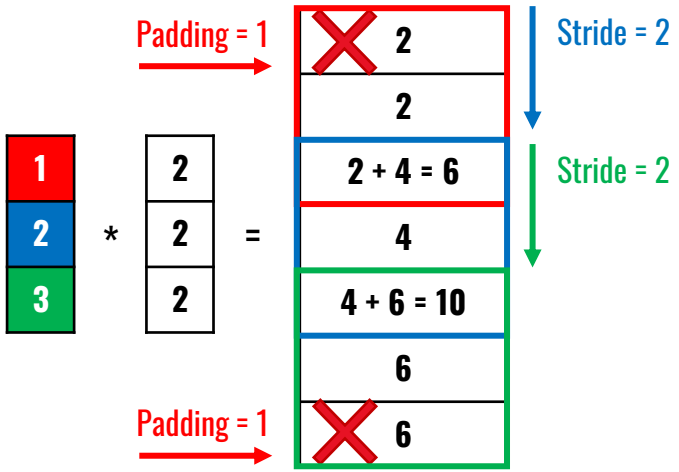


Transposed Convolution



Transposed Convolution

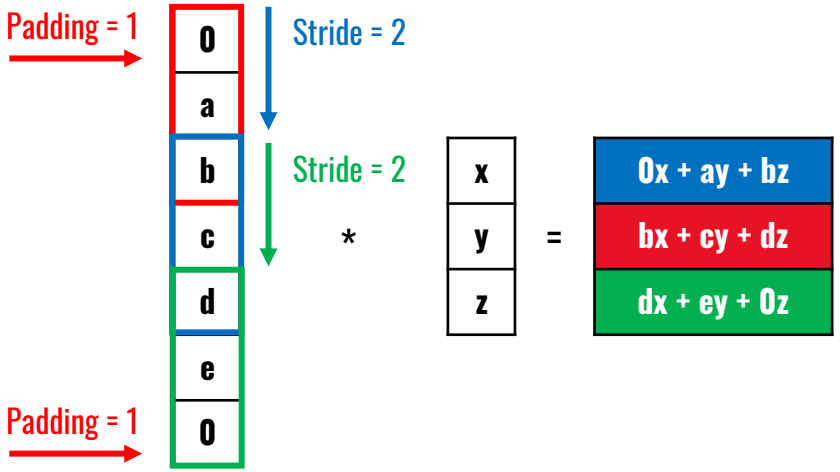




Summary on 1D

Convolution

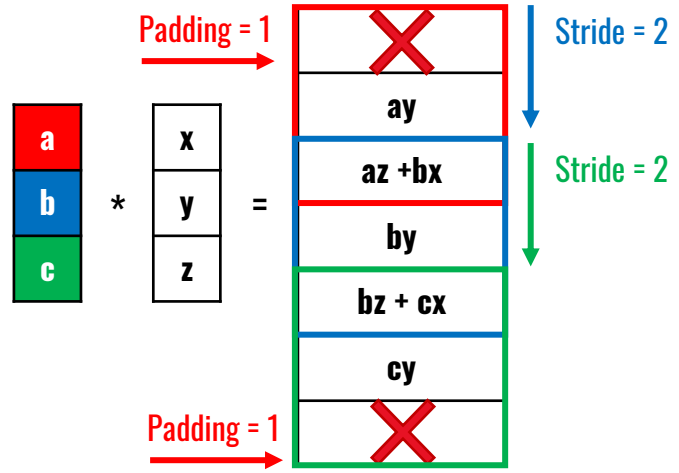
Input size: 5 **Filter size:** 3 **Stride:** 2 **Padding:** 1 **Output size:** 3



VS

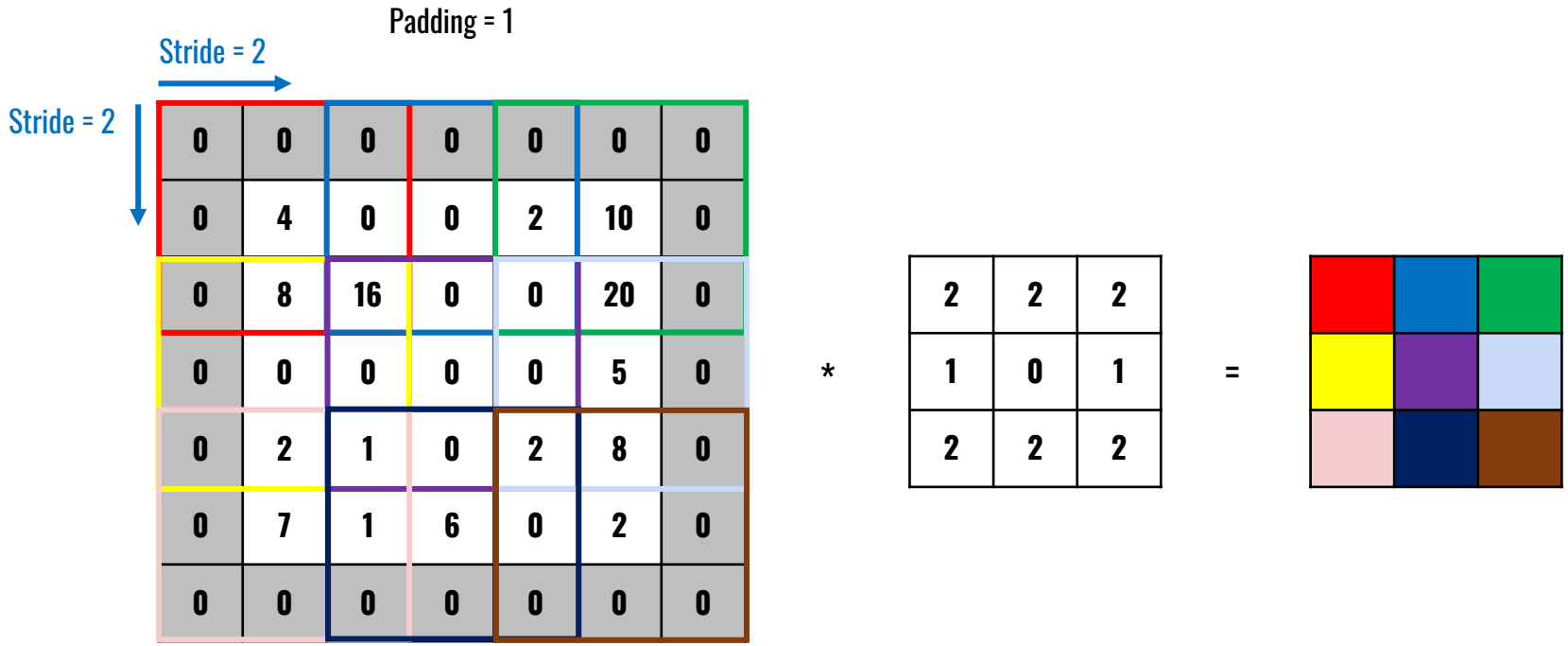
Transposed Convolution

Input size: 3 **Filter size:** 3 **Stride:** 2 **Padding:** 1 **Output size:** 5



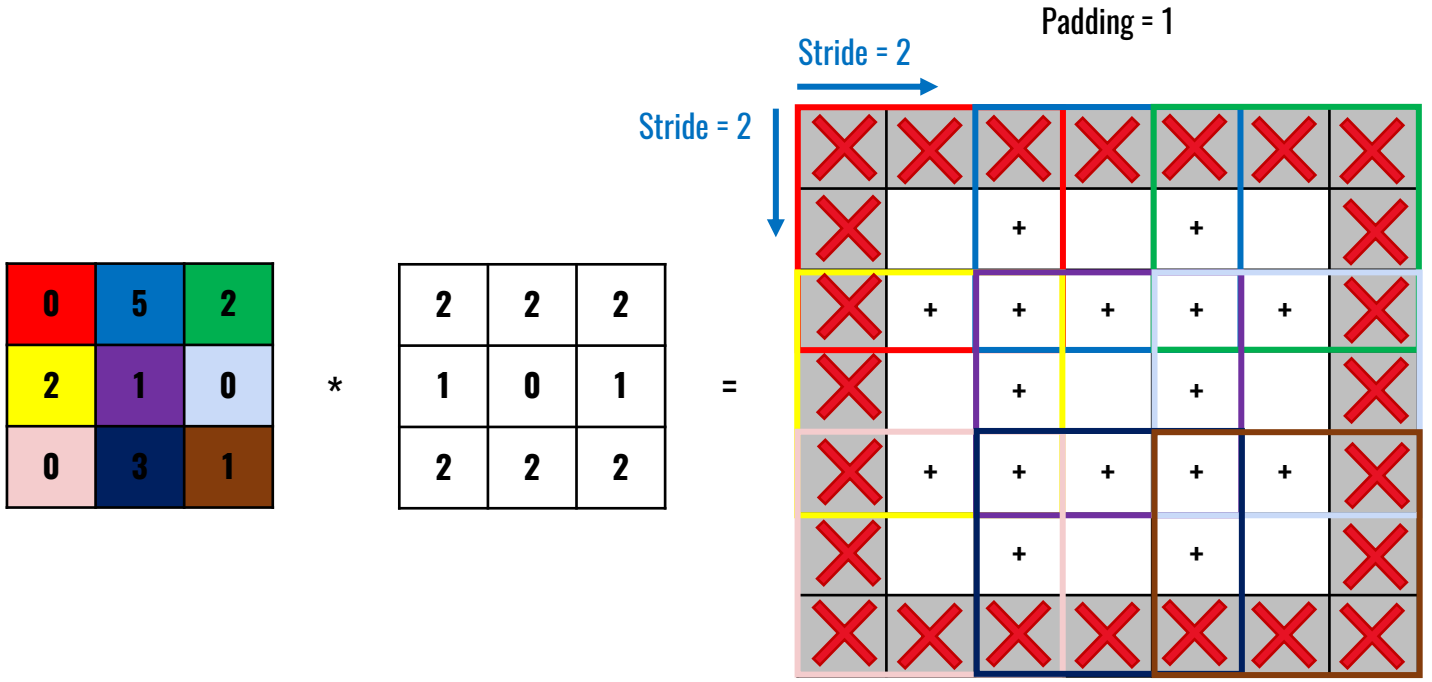
Convolution

- 2D example



Transposed Convolution

- 2D example



Interactive Jupyter Notebook available on Canvas

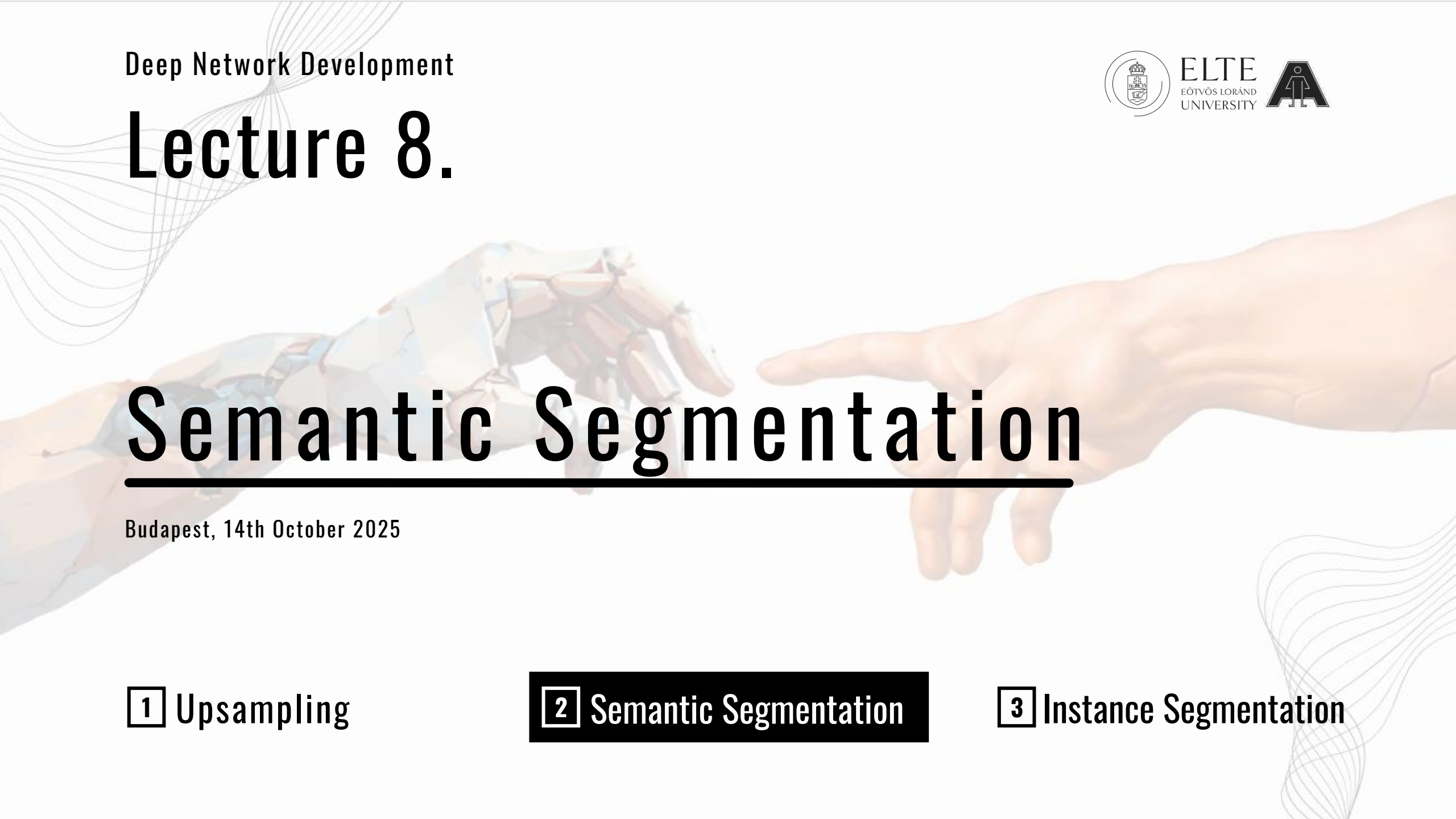
```
$jupyter nbconvert <notebook_name>.ipynb --to slides --post serve
```

Interactive Explanation on HuggingFace

https://huggingface.co/spaces/PercibalBuxus/TransposedConvolutions

Deep Network Development

Lecture 8.



Semantic Segmentation

Budapest, 14th October 2025

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

Semantic Segmentation

Semantic image segmentation is the task of **labelling each pixel** of an image with a corresponding **class** of what is being represented.

Input: RGB image (height × width × 3) or a grayscale (height × width × 1)

Output: a segmentation map (height × width × 1), where each pixel contains a class label represented as an integer.

predict

Person
Bicycle
Background

Input

segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Semantic Labels

Multiclass Classification (Recap)

Output of the neural network is a *K* long vector

How should we encode the ground-truth values?

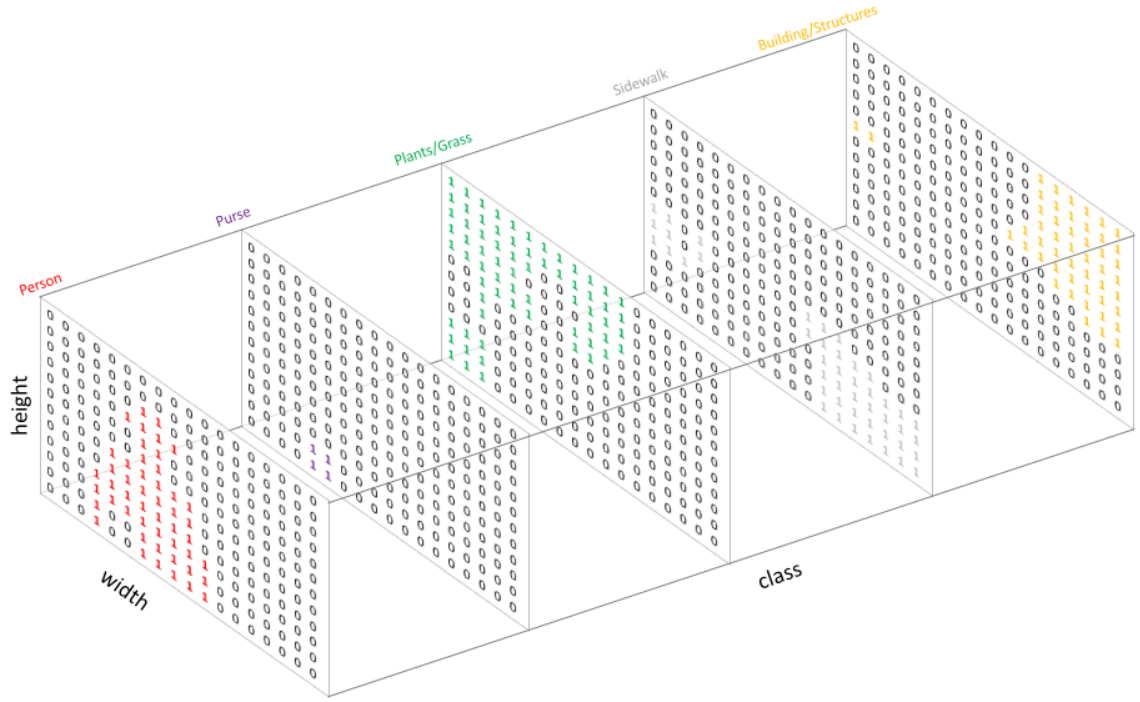
One-hot encoding (K=3):

(cat) 1 -> [1, 0, 0]
(dog) 2 -> [0, 1, 0]
(horse) 3 -> [0, 0, 1,]

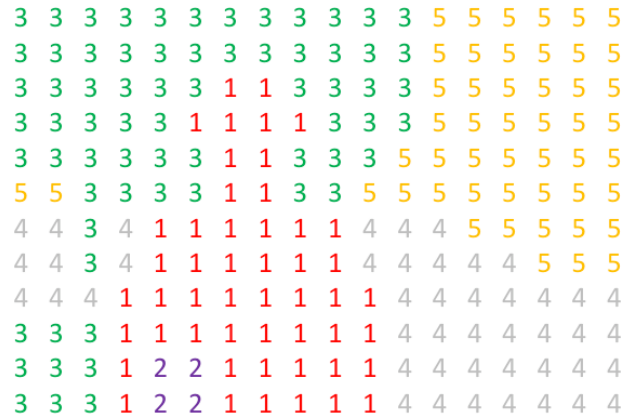
Just as h(x): values between 0 and 1, sum up to 1

Semantic Segmentation

Similar to how we treat standard categorical values, we'll create our **target** by one-hot encoding the class labels - essentially creating **an output channel for each of the possible classes**.



Depth-wise argmax

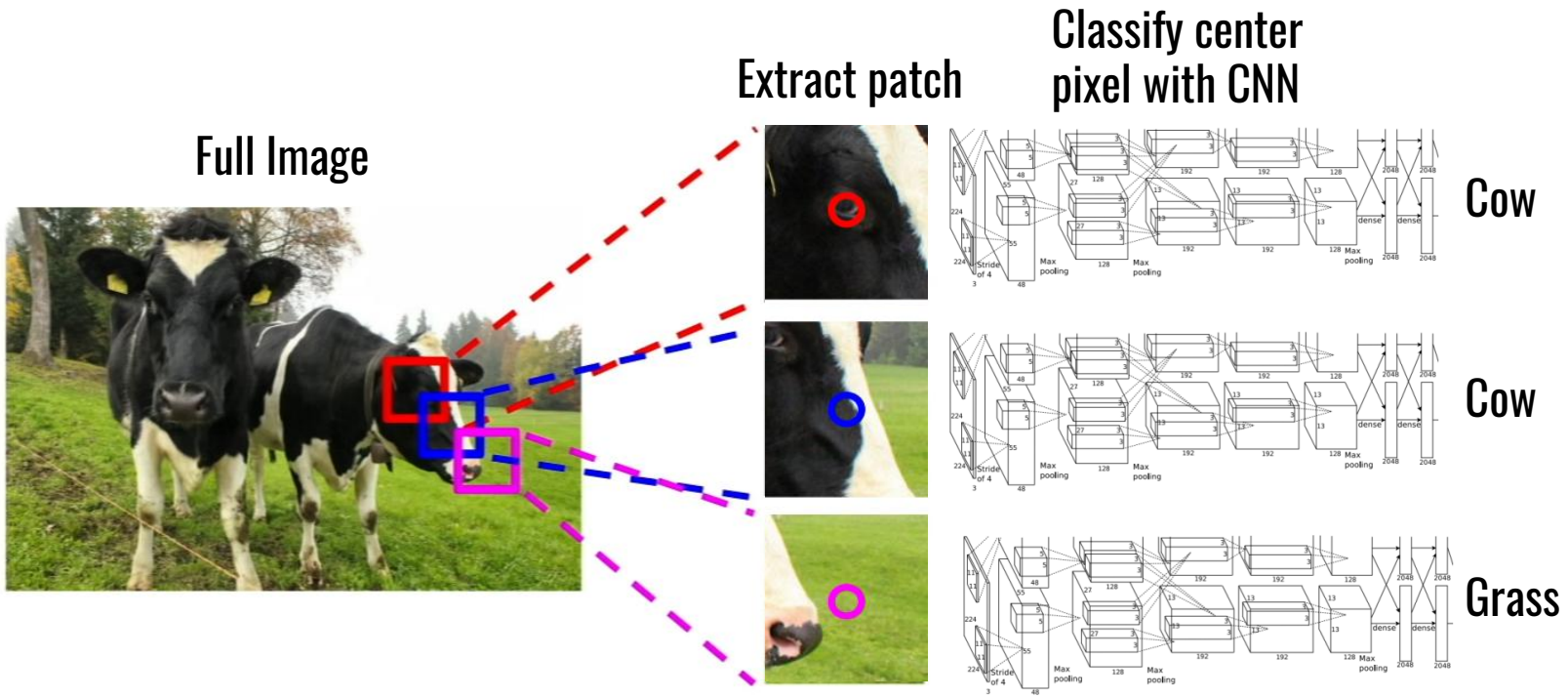


Semantic Labels

A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.

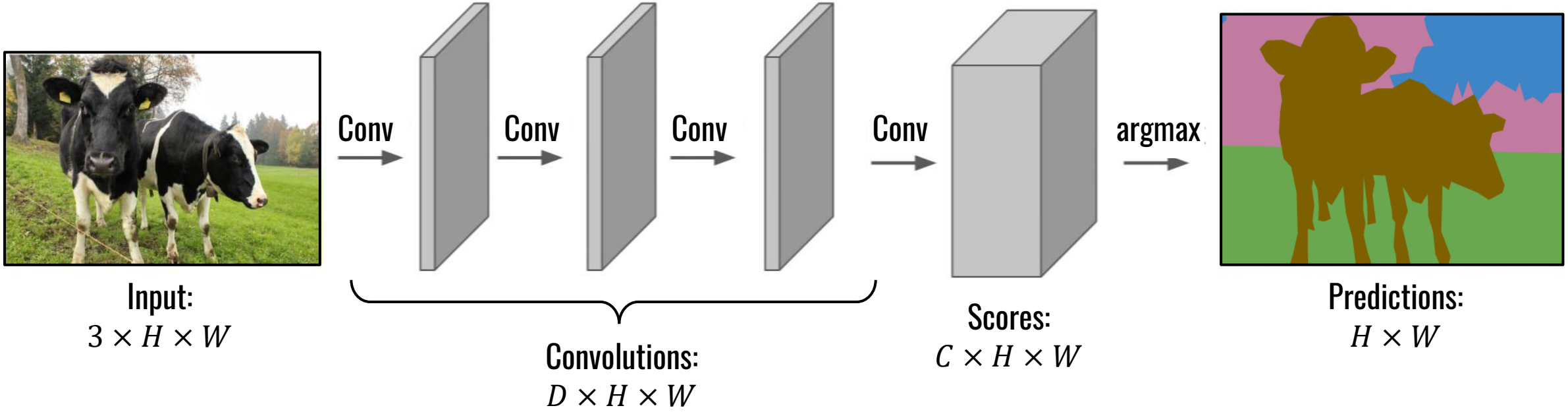
How to solve it? The naïve approach:

- Sliding Windows + Classification
  - Computationally expensive
  - Not reusing shared features



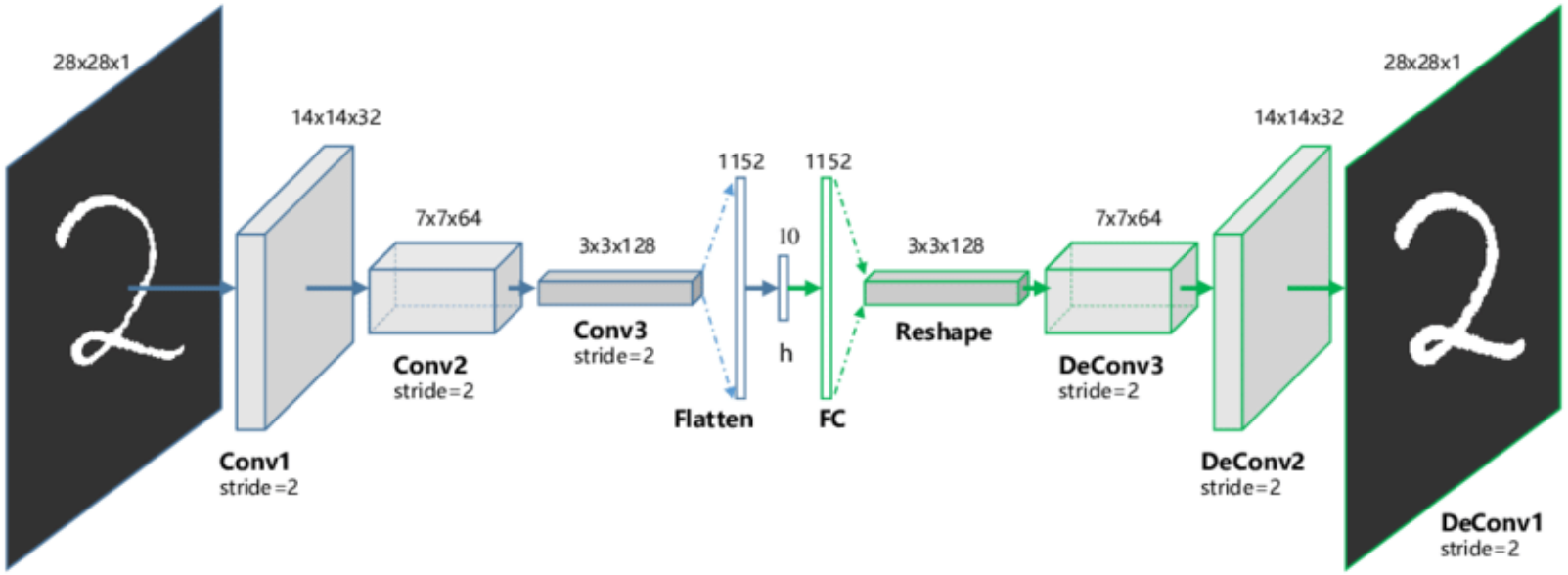
How to solve it? The naïve approach:

- Apply convolutions for all pixels at once, keeping original resolution
  - Computationally expensive
  - Does not enforce network to learn key features. It only learns a direct mapping from input pixels to the segmentation pixels.



Convolutional Autoencoders

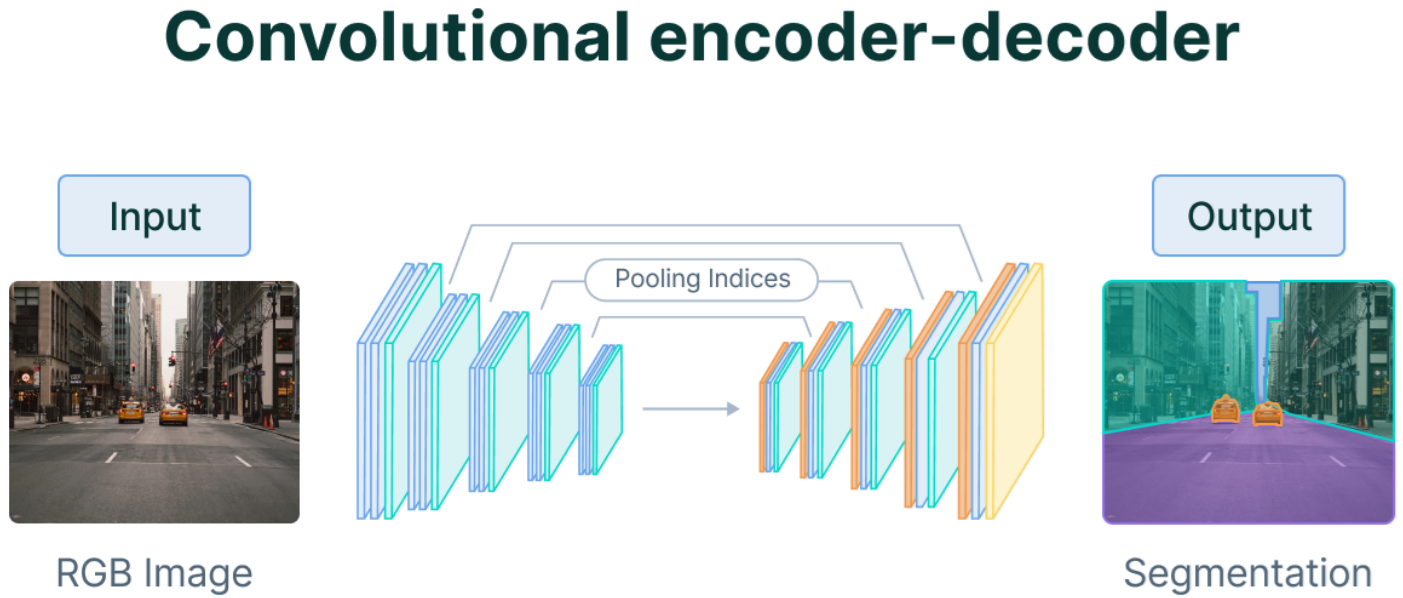
- Specifically designed for **image data**.
- They employ convolutional layers in both the encoder and decoder parts of the network.
- This architecture allows them to **capture spatial dependencies and hierarchical features** effectively.
- The reconstruction of the input image is often blurry and of lower quality due to compression during which information is lost.



Convolutional Autoencoders

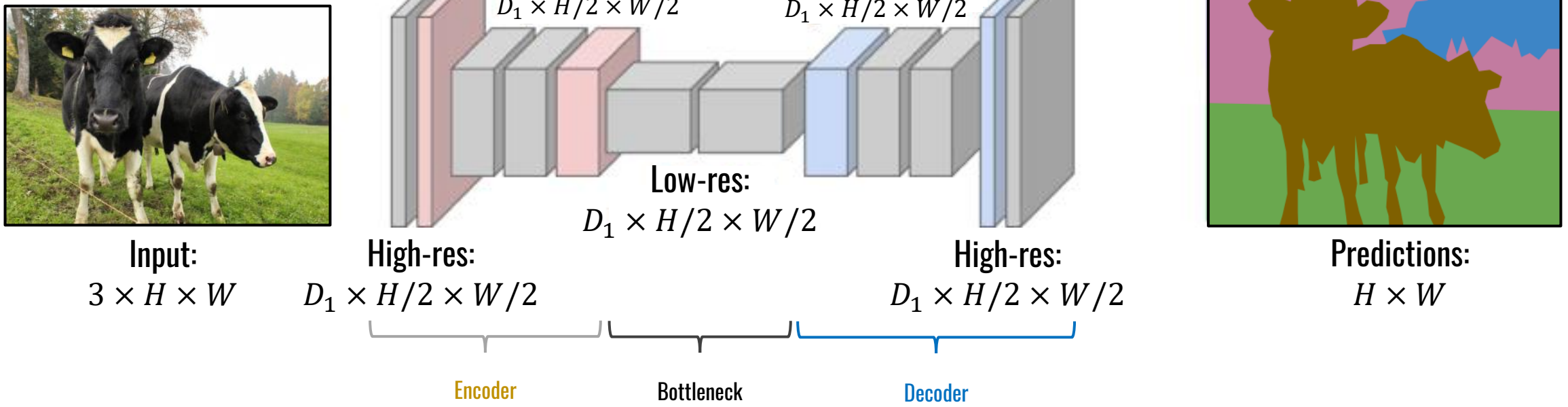
Image Segmentation

- Image segmentation is the process of **partitioning** an image **into multiple segments** each belonging to a class.
- The goal is to simplify and/or change the representation of an image by **grouping pixel values according to the class they belong to**.



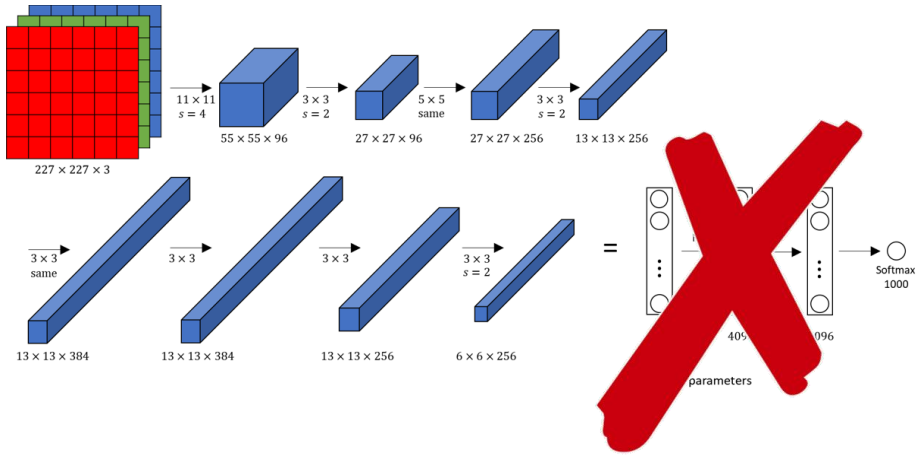
Encoder-Decoder Structure

- One popular approach for image segmentation models is to follow an **encoder**/**decoder** **structure** where we **downsample** the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and then **upsample** the feature representations into a full-resolution segmentation map.

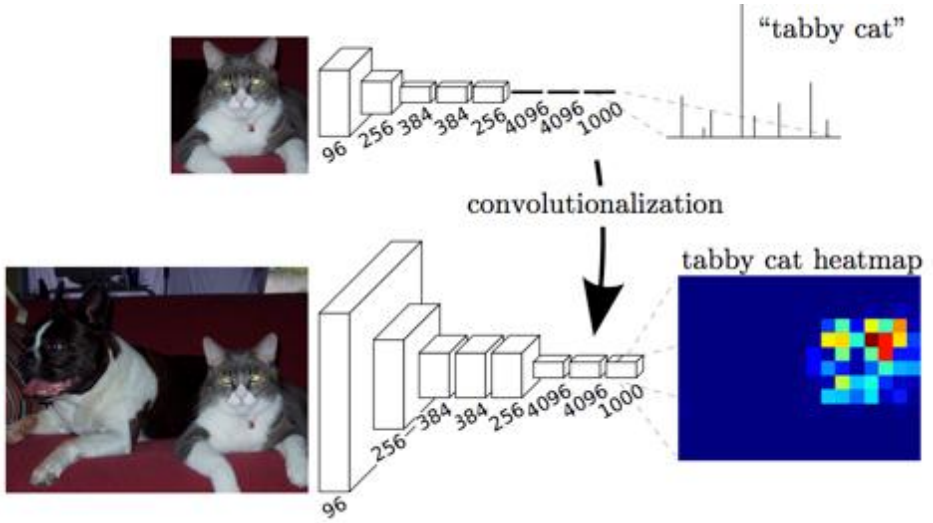


Fully Convolutional Network (FCN) [1]

The approach of using a *"Fully Convolutional"* network trained end-to-end, pixels-to-pixels for the task of image segmentation was introduced by Long et al. in late 2014. The paper's authors propose adapting existing, well-studied **image classification** networks (e.g. **AlexNet**) to serve as the encoder module of the network, appending a decoder module with transpose convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.



Original AlexNet model

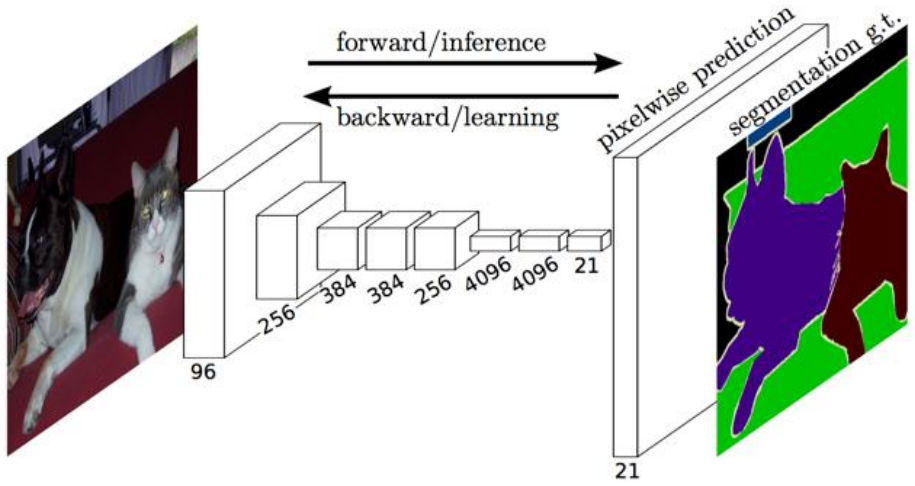


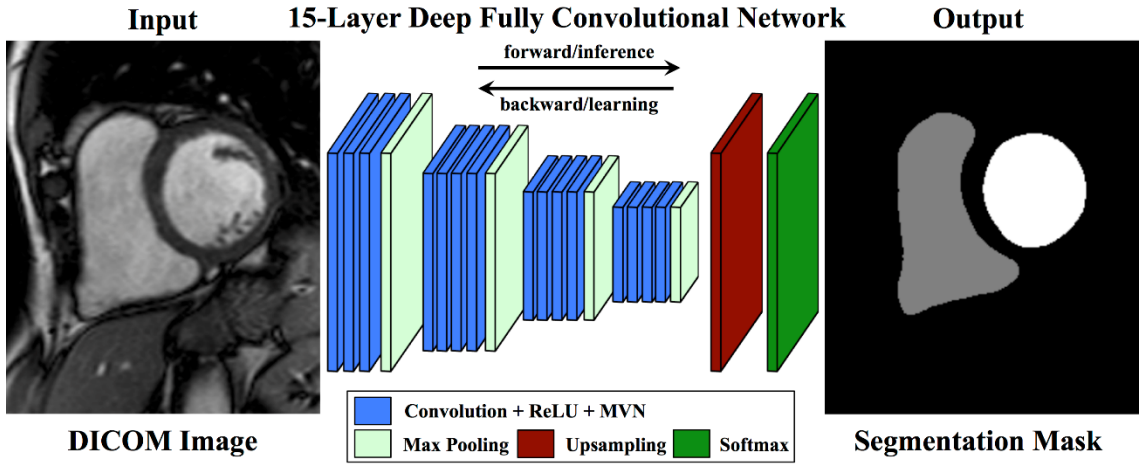
Repurposed AlexNet model

The encoder produces a coarse feature map which is then refined by the decoder module.

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Fully Convolutional Network (FCN) [1]





However, because the encoder module reduces the resolution of the input by a factor of 32, the decoder module **struggles to produce fine-grained segmentations**.

Ground truth target

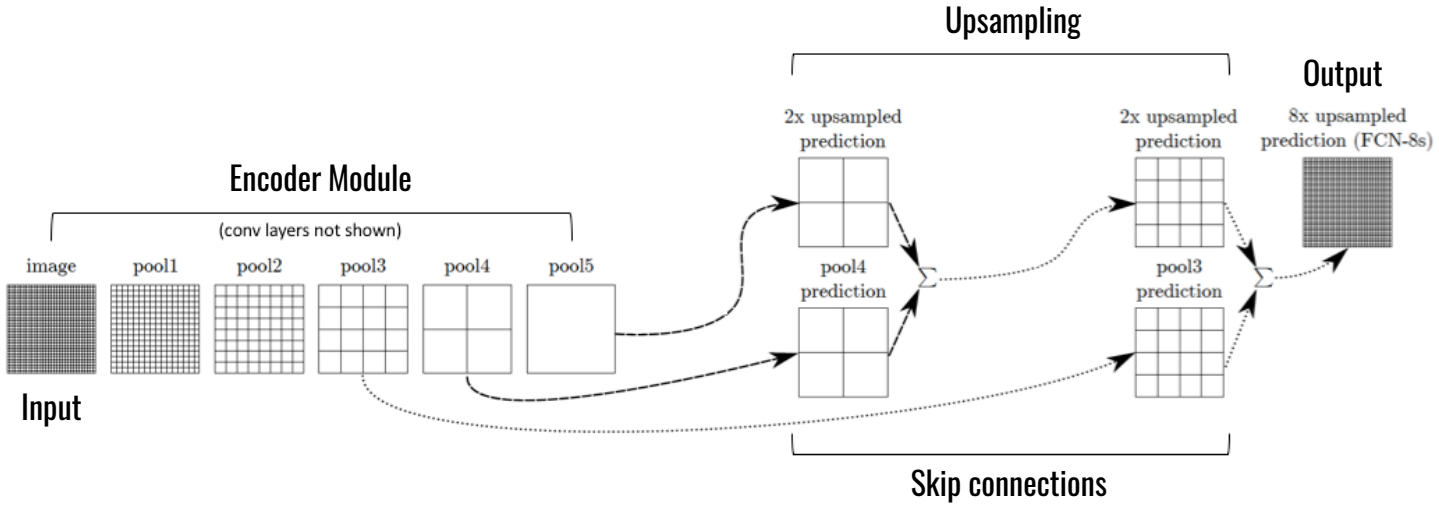
Predicted segmentation

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Fully Convolutional Network (FCN) [1]

- **Adding skip connections** - The authors address this tension by slowly upsampling (in stages) the encoded representation, adding "skip connections" from earlier layers, and summing these two feature maps.

- These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail to reconstruct accurate shapes for segmentation boundaries. Indeed, we can recover more fine-grain detail with the addition of these skip connections.



Before

Ground truth target

Predicted segmentation

with skip connections

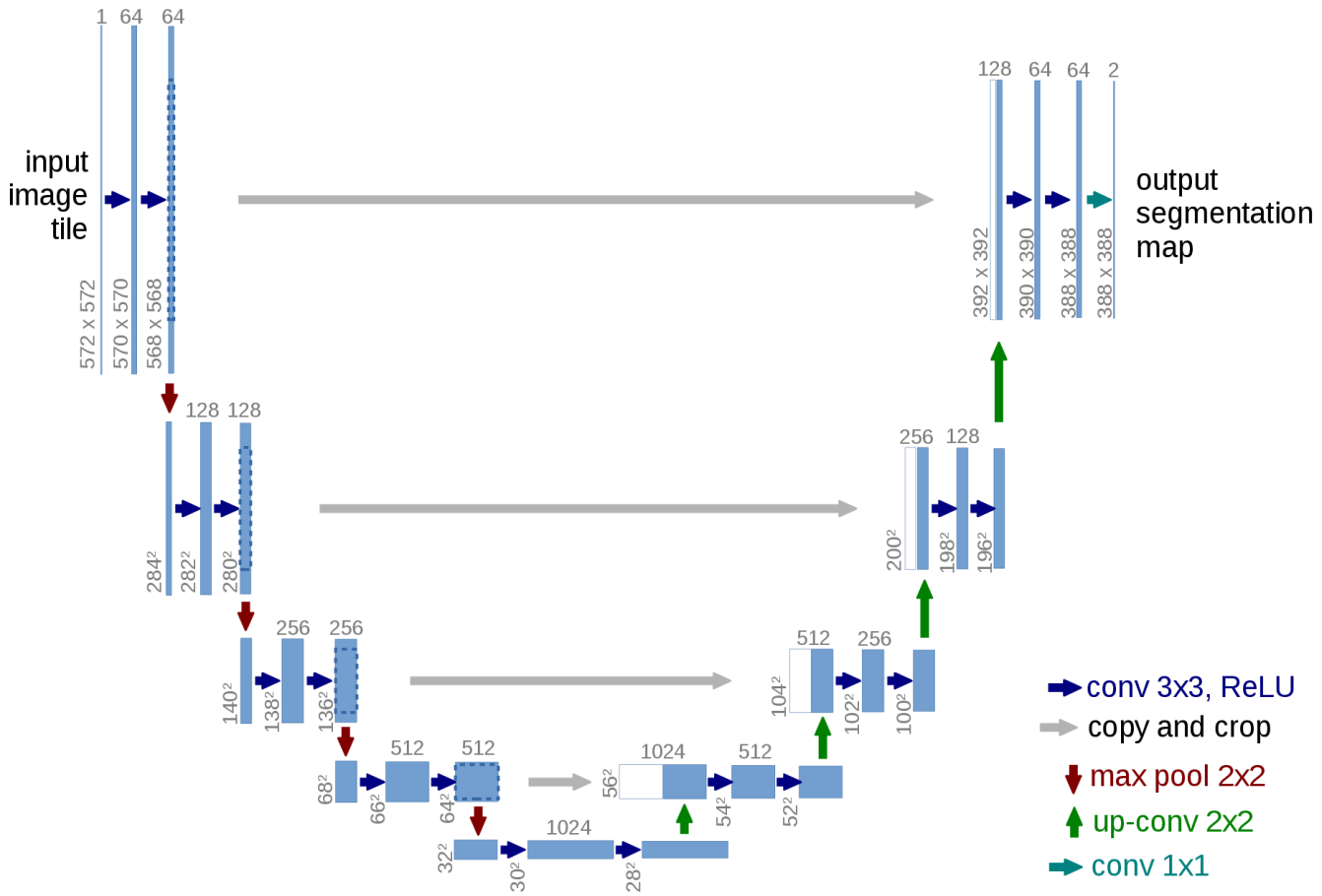
Ground truth target

Predicted segmentation

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

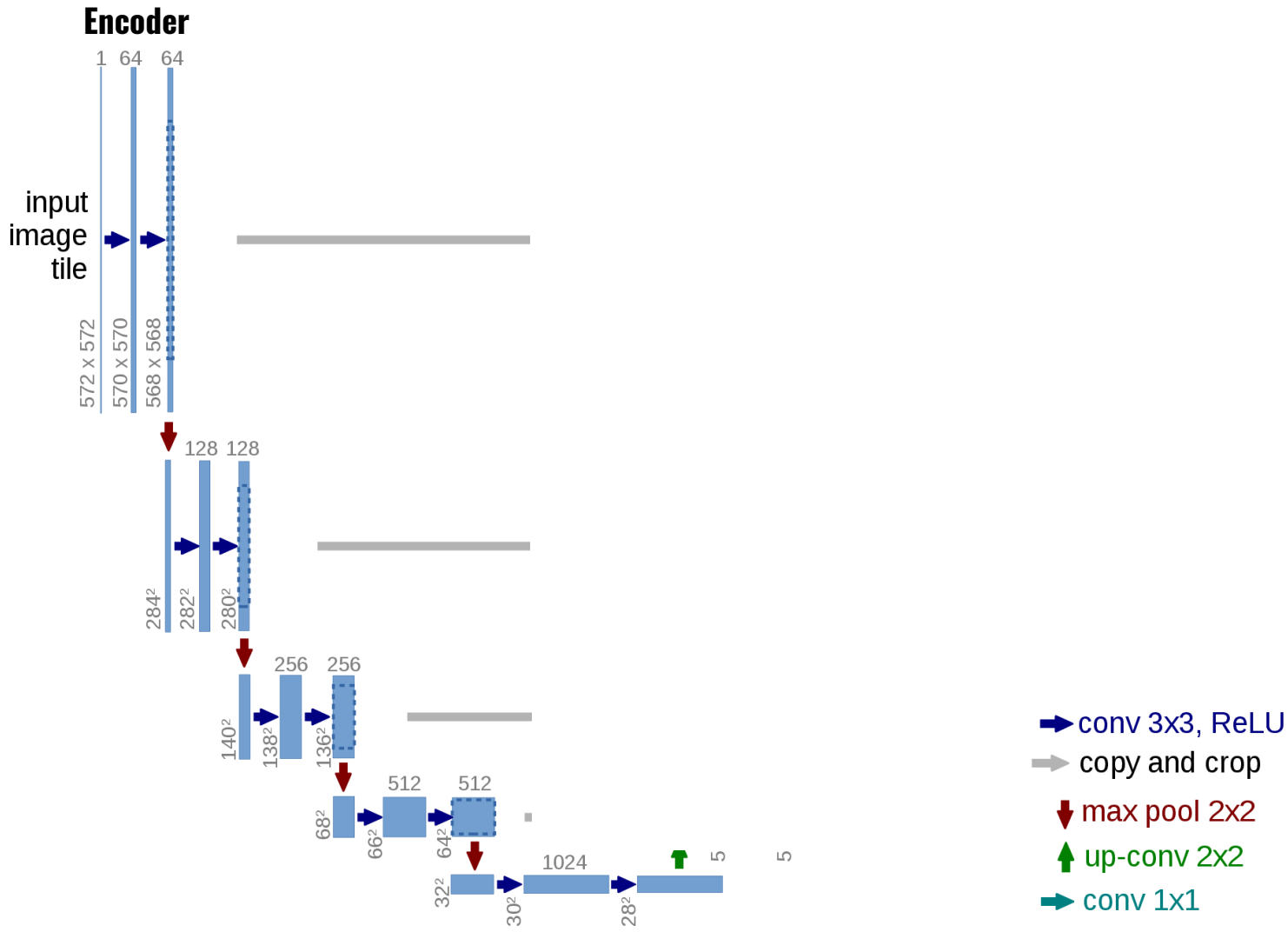
U-Net: Convolutional Networks for Biomedical Image Segmentation [2]

- Ronneberger et al. improve upon the "fully convolutional" architecture primarily through ***expanding the capacity of the decoder*** module of the network. More concretely, they propose the **U-Net architecture** which "consists of a contracting path to capture context and a ***symmetric*** expanding path that enables precise localization."

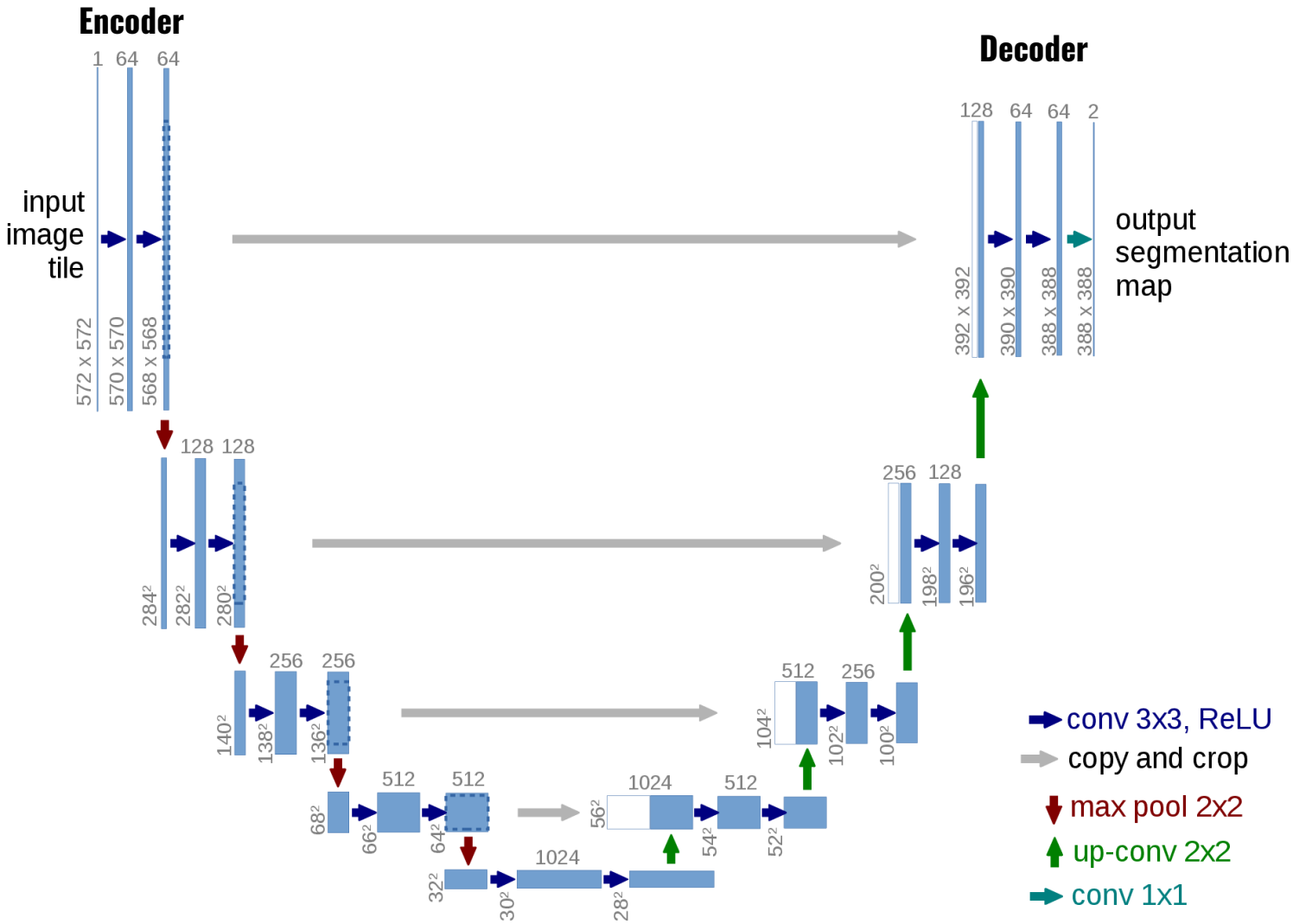


[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer International Publishing, 2015.

U-Net


Encoder
1 64 64
input image tile
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
32²
30²
28²
5
5
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

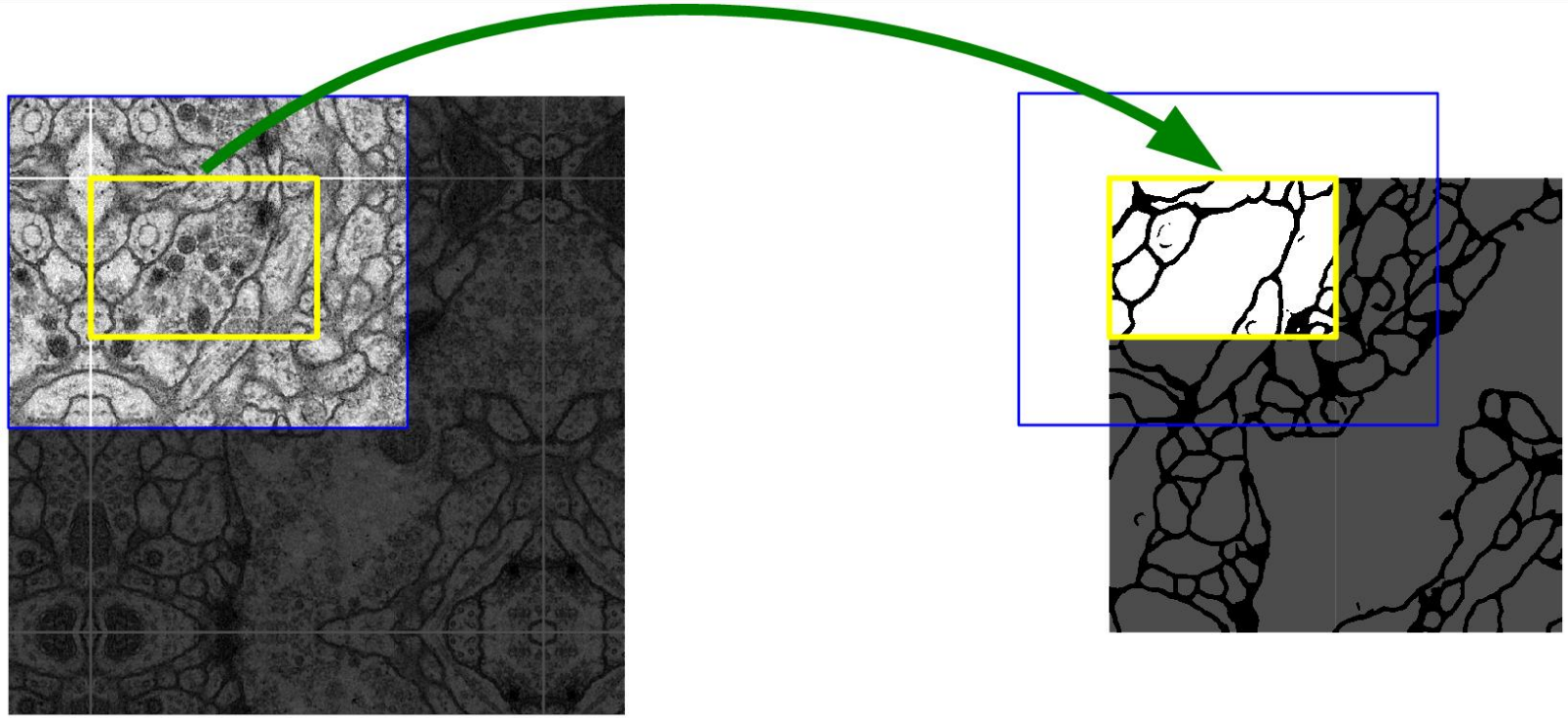
U-Net


Encoder
Decoder
input image tile
output segmentation map
1 64 64
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
32²
30²
28²
1024
512
56²
54²
52²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

U-Net

How can we use it for images with arbitrary size?

- Do the segmentation for smaller regions of the image
- On the edges mirror the image



Architectures

Advanced U-Net variants

- The standard U-Net model consists of a series of convolution operations for each "block" in the architecture. These convolutional blocks can be replaced for more advanced ones such as:
  - ResNet blocks
  - Inception modules
  - Dense blocks
  - Etc.

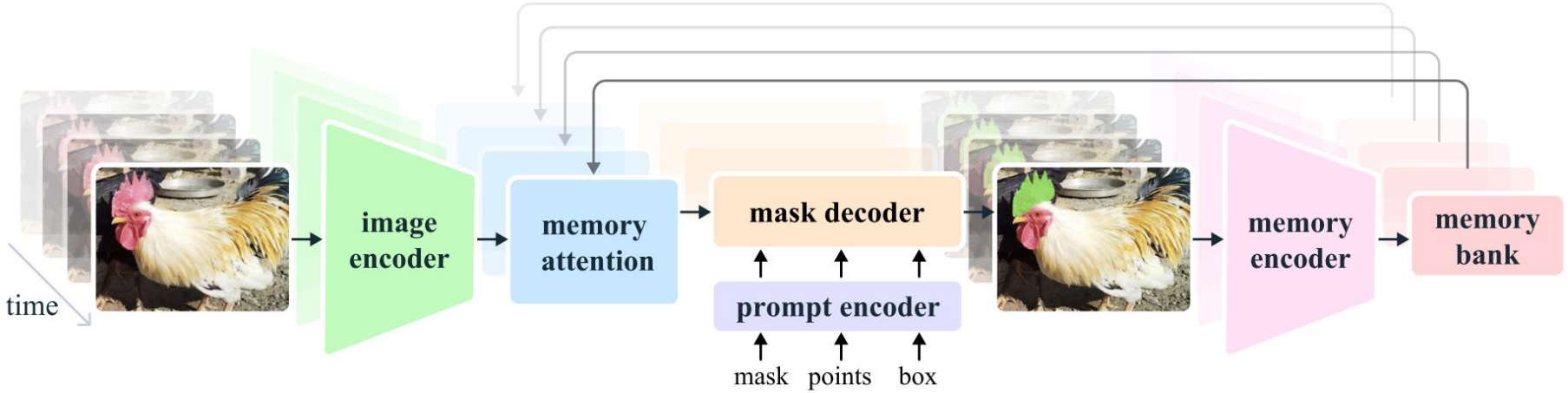
DeepLab

- Deeplab from a group of researchers from Google have proposed a multitude of techniques to improve the existing results and get finer output at lower computational costs. The 3 main improvements suggested as part of the research are:
  - Atrous convolutions
  - Atrous Spatial Pyramidal Pooling
  - Conditional Random Fields usage for improving final output

DeepLab v3: https://arxiv.org/abs/1706.05587v3

Segment Anything Model (SAM 2)

- Extending SAM with memory to keep the masks over the whole video
- Images are considered as a single frame video
- iVOS/VOS
- https://sam2.metademolab.com/demo

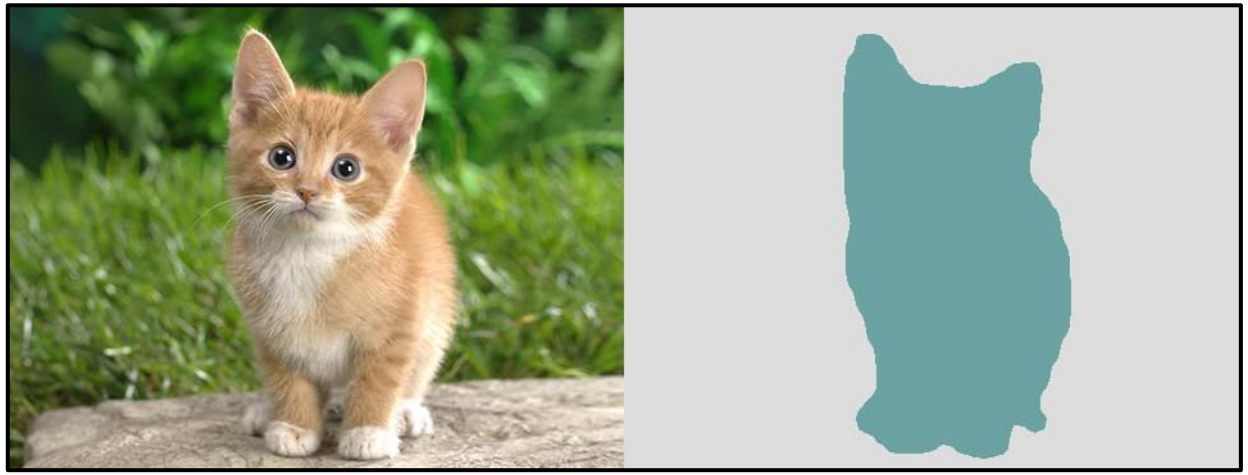


[7] Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., ... Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2408.00714

Training

How to train such networks?

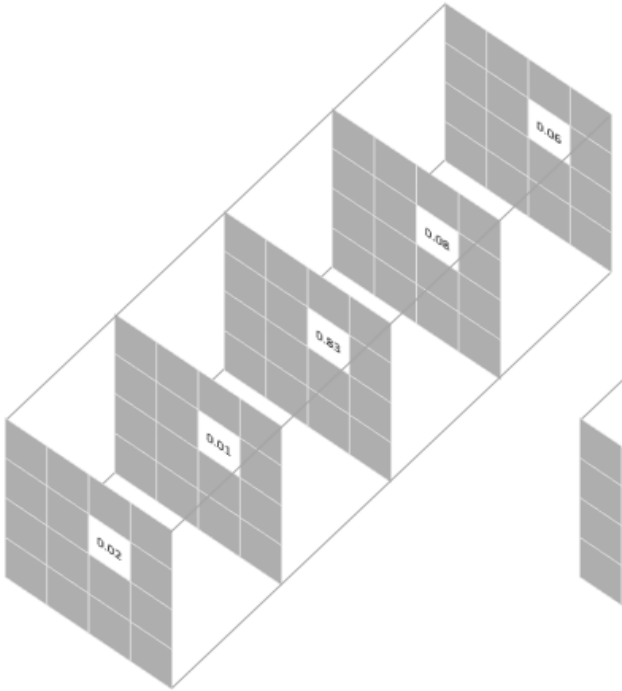
- Have input X (images) and labels Y (masks).
- Define architecture
- Set hyperparameters
- Define loss function and metrics



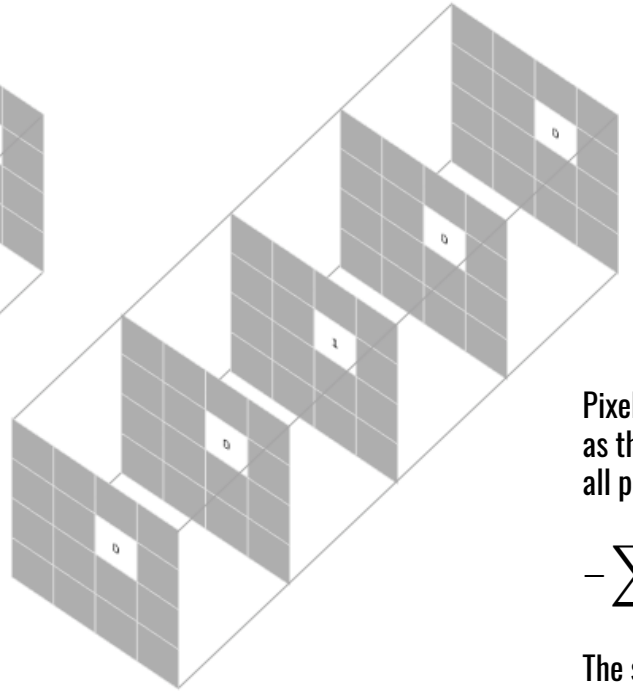
Losses

Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.
- Problematic for unbalanced classes



Prediction for a selected pixel



Target for the corresponding pixel

Pixel-wise loss is calculated as the log loss, summed over all possible classes

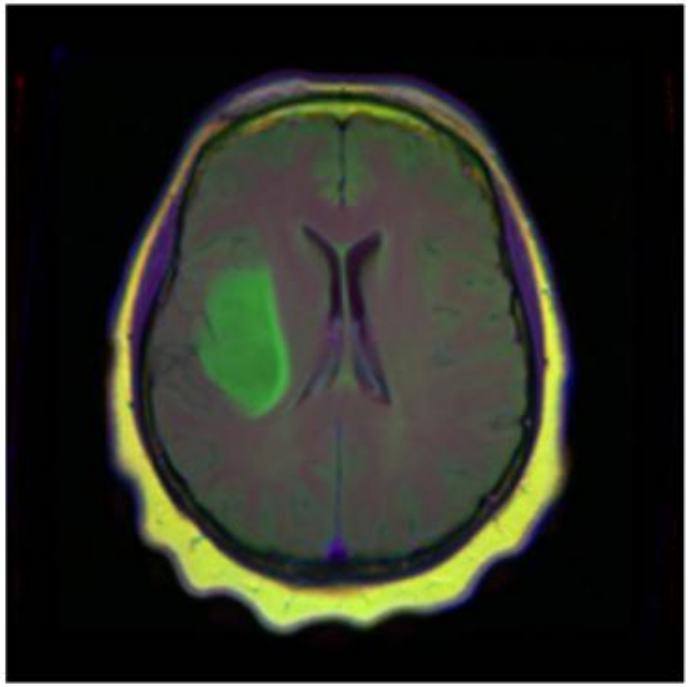
$$-\sum_{classes} y_{true} \log(y_{pred})$$

The scoring is repeated over all pixels and averaged

Losses

Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.
- Problematic for unbalanced classes



Losses

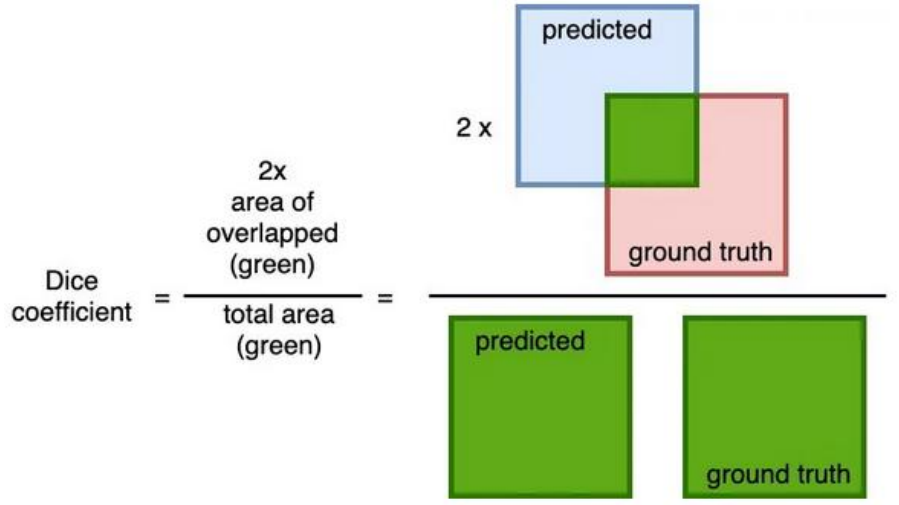
Dice Loss

- Another popular loss function for image segmentation tasks is based on the Dice coefficient, which is essentially a measure of overlap between two samples. This measure ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap. The Dice coefficient was originally developed for binary data, and can be calculated as:

$$Dice = 2 \times \frac{|A \cap B|}{|A| + |B|}$$

where:

- $|A \cap B|$ represents the common elements between sets A and B, and
- $|A|$ represents the number of elements in set A
- $|B|$ represents the number of elements in set B



Deep Network Development

Lecture 8.

Instance Segmentation

Budapest, 21st March 2025

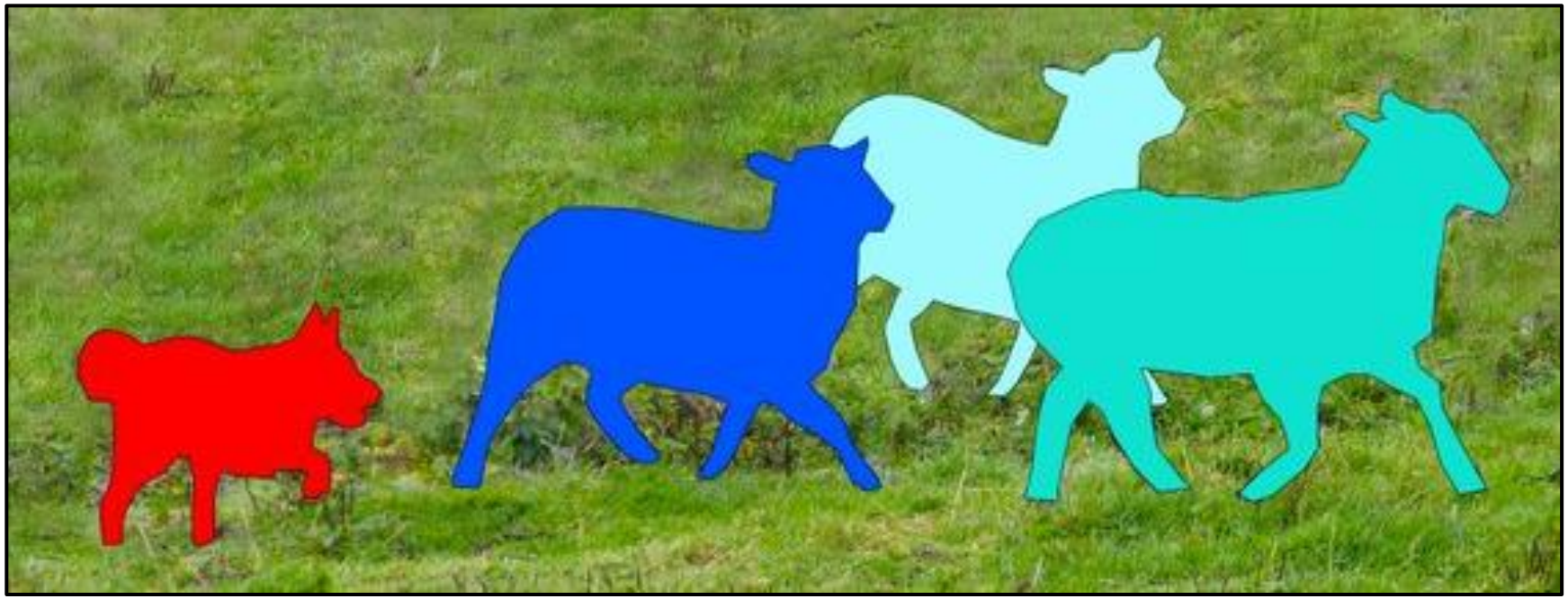
1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

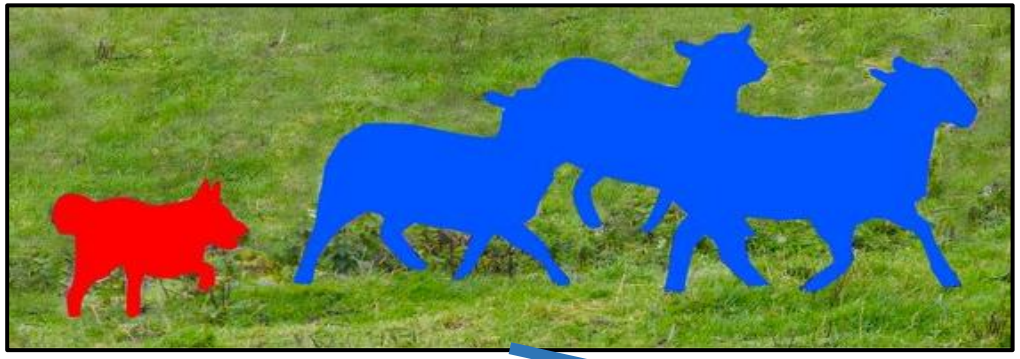
What is Instance Segmentation?

- Instance Segmentation is identifying each object instance for every known object within an image. It assigns a label to each pixel of the image.

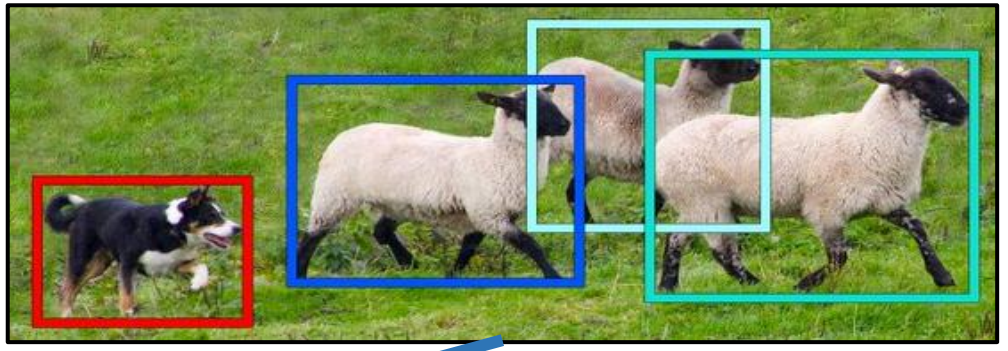


What is Instance Segmentation?

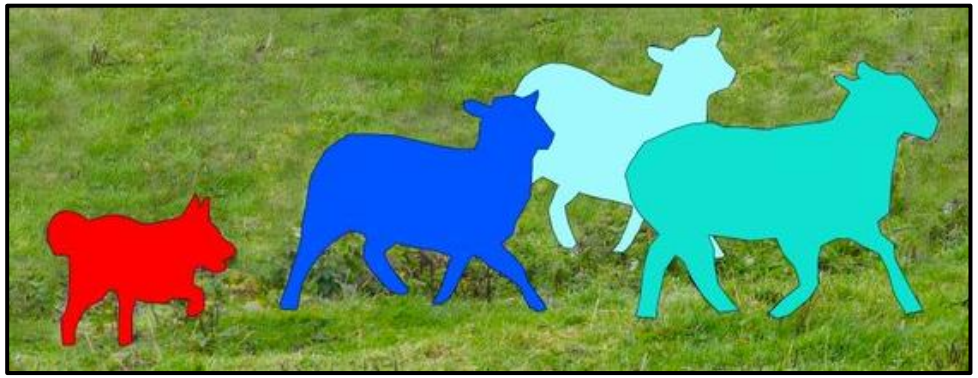
Semantic Segmentation: gives per-pixel labels, but merges instances.



Object Detection: detects individual object instances, but only gives boxes.



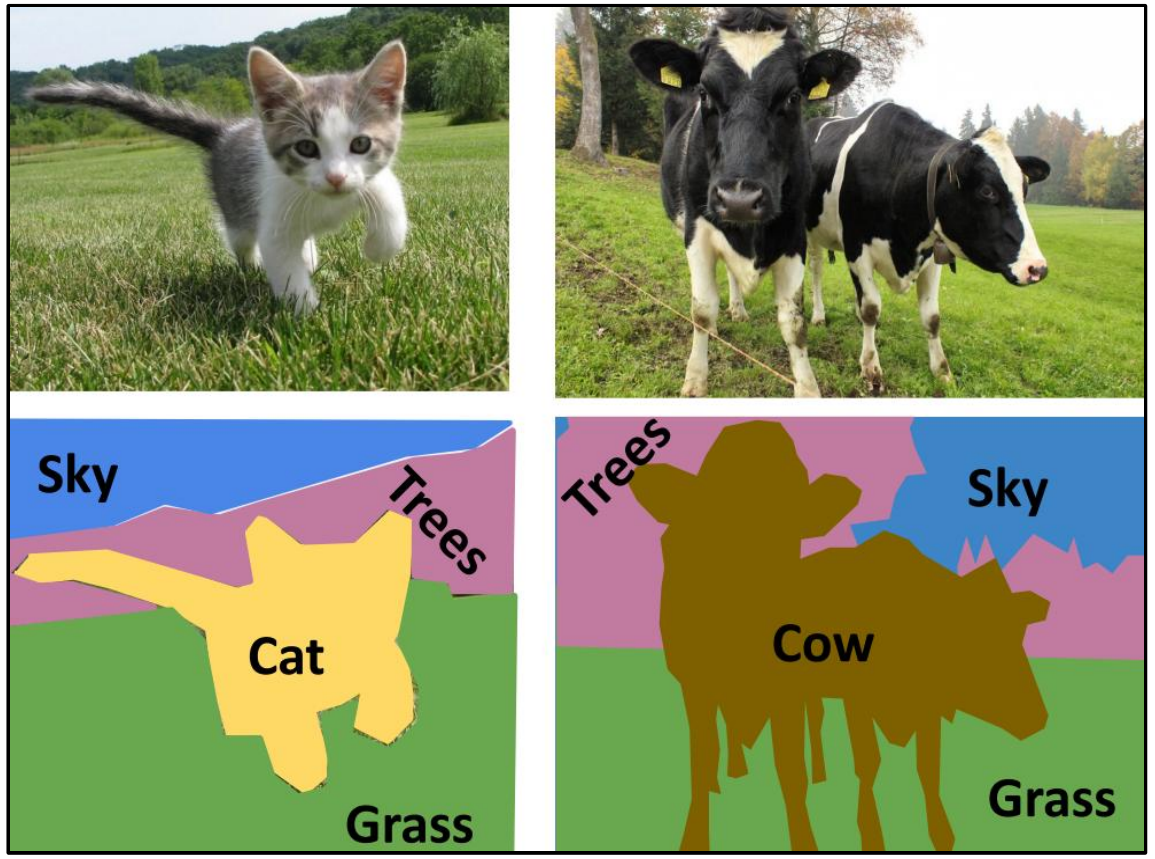
Instance Segmentation



What is Instance Segmentation?

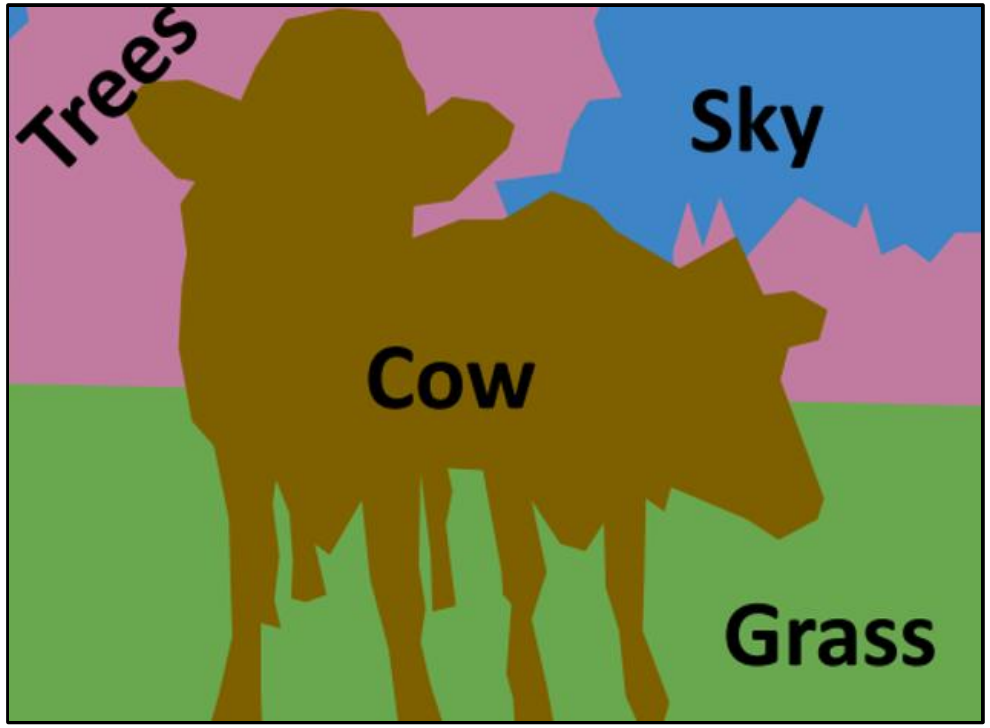
Things and Stuff

- **Things:** Object categories that can be separated into object instances (e.g. cats, cars, person)
- **Stuff:** Object categories that cannot be separated into instances (sky, grass, water, trees)

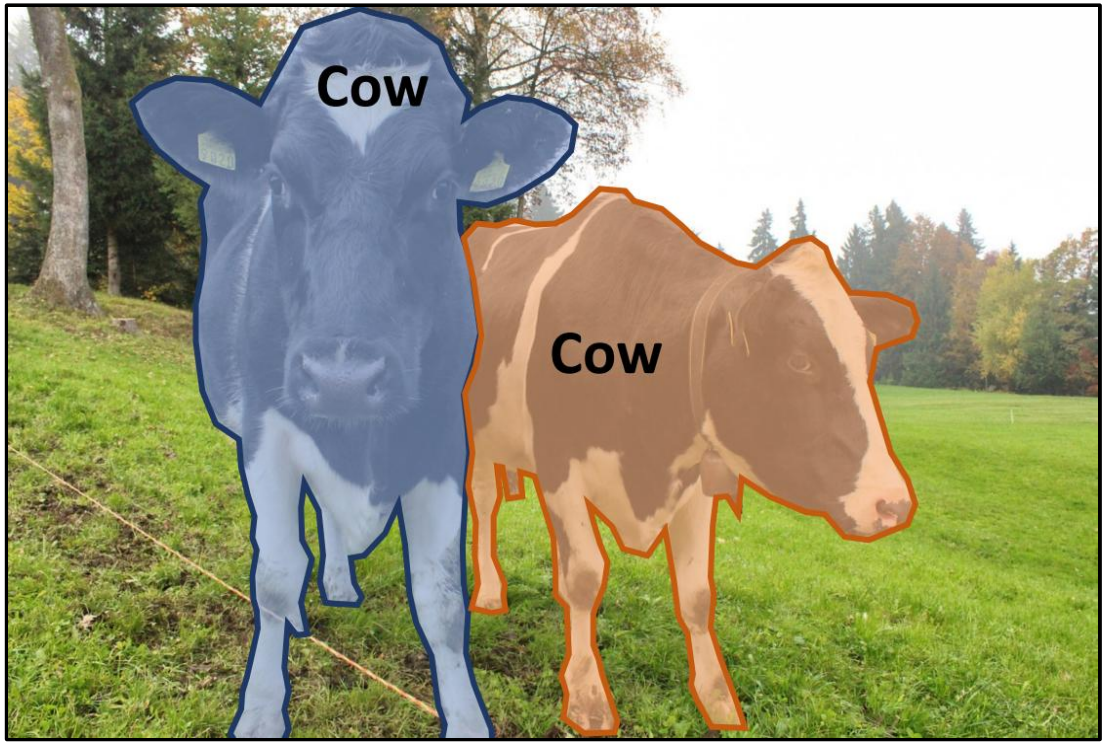


What is Instance Segmentation?

Semantic Segmentation: detects both objects and regions but doesn't distinguish individual instances.



Instance Segmentation: distinguishes individual object instances, but only for countable objects.

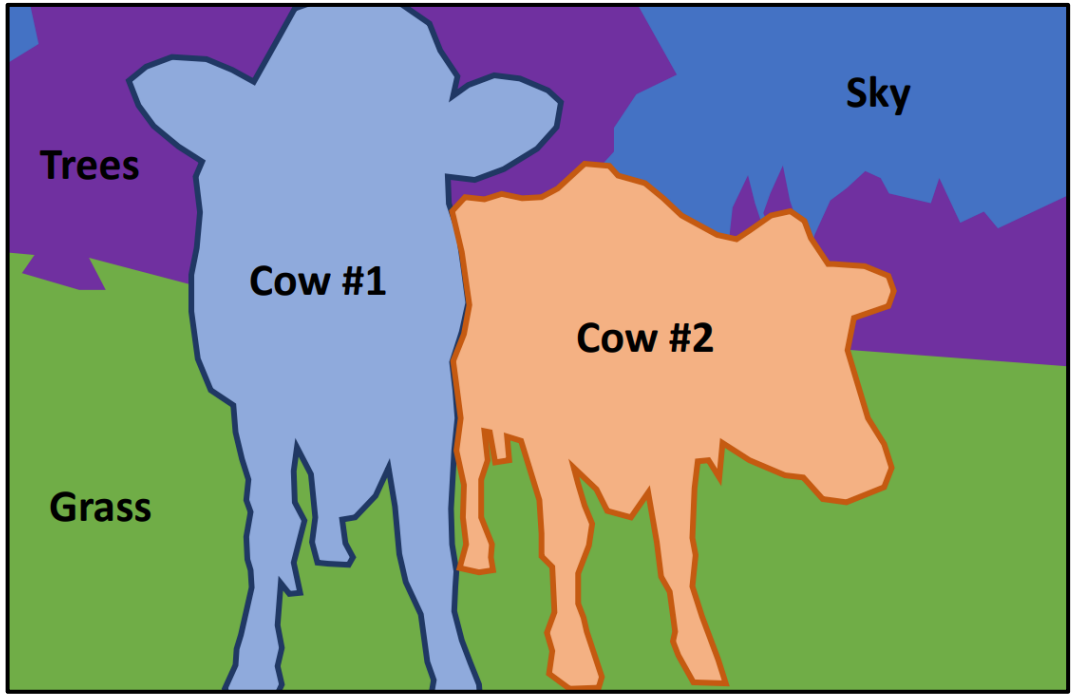


What is Instance Segmentation?

Beyond Instance Segmentation: Panoptic Segmentation [3]

Label all pixels in the image (both things and stuff).

For “thing” categories also separate into instances.



[3] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.00868

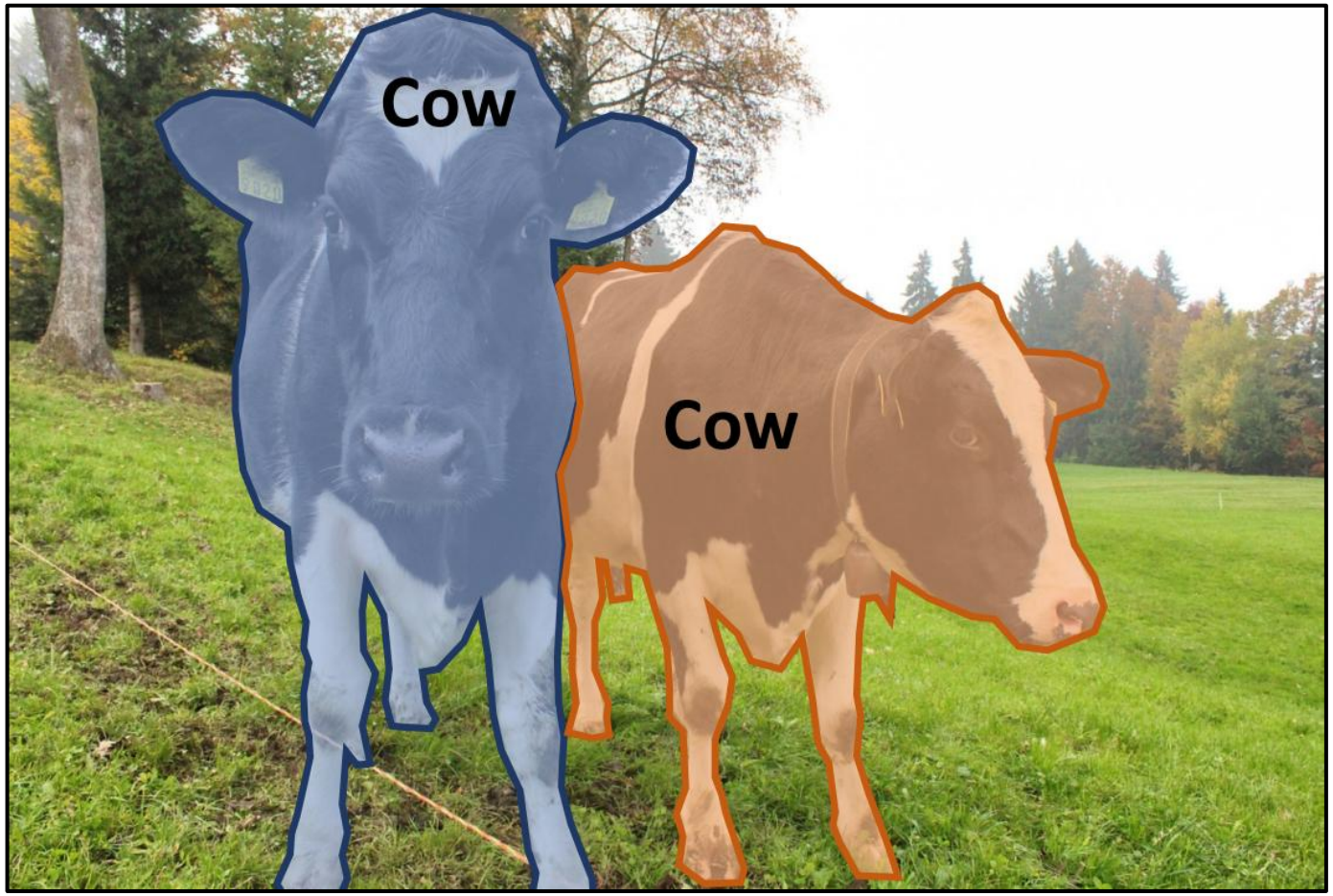
How does Instance Segmentation work?

Instance Segmentation:

- Detect all objects in the image and identify the pixels that belong to each object (Only things!)

Approach:

- Perform object detection, then predict a segmentation mask for each object!

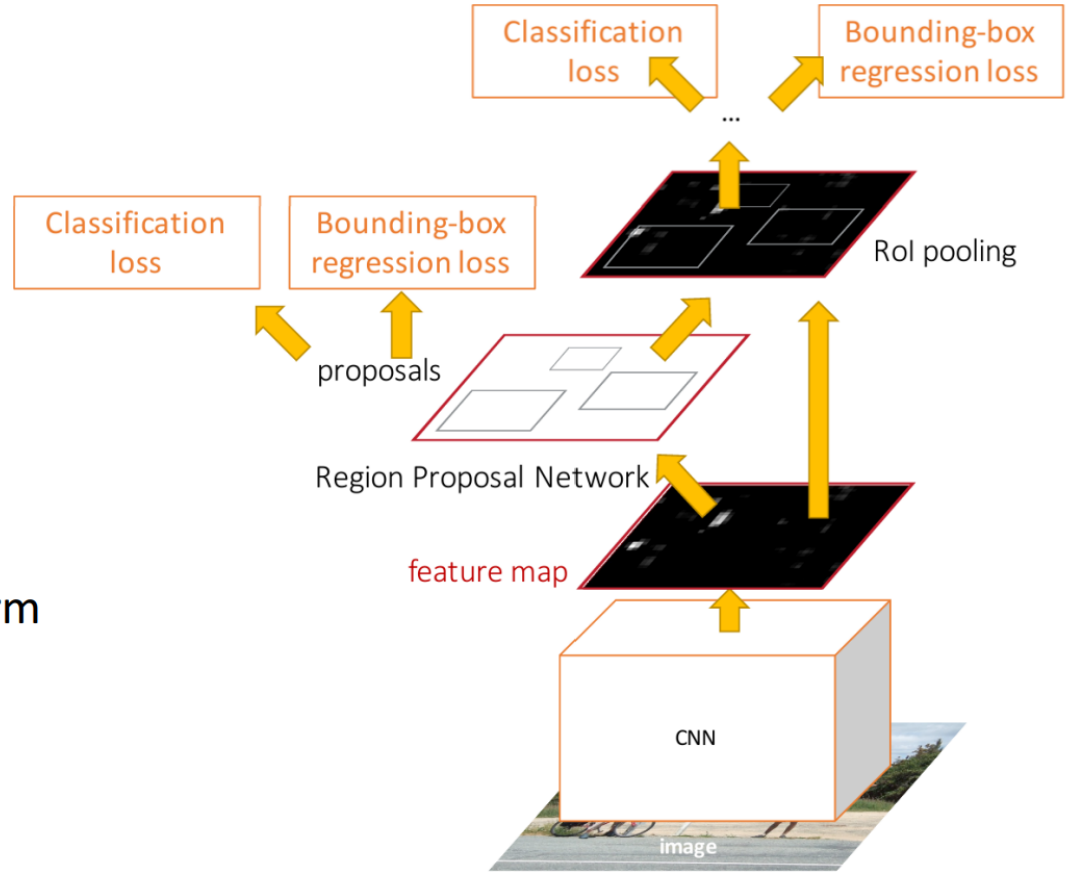


How does Instance Segmentation work?

Faster R-CNN: Learnable Region Proposals [4]

Jointly train with 4 losses:

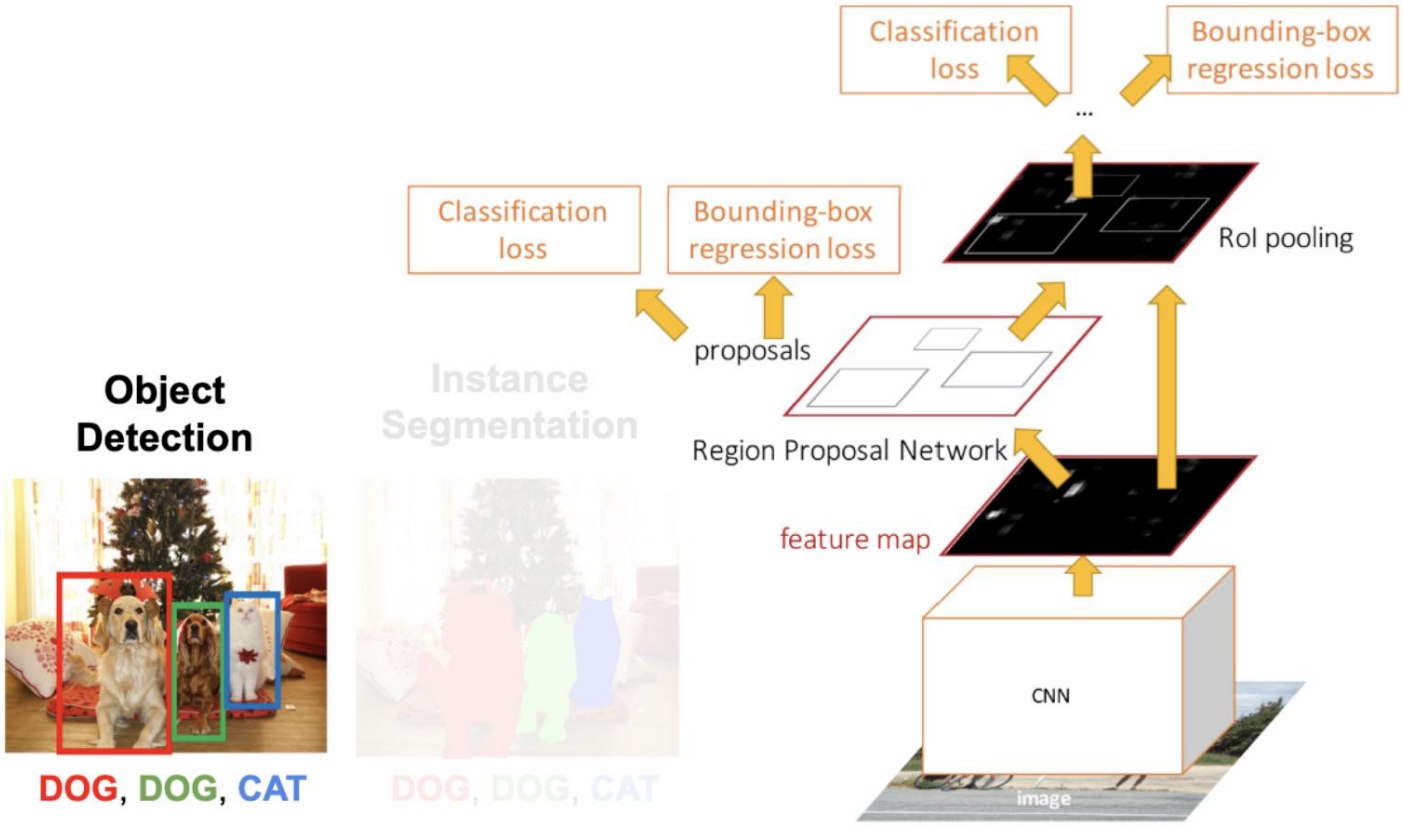
1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



[4] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

How does Instance Segmentation work?

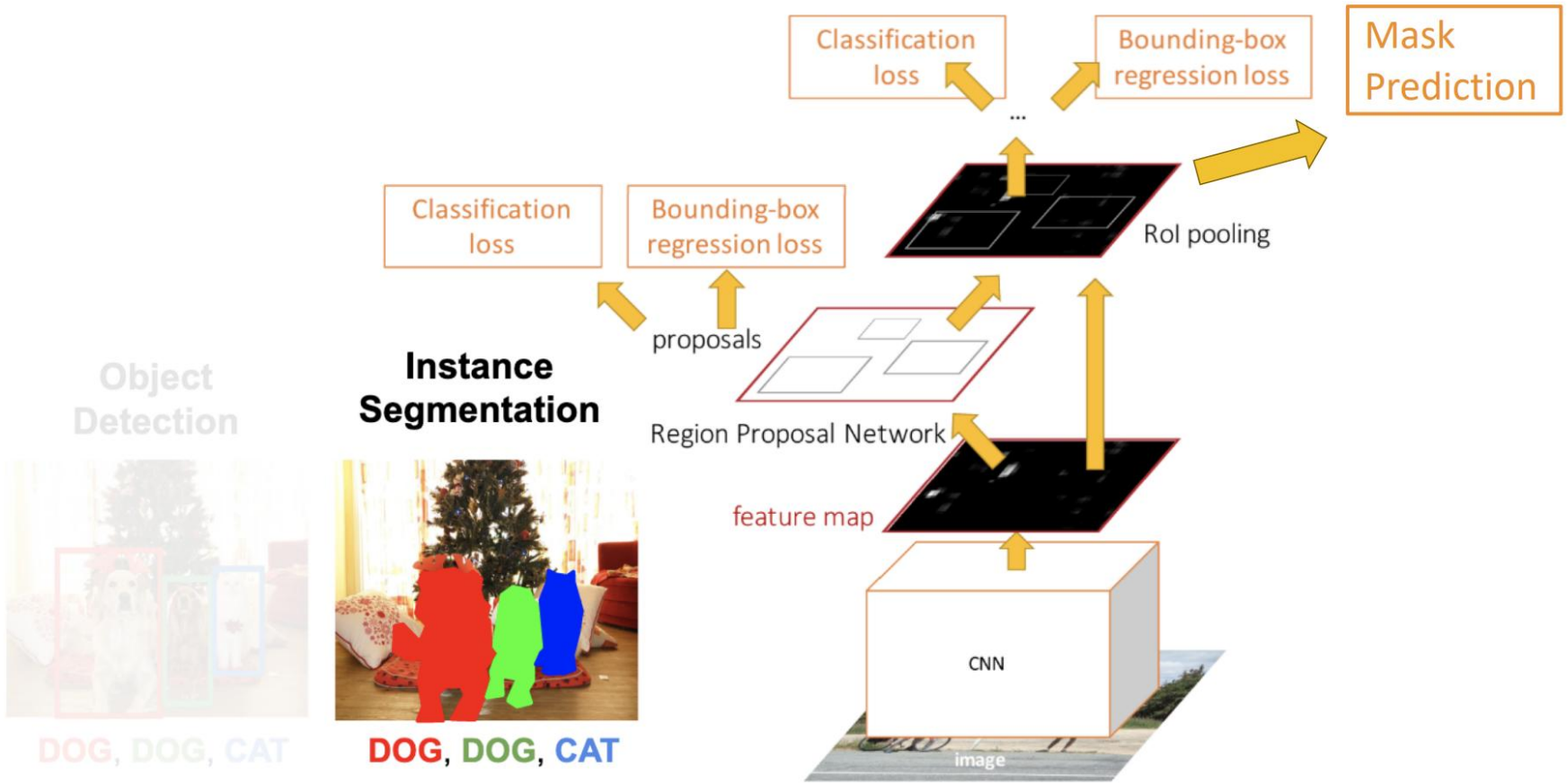
Object Detection: Faster R-CNN [4]



[4] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

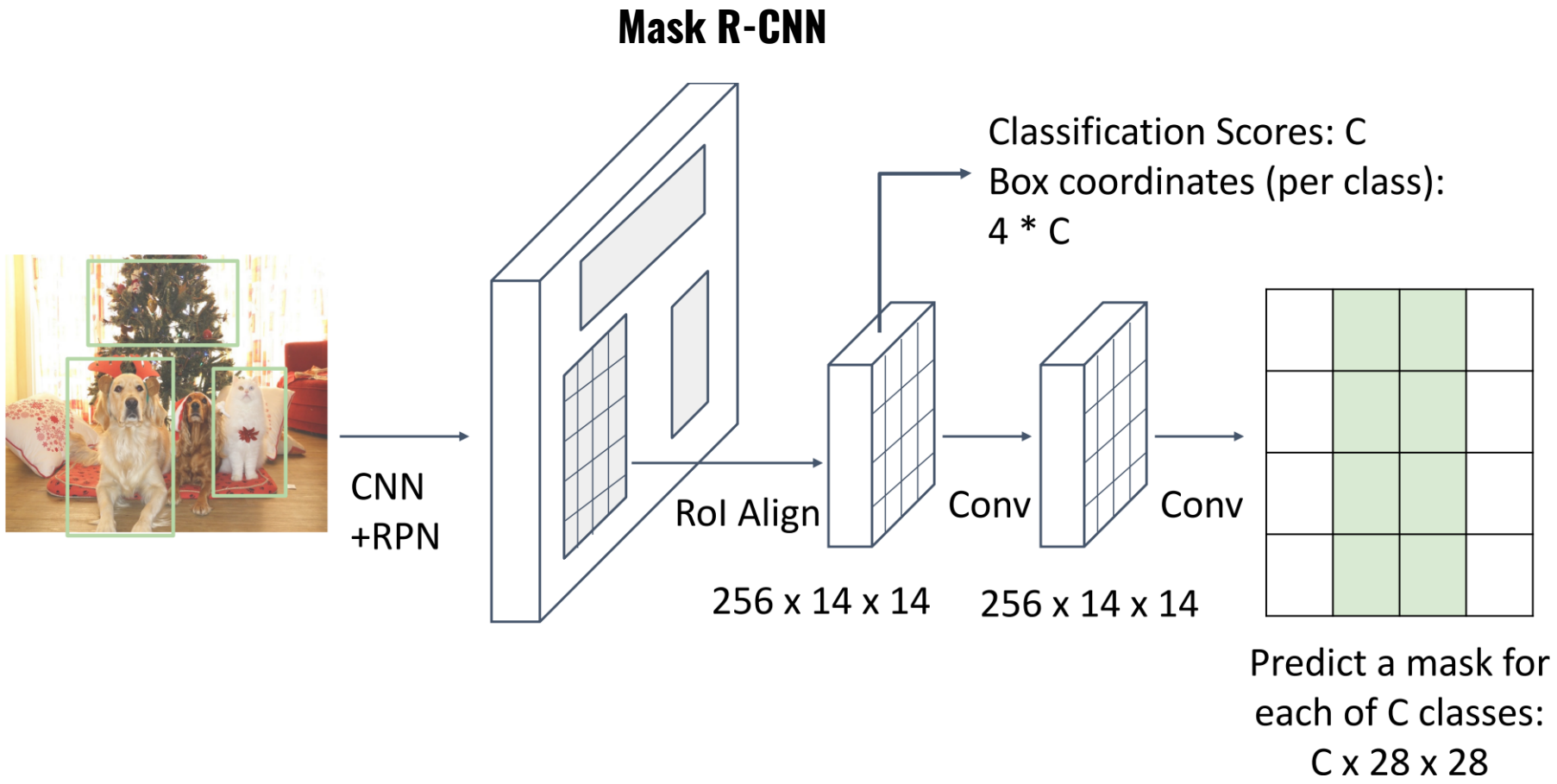
How does Instance Segmentation work?

Instance Segmentation: Mask R-CNN [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

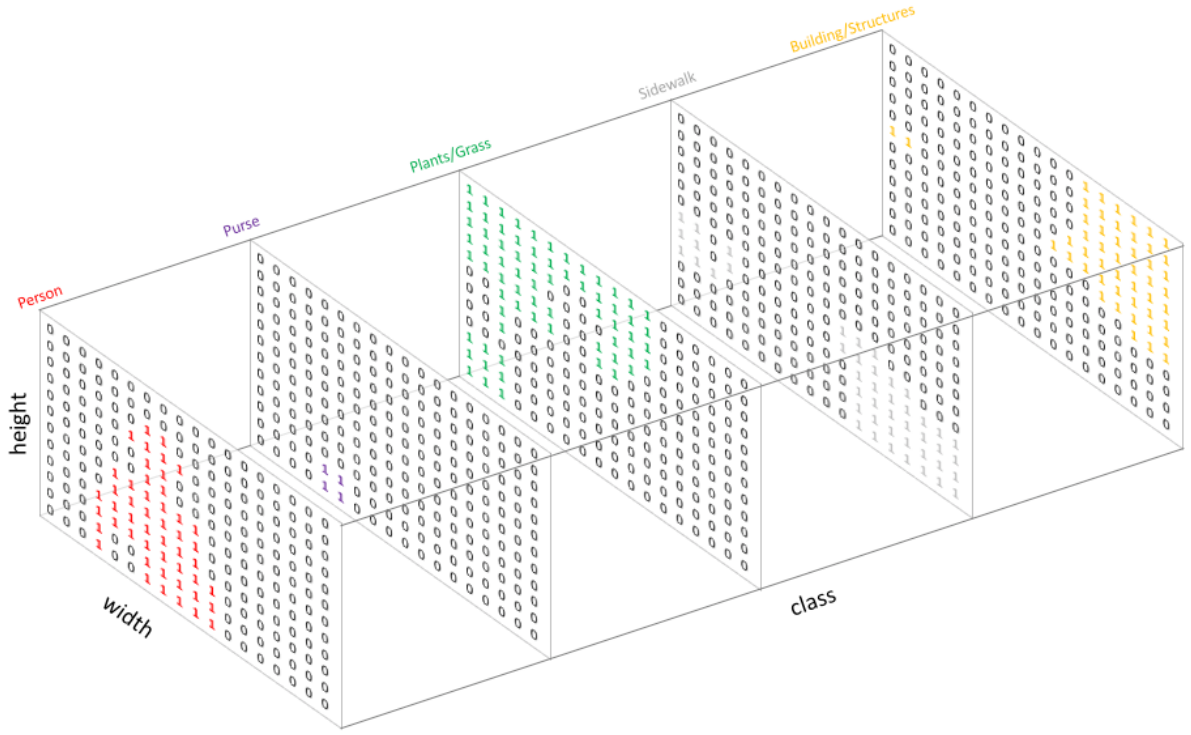
Mask R-CNN architecture (2017) [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Segmentation labels (recap)

- Similar to how we treat standard categorical values, we'll create our target by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.
- A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.



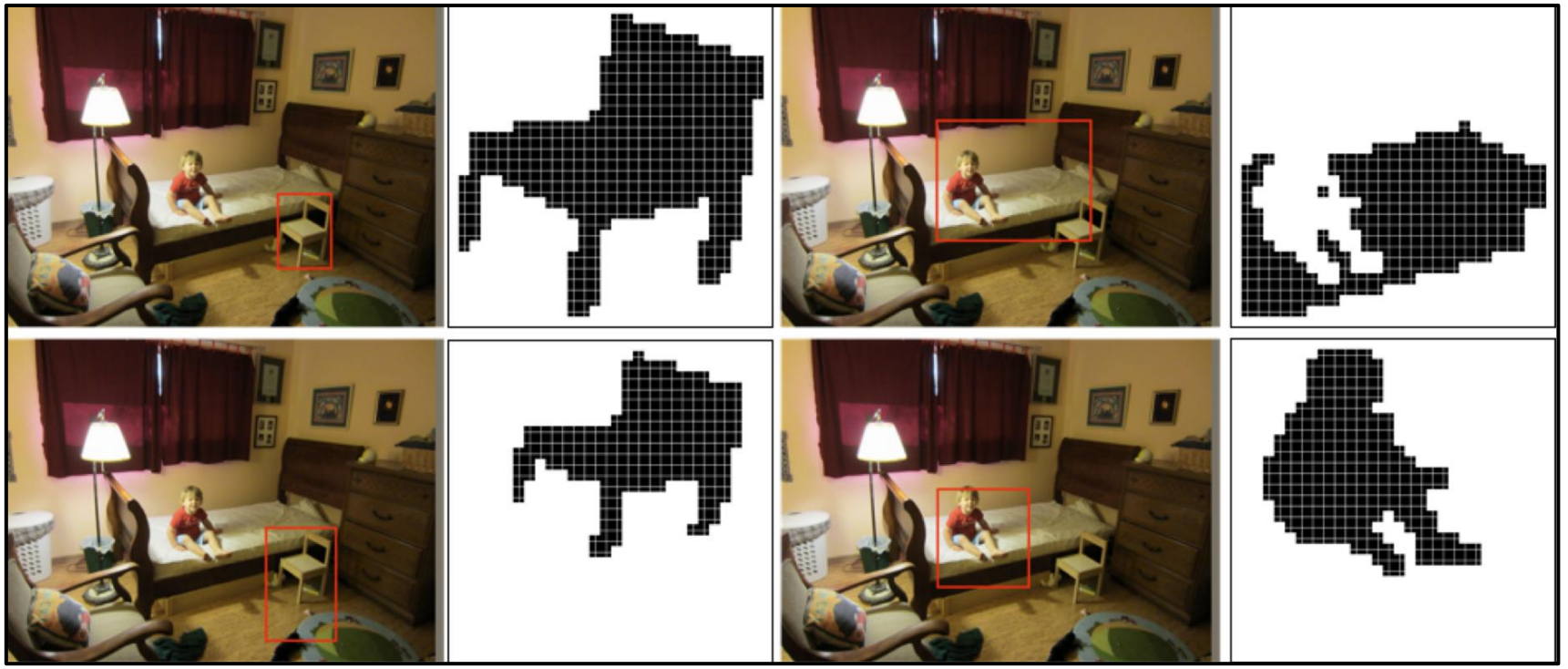
Depth-wise argmax

3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 1 1 1 1 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 5 5 5 5 5 5 5
5 5 3 3 3 3 1 1 3 3 5 5 5 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 4 4 5 5 5
4 4 4 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4

Semantic Labels

Instance Segmentation labels

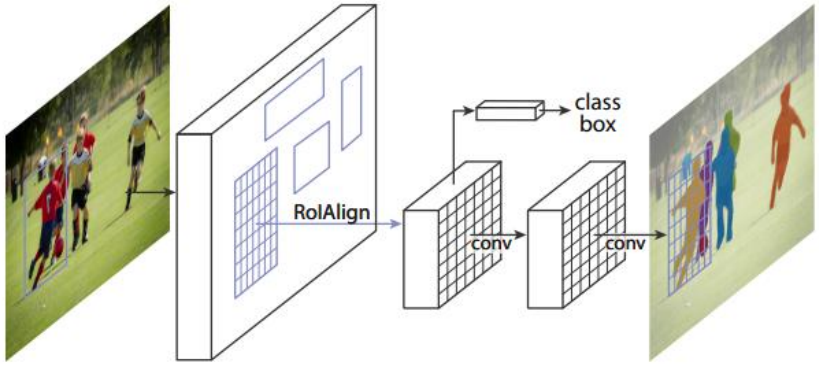
Mask R-CNN: Example Training Targets [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN architecture

- From [5] (More details on the paper)



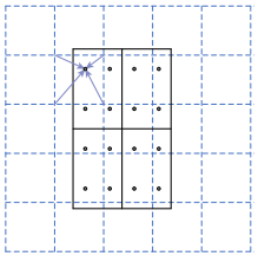


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

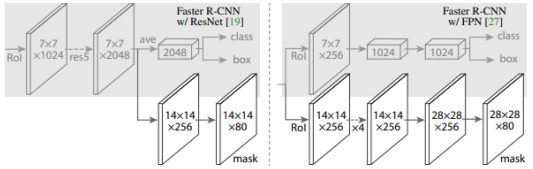
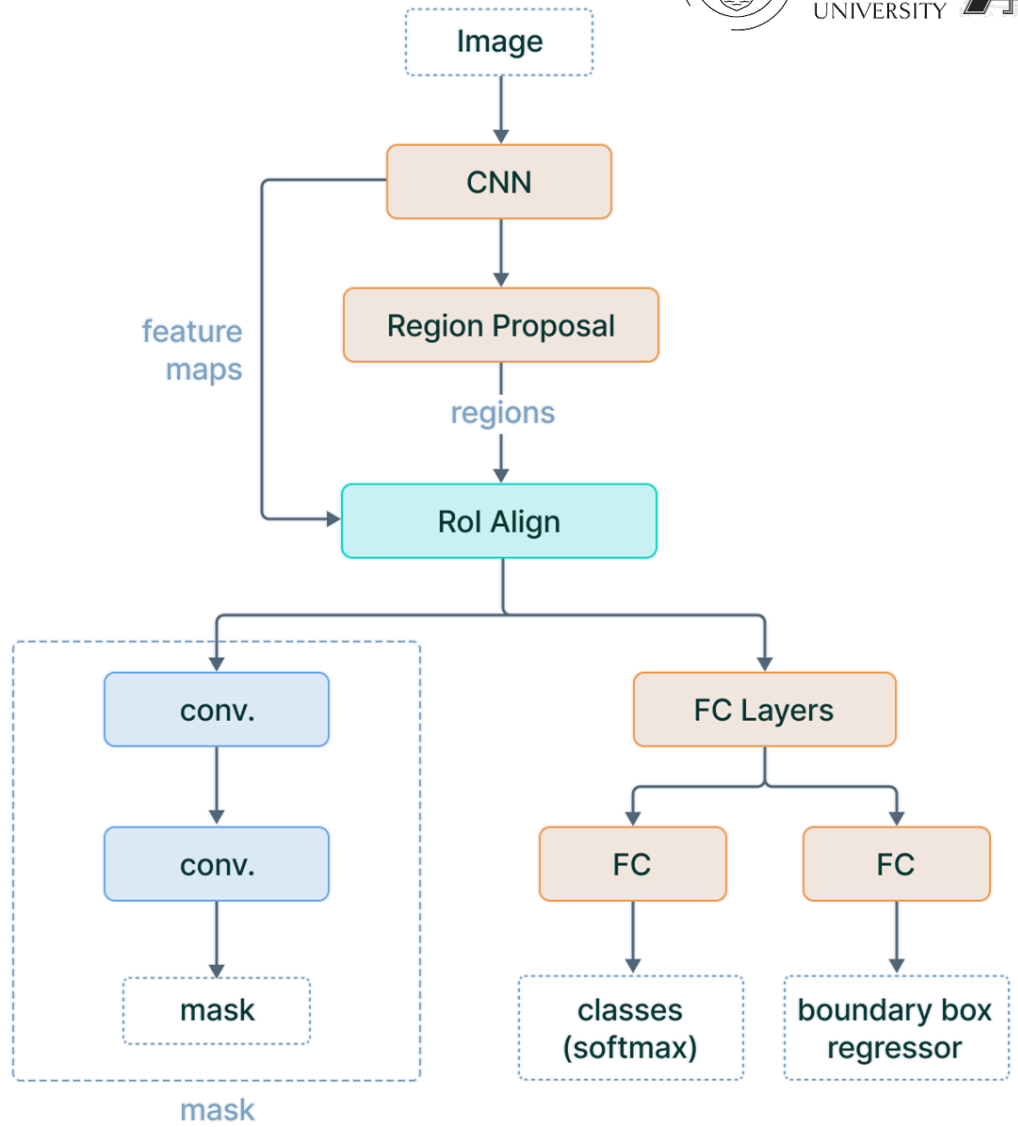
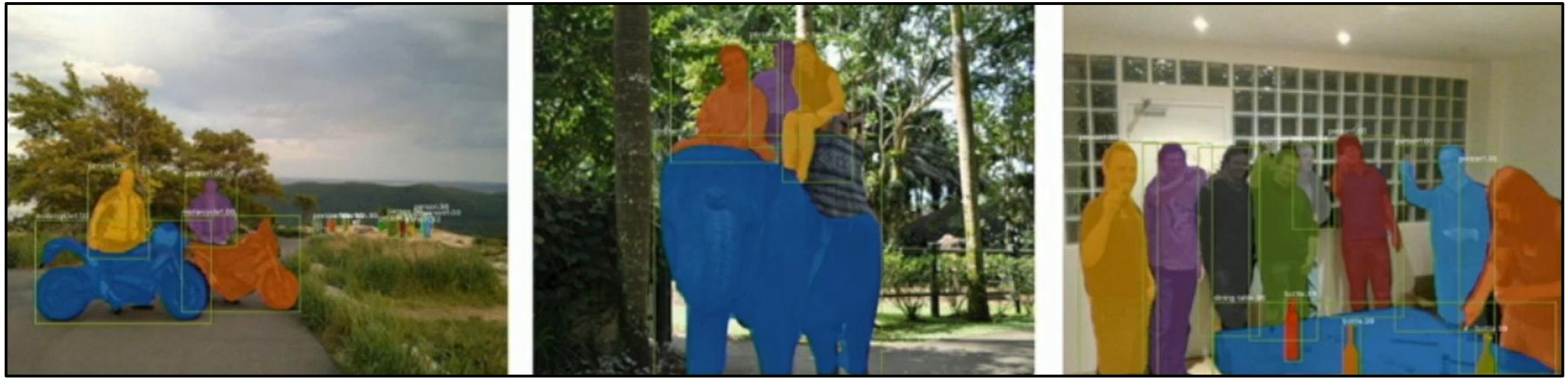


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN results [5]

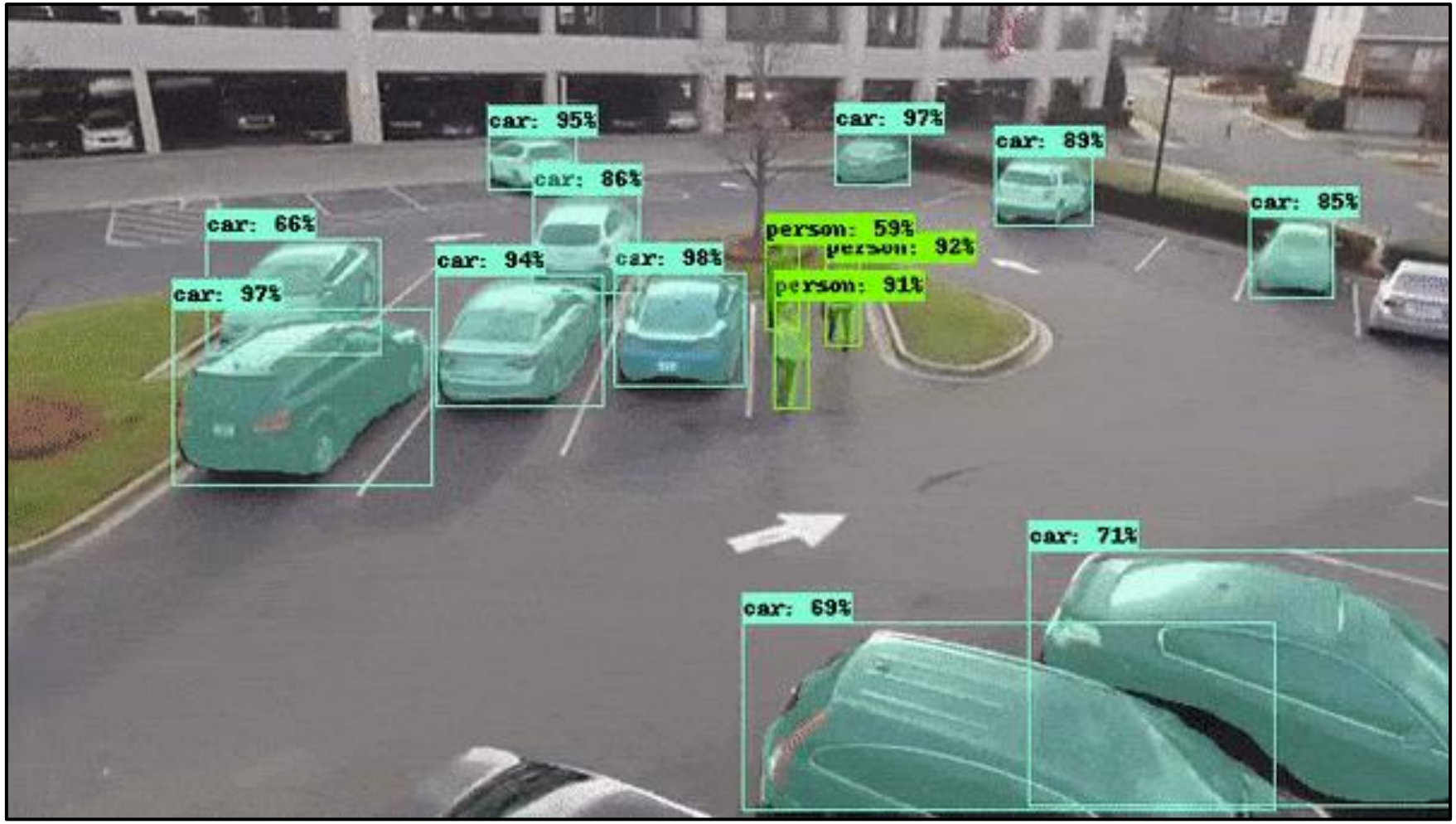


	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN results [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Summary

- **Object Detection** is a supervised learning task
- Goal is to predict what and where are the objects in an image

- (Core) Region proposal methods:
  - **R-CNN:** uses selective search to find regions
  - **Fast R-CNN:** first extracts features and then uses selective search
  - **Faster R-CNN:** uses a learnable region proposal network

- One-stage methods like **YOLO** and **SSD**, remove the region proposal part used in two-stage methods like the R-CNN family

- Many useful applications

- Techniques like **IoU** and **NMS** improve the predictions of object detectors
- **mAP** is a metric to evaluate object detectors

Summary

- **Upsampling** is essential to reconstruct the original image from lower-resolution feature maps.
- By increasing the resolution, **upsampling** enlarges images with the following methods:
  - **Unpooling** upsamples by distributing a single value over higher resolution.
  - **Transpose Convolution** reverses the operation of convolution.

- Object masks are predicted within an image through **Image Segmentation.**
- **Fully Convolutional Networks (FCNs)** serve as encoders for coarse feature maps but struggle with detailed segmentations.
- **U-Net** improves localization by expanding the decoder's capacity for segmentation tasks.
- With **Mask R-CNN**, adding a mask prediction head allows for extended segmentation capabilities.
- **Semantic Segmentation**: Treats all objects of the same class as one, using one-hot encoded class labels.
- **Instance Segmentation**: Identifies individual instances of the same object.

Further Links + Resources

- A survey of loss functions for semantic segmentation - https://arxiv.org/pdf/2006.14822
- R-CNN - https://medium.com/@selfouly/r-cnn-3a9beddfd55a
- Review: Fully Convolutional Network (Semantic Segmentation) - https://medium.com/towards-data-science/review-fcn-semantic-segmentation-eb8c9b50d2d1
- https://www.youtube.com/watch?v=TB-fdlSzpHQ
- https://www.youtube.com/watch?v=9AyMR4IhSWQ
- https://www.youtube.com/watch?v=ag3DLKsl2vk (for the explanation only)

Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

That’s all for today!