SEGMENTATION AND OBJECT DETECTION

INF5860 — Machine Learning for Image Analysis

Ole-Johan Skrede

21.03.2018

University of Oslo

Messages

- Mandatory 2, deadline on Friday
- Mandatory 3:
  - Not completely determined
  - Attempt to make a fun assignment
  - Will be in TensorFlow
  - Hand-out around Friday, April 13th

Outline

- Introduction and motivation
- Object classification and localization
- Object detection
- Semantic segmentation
- Instance segmentation

INTRODUCTION AND MOTIVATION

Image classification and object localization

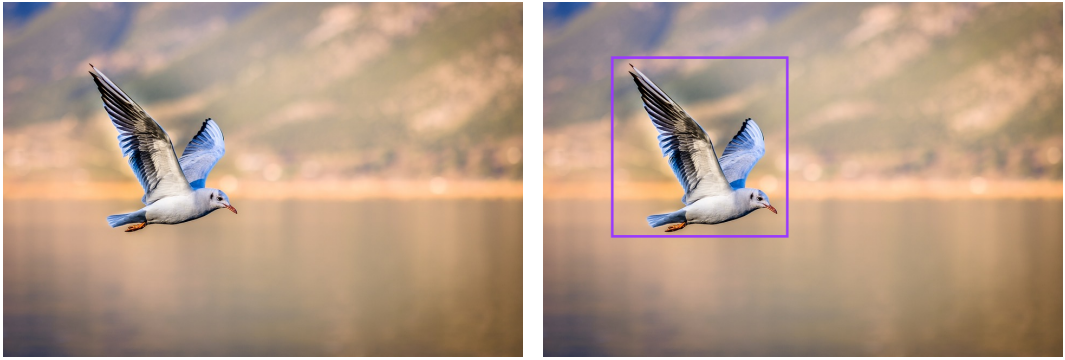
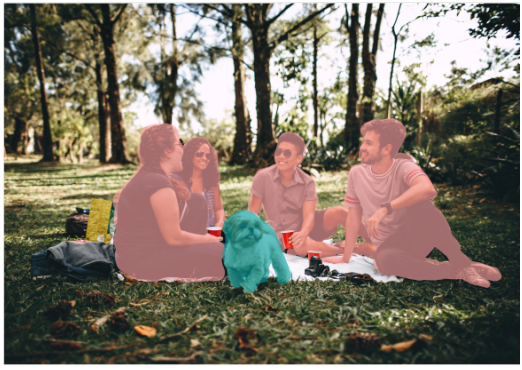


Figure 1: Seagull. Image source: https://www.pixabay.com

Object detection — Multiple instances

Figure 2: Rooster, cat, dog, donkey. Image source: https://www.pixabay.com

Semantic segmentation



Instance segmentation

Learning outcome

- We will see a lot of different approaches
- Not expected to know every algorithm inside-out
- Most important that you are educated about the possibilities
- Curriculum (exam-relevant), will be decided well in time before the exam

IMAGE CLASSIFICATION AND LOCALIZATION

Objective

- Classify an image with a single object
- Draw a bounding box around the object

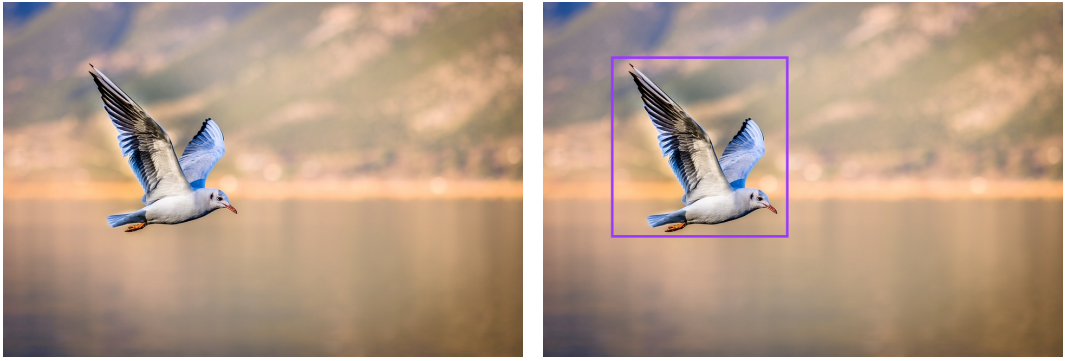


Figure 3: Seagull. Image source: https://www.pixabay.com

Label vector

- Add object/no object indicator c_0
- Interpret c_0 as $c_0 = \Pr(\text{there is an object in this box})$
- c_0 is often referred to as the *objectness*, but can also be thought of as a "catch-all" background class indicator
- Standard category probabilities from classification $(c_1, c_2, \ldots, c_{N_c})$
- Interpret c_i as $c_i = \Pr(\text{class}_i | c_0 = 1)$, $i = 1, \ldots, n$
- Add bounding box location specifiers
  - b_r: Center row coordinate
  - b_c: Center column coordinate
  - b_h: Box height
  - b_w: Box width

$$\hat{y} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N_c} \\ b_r \\ b_c \\ b_h \\ b_w \end{bmatrix}$$

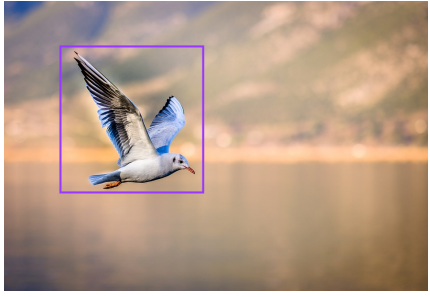


Figure 4: Seagull. Image source: https://www.pixabay.com

Example: Big cats

- c_1: Tiger
- c_2: Leopard
- c_3: Lion

$$\hat{y} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ b_r \\ b_c \\ b_h \\ b_w \end{bmatrix}$$

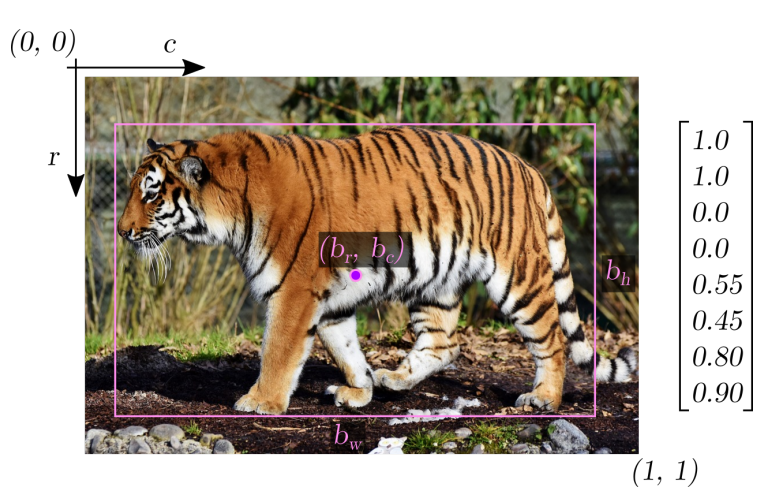


Figure 5: Tiger. Image source: https://www.pixabay.com

Example: Big cats

- c_1: Tiger
- c_2: Leopard
- c_3: Lion

$$\hat{y} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ b_r \\ b_c \\ b_h \\ b_w \end{bmatrix}$$

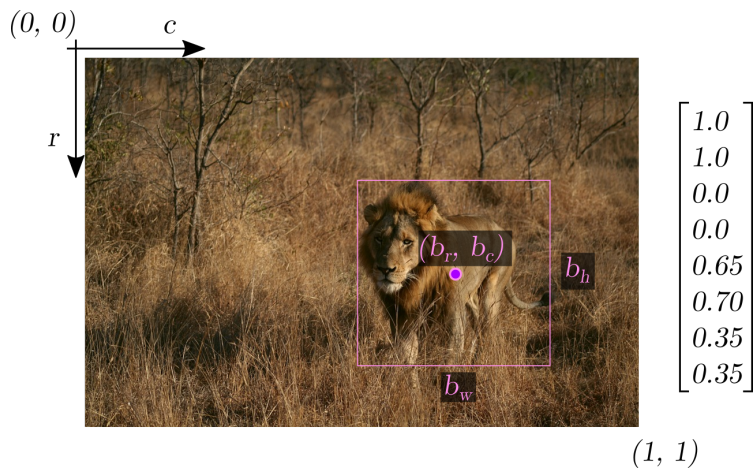


Figure 6: Lion. Image source: https://www.pixabay.com

Example: Big cats

- c_1: Tiger
- c_2: Leopard
- c_3: Lion

$$\hat{y} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ b_r \\ b_c \\ b_h \\ b_w \end{bmatrix}$$

Note that $c_0 = 0$, so we do not care about the rest, symbolized by $\emptyset$.

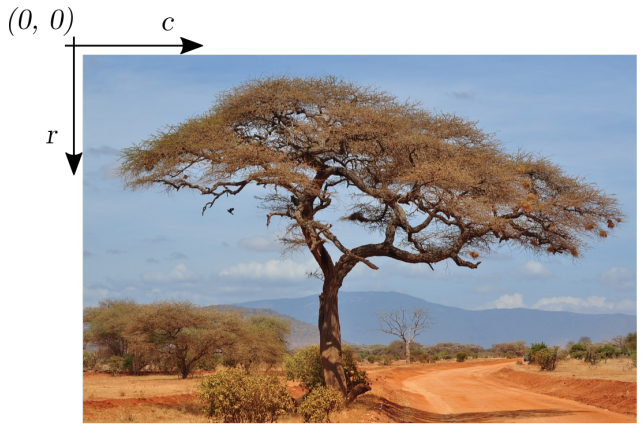


Figure 7: Savannah. Image source: https://www.pixabay.com

Loss function

- We could use a conditioned L_2 loss for all elements

$$L(\hat{y}, \tilde{y}) = \begin{cases} \sum_{i=0}^{|\hat{y}|-1} (\hat{y}_i - \tilde{y}_i)^2, & \text{if } \hat{y}_0 = 1 \\ (\hat{y}_0 - \tilde{y}_0)^2, & \text{if } \hat{y}_0 = 0 \end{cases}$$

- where
  - $\hat{y}$ is the reference (ground truth) label vector
  - $\tilde{y}$ is the proposed label vector

Multi-task loss function

- Partition y into $y = [c, b]$, with
  - $c = [c_0, c_1, \ldots, c_{N_c}]$
  - $b = [b_r, b_c, b_h, b_w]$
- L_2 loss for object bounding box location b

$$L_b(\hat{b}, \tilde{b}) = \sum_{i \in \{x,y,h,w\}} (\hat{b}_i - \tilde{b}_i)^2$$

- Cross entropy loss for object categories c

$$L_c(\hat{c}, \tilde{c}) = -\sum_{i=1}^{n} \tilde{c}_i \log \hat{c}_i$$

- The total loss can be written as

$$L(\hat{y}, \tilde{y}) = L_c + [\hat{c}_0 > 0] L_b$$

- Only compare bounding box if there is an object

Localization evaluation

- Sensitivity
- Specificity
- Precision
- Jaccard index

Binary case partition

- Let B_R be the set of pixels enclosed by the *reference* bounding box
- Let B_P be the set of pixels enclosed by the *proposed* bounding box
- True positive: $TP = |B_R \cap B_P|$
- False negative: $FN = |B_R \setminus B_P|$ (*Type I error*)
- False positive: $FP = |B_P \setminus B_R|$ (*Type II error*)
- True negative: $TN = |(B_R \cup B_P)^c|$
- Where, for sets $A, B \subseteq C$
  - $A \cap B = \{x : x \in A, x \in B\}$ is the *intersection* of A and B
  - $A \setminus B = \{x : x \in A, x \notin B\}$ is the *set difference* between A and B
  - $(A \cup B)^c = \{x : x \notin A, x \notin B, x \in C\}$ is the *complement* of $(A \cup B)$

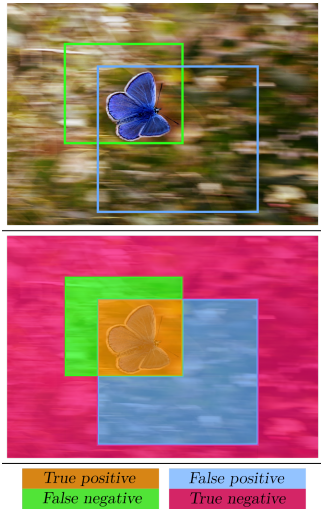


Figure 8: Top: reference (green), proposal (blue). Bottom: TP (orange), FN (green), FP (blue), TN (red). Image source: https://www.pixabay.com

Sensitivity

- Proportion of positive reference instances labeled as positive by the proposed method

$$tpr = \frac{|B_R \cap B_P|}{|B_R|}$$

- Also known as
  - *true positive rate* (tpr)
  - *recall*
- Example on the right is pixel classification, but it also applies to object instances

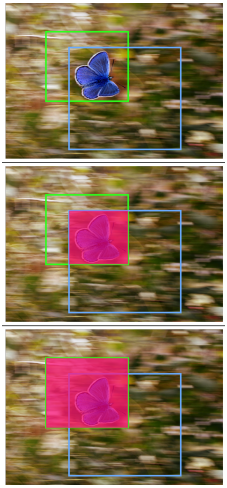


Figure 9: Top: reference (green), proposal (blue). Middle: True positive (red). Bottom: Reference positive (red). Image source: https://www.pixabay.com

Specificity

- Proportion of negative reference instances labeled as negative by the proposed method

$$tnr = \frac{|(B_R \cap B_P)^c|}{|B_R^c|}$$

- Also known as *true negative rate* (tnr)

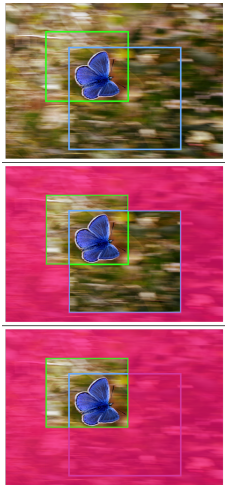


Figure 10: Top: reference (green), proposal (blue). Middle: True negative (red). Bottom: Reference negative (red). Image source: https://www.pixabay.com

Precision

- Proportion of proposed positive instances that are also labeled positive by the reference

$$ppv = \frac{|B_R \cap B_P|}{|B_P|}$$

- Also known as *positive predictive value* (ppv)
- Example on the right is pixel classification, but it also applies to object instances

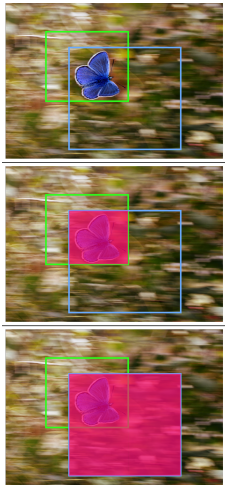


Figure 11: Top: reference (green), proposal (blue). Middle: True positive (red). Bottom: Proposed positive (red). Image source: https://www.pixabay.com

Jaccard index

- The proportion of all instances classified as positive by the reference and/or the proposal method, that are classified as positive by both the reference and the proposal method

$$iou = \frac{|B_R \cap B_P|}{|B_R \cup B_P|}$$

- Also known as
  - Intersection over Union (IoU)
  - Tanimoto index
- Example on the right is pixel classification, but it also applies to object instances

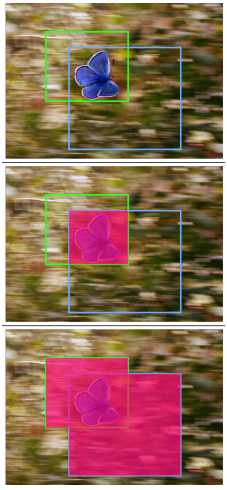


Figure 12: Top: reference (green), proposal (blue). Middle: Intersection (red). Bottom: Union (red). Image source: https://www.pixabay.com

OBJECT DETECTION

Classes of object detection

OD: Objectness detection

- Detect all objects in an image
- Does not care about category

SOD: Salient object detection

- Inspired by human visual attention system
- Focus on a few informative image regions

COD: Category-specific object detection

- Detect objects from predefined categories
- Detect and classify object
- This is the focus in this lecture

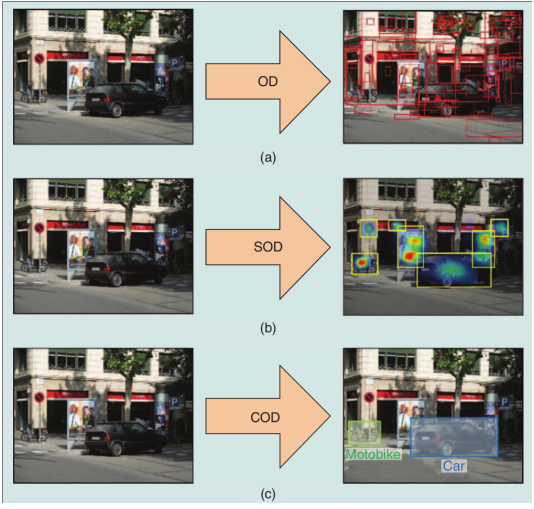
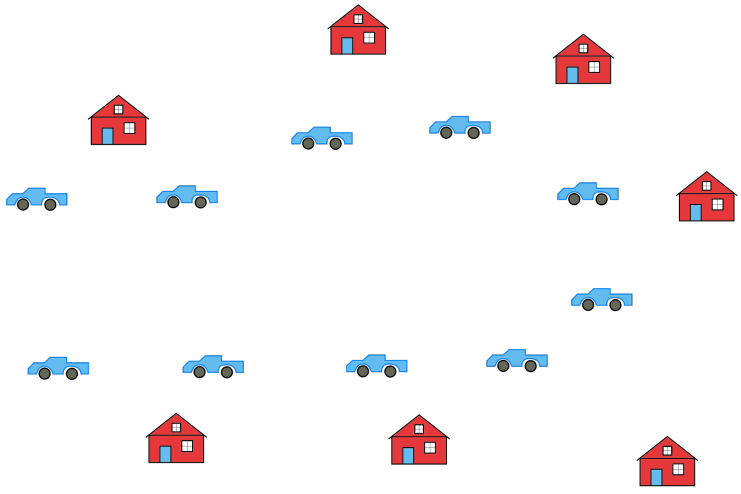


Figure 13: Source: [Han et al., 2018]

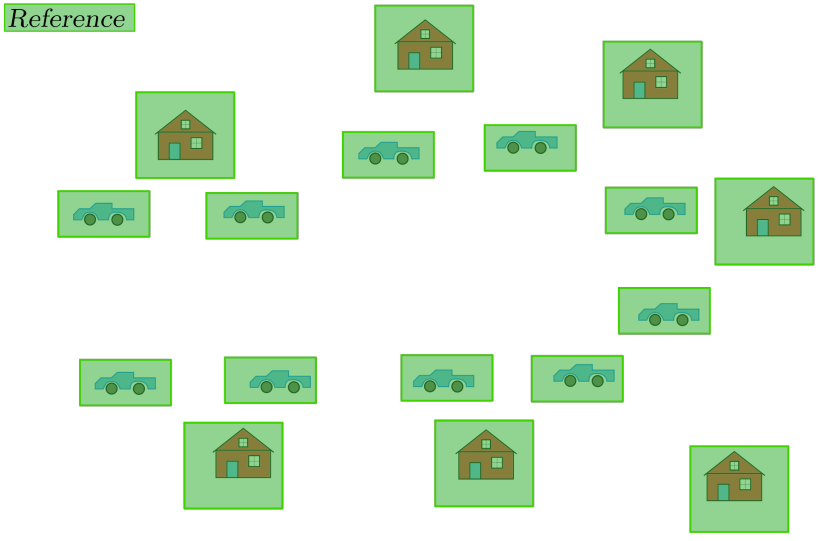
Object detection evaluation — mean average precision

- Mean average precision (mAP) is a common evaluation metric in object detection
- Each detected region (a bounding box) has a confidence score for each class
- Set all detected regions with an $IoU < T$ to be a false positive region, no matter the confidence score
- Common threshold values are $T = 0.5$
- For each class, compute the *Average Precision* (AP)
  - Sort detections by decreasing confidence
  - Compute precision and recall cumulatively as you "walk" from high confidence to low confidence
  - Average all computed precision values by some method (will be discussed more in detail later)
- Average the computed average precision scores over all classes

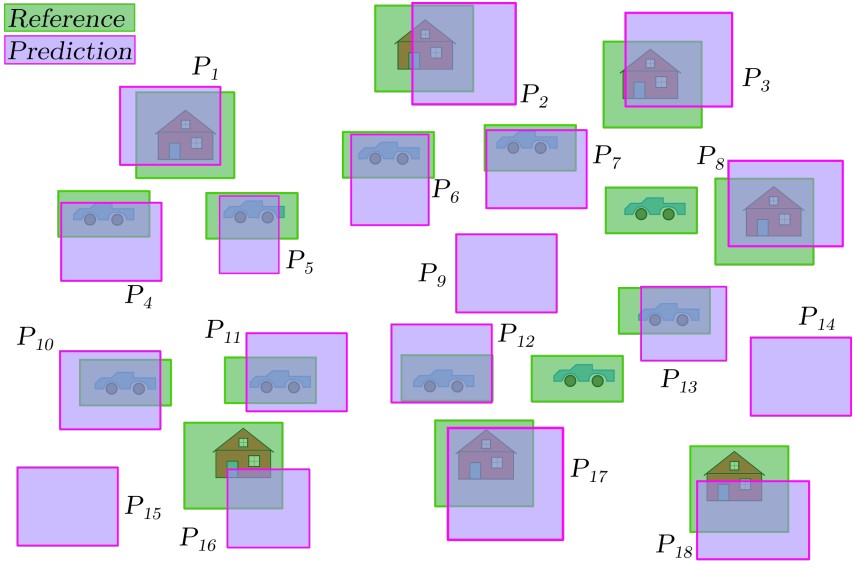
Mean average precision example — initial image



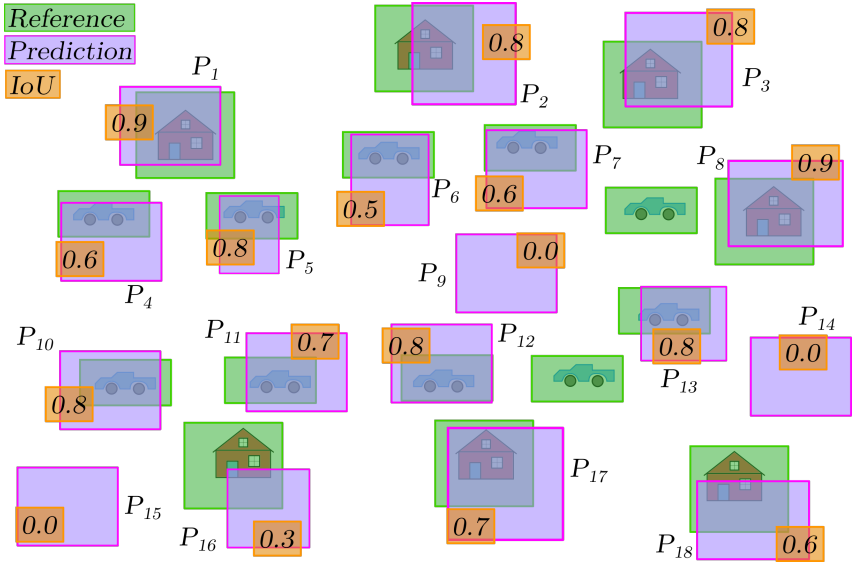
Mean average precision example — reference



Mean average precision example — reference and prediction



Mean average precision example — Intersection over Union



Mean average precision example — class confidences

	P confidence		R confidence		$IoU \geq 0.5$
	Car	House	Car	House	
P_1	0.1	0.9	0	1	✓
P_2	0.3	0.7	0	1	✓
P_3	0.3	0.7	0	1	✓
P_4	0.6	0.4	1	0	✓
P_5	0.7	0.3	1	0	✓
P_6	0.5	0.5	1	0	✓
P_7	0.8	0.2	1	0	✓
—	—	—	1	0	✗
P_8	0.1	0.9	0	1	✓
P_9	0.4	0.6	—	—	✗
P_{10}	0.8	0.2	1	0	✓
P_{11}	0.6	0.4	1	0	✓
P_{12}	0.9	0.1	1	0	✓
—	—	—	1	0	✗
P_{13}	0.7	0.3	1	0	✓
P_{14}	0.6	0.4	—	—	✗
P_{15}	0.5	0.5	—	—	✗
P_{16}	0.7	0.3	0	1	✗
P_{17}	0.2	0.8	0	1	✓
P_{18}	0.4	0.6	0	1	✓

Mean average precision — Precision / recall

- We compute the precision and recall (sensitivity) for each class independently
- First, we sort the predictions by decreasing confidence
- Then, we compute the precision and recall for the different confidence levels
- The recall at a confidence level is the number of true positive predictions at this confidence level and above, divided by all reference positive instances
- The precision at a confidence level is the number of true positive predictions at this confidence level and above, divided by all proposed positive instances at this confidence level and above

Mean average precision example — Precision / recall for car

	P car confidence	R car	$IoU \geq 0.5$	Recall	Precision
P_{12}	0.9	1	✓	$\frac{1}{10}$	$\frac{1}{1}$
P_{10}	0.8	1	✓	$\frac{2}{10}$	$\frac{2}{2}$
P_7	0.8	1	✓	$\frac{3}{10}$	$\frac{3}{3}$
P_{16}	0.7	0	✗	$\frac{3}{10}$	$\frac{3}{4}$
P_{13}	0.7	1	✓	$\frac{4}{10}$	$\frac{4}{5}$
P_5	0.7	1	✓	$\frac{5}{10}$	$\frac{5}{6}$
P_{14}	0.6	—	✗	$\frac{5}{10}$	$\frac{5}{7}$
P_{11}	0.6	1	✓	$\frac{6}{10}$	$\frac{6}{8}$
P_4	0.6	1	✓	$\frac{7}{10}$	$\frac{7}{9}$
P_{15}	0.5	—	✗	$\frac{7}{10}$	$\frac{7}{10}$
P_6	0.5	1	✓	$\frac{8}{10}$	$\frac{8}{11}$
P_{18}	0.4	0	✓	$\frac{8}{10}$	$\frac{8}{12}$
P_9	0.4	—	✗	$\frac{8}{10}$	$\frac{8}{13}$
P_3	0.3	0	✓	$\frac{8}{10}$	$\frac{8}{14}$
P_2	0.3	0	✓	$\frac{8}{10}$	$\frac{8}{15}$
P_{17}	0.2	0	✓	$\frac{8}{10}$	$\frac{8}{16}$
P_8	0.1	0	✓	$\frac{8}{10}$	$\frac{8}{17}$
P_1	0.1	0	✓	$\frac{8}{10}$	$\frac{8}{18}$

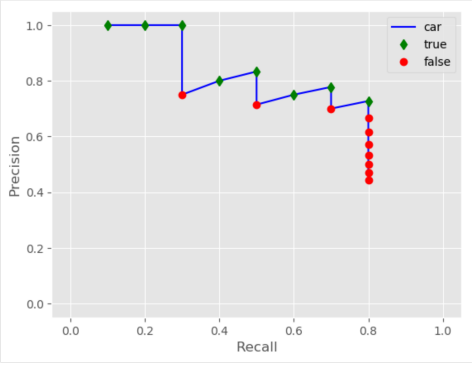


Figure 14: Left: Car predictions sorted by decreaseing confidence. Right: Corresponding precision-recall plot.

Mean average precision example — Precision / recall for house

	P house confidence	R house	$IoU \geq 0.5$	Recall	Precision
P_1	0.9	1	✓	$\frac{1}{7}$	$\frac{1}{1}$
P_8	0.9	1	✓	$\frac{2}{7}$	$\frac{2}{2}$
P_{17}	0.8	1	✓	$\frac{3}{7}$	$\frac{3}{3}$
P_2	0.7	1	✓	$\frac{4}{7}$	$\frac{4}{4}$
P_3	0.7	1	✓	$\frac{5}{7}$	$\frac{5}{5}$
P_9	0.6	—	✗	$\frac{5}{7}$	$\frac{5}{6}$
P_{18}	0.6	1	✓	$\frac{6}{7}$	$\frac{6}{7}$
P_6	0.5	0	✓	$\frac{6}{7}$	$\frac{6}{8}$
P_{15}	0.5	—	✗	$\frac{6}{7}$	$\frac{6}{9}$
P_4	0.4	0	✓	$\frac{6}{7}$	$\frac{6}{10}$
P_{11}	0.4	0	✓	$\frac{6}{7}$	$\frac{6}{11}$
P_{14}	0.4	—	✗	$\frac{6}{7}$	$\frac{6}{12}$
P_5	0.3	0	✓	$\frac{6}{7}$	$\frac{6}{13}$
P_{16}	0.3	1	✗	$\frac{6}{7}$	$\frac{6}{14}$
P_{13}	0.3	0	✓	$\frac{6}{7}$	$\frac{6}{15}$
P_7	0.2	0	✓	$\frac{6}{7}$	$\frac{6}{16}$
P_{10}	0.2	0	✓	$\frac{6}{7}$	$\frac{6}{17}$
P_{12}	0.1	0	✓	$\frac{6}{7}$	$\frac{6}{18}$

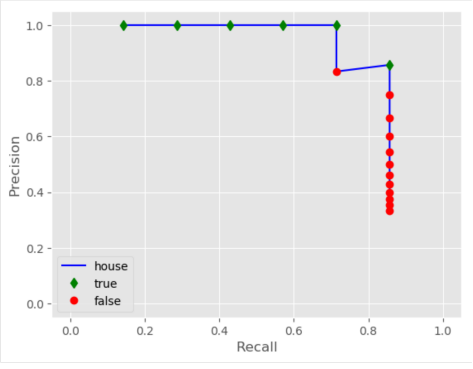
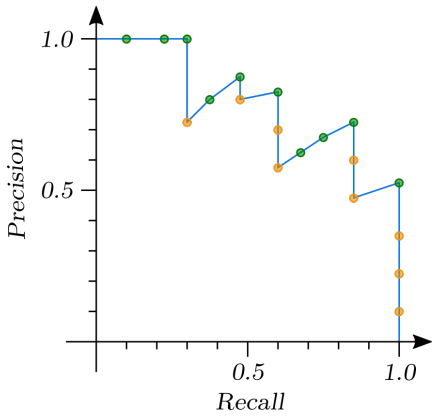


Figure 15: Left: House predictions sorted by decreaseing confidence. Right: Corresponding precision-recall plot.

Mean average precision — Averaging

- We get the mean average precision by averaging the *Average Precision* (AP) over all classes
- The average precision is usually related to the area under the curve (AUC) made by graphing the precision vs recall.
- Some common definitions of AP:
  - The area under the precision / recall curve
  - The area under the *interpolated* precision / recall curve
- See e.g. [Everingham et al., 2015] for more info

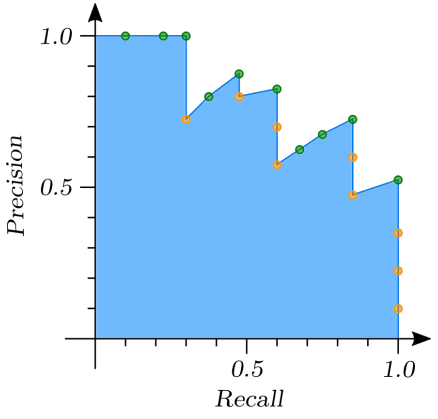


Mean average precision — AP from AUC

- Compute the true area under the precision / recall curve
- Area of the blue region in the figure to the right
- E.g. used in the PASCAL VOC object detection competition from 2010 and onwards
- Common to use the trapezoidal rule to compute the area
- Can also use a simpler estimate (e.g. in `sklearn.metrics.average_precision_score`)

$$AP = \sum_{n=2}^{N} (r_n - r_{n-1}) p_n \tag{1}$$

for (precision, recall) entries $\{(p_1, r_1), (p_2, r_2), \ldots, (p_N, r_N)\}$



Mean average precision — AP from interpolated AUC

- Interpolate the precision / recall curve

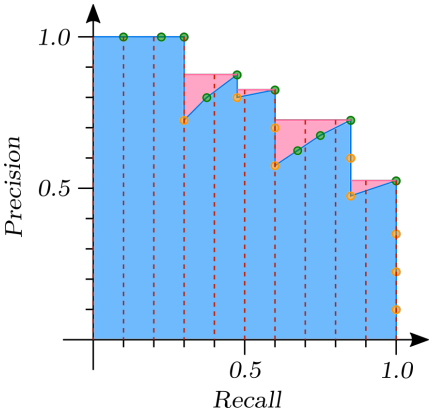
$$p_{\text{interp}}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r})$$

  where $p(\tilde{r})$ is the precision at recall $\tilde{r}$.

- Take the average of the interpolated precision at evenly sampled recall values

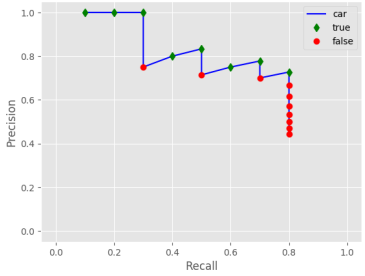
$$AP = \frac{1}{11} \sum_{r\in\{0.0,0.1,\ldots,1.0\}} p_{\text{interp}}(r)$$

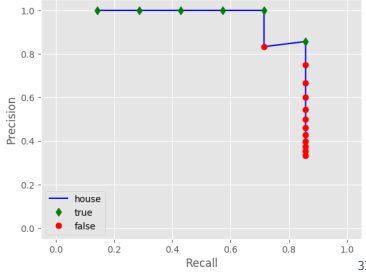
- Used in the PASCAL VOC object detection challenge up until 2009
- Intended to reduce the impact of “wiggles” in the precision / recall curve
- Downside: Can be too crude



Mean average precision example — Averaging over classes

- Using eq. (1) to compute AP, we get
  - Car class: $AP_{\text{car}} \approx 0.5888$
  - House class: $AP_{\text{car}} \approx 0.6939$
- The mean of the two is then $mAP \approx 0.6414$





Bounding box regression for object detection

- Impractical to encode a bounding box $b = [b_x, b_y, b_h, b_w]$ for every class
- Every image would need a custom number of outputs

Figure 17: Image source: https://www.pixabay.com

Sliding window object detection

- Train a classification net on images tightly cropped around objects
- Slide a window (of multiple sizes) over the image you want to detect objects in
- Give each pixel a score based on the trained classification net on each window
- Ok for cheap classification methods
- Very slow for CNN classification

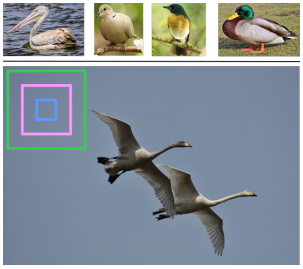


Figure 18: Image source: https://www.pixabay.com

- Convolutional implementation can fix the efficiency problem (see e.g. OverFeat method)
- Still not very precise bounding boxes

Region proposal detection

- A subclass of object detection methods
- Use a separate method to find candidate regions
- Filter out regions without an object, or redundant, overlapping regions with an object
- Classify these regions and refine region boundary

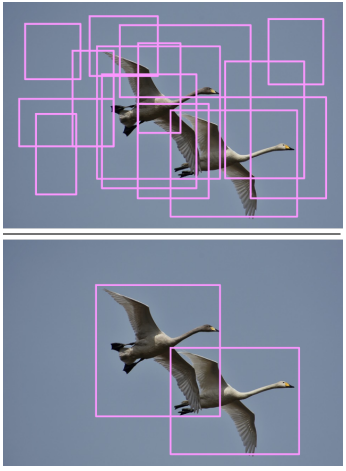


Figure 19: Image source: https://www.pixabay.com

R-CNN — overview

- Originally proposed in [Girshick et al., 2014]
- Combines region proposals and convolutional neural networks
- Consists of three modules:
  - For each image, propose a set of category-independent regions
  - Extract a fixed-length feature vector from each region using a CNN
  - Classify the feature vectors with a class-specific linear SVM

Figure 20: Image source: https://www.pixabay.com

R-CNN

- Propose regions (~ 2000)
  - In the original publication they use *selective search* [Uijlings et al., 2013]

Figure 21: Image source: https://www.pixabay.com

R-CNN

- Propose regions (~ 2000)
  - In the original publication they use *selective search* [Uijlings et al., 2013]
- Warp the regions (regardless of shape) to a fixed size

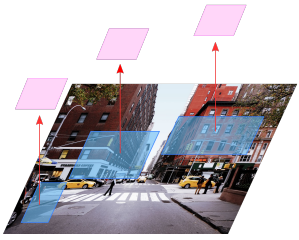


Figure 22: Image source: https://www.pixabay.com

[1]Pretrained on ImageNet, finetuned to Pascal VOC, see paper for details

R-CNN

- Propose regions (~ 2000)
  - In the original publication they use *selective search* [Uijlings et al., 2013]
- Warp the regions (regardless of shape) to a fixed size
- Feature extraction
  - Feed the warped regions into a pretrained CNN
  - Output a 4096-dimensional feature vector

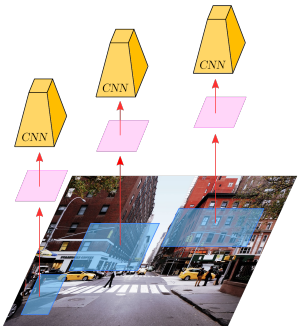


Figure 23: Image source: https://www.pixabay.com

[1] Pretrained on ImageNet, finetuned to Pascal VOC, see paper for details

R-CNN

- Propose regions (~ 2000)
  - In the original publication they use *selective search* [Uijlings et al., 2013]
- Warp the regions (regardless of shape) to a fixed size
- Feature extraction
  - Feed the warped regions into a pretrained CNN
  - Output a 4096-dimensional feature vector
- A pretrained SVM scores each region
- Reject regions with non-maximum suppression
- Tune bounding boxes with a pretrained regression model

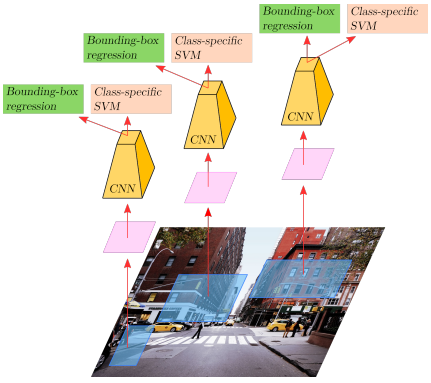


Figure 24: Image source: https://www.pixabay.com

Non-max suppression

- Important step in several object detection algorithms
- Remove all boxes with associate objectness c_0 smaller than some threshold, say $c_0 < 0.6$
- For each class $i = 1, 2, \ldots, n$
  - Create a list of unseen regions U_i that contains all the regions in the image
  - Create an empty list of regions to keep K_i
  - While there are regions left in U_i
    - Find the most probable region $R_{\max}$
    - $R_{\max}$ can be the region with highest value of $c_0 c_i$ (or some similar criterion)
    - Remove all regions that overlaps with $R_{\max}$ (e.g. with $iou > 0.5$), from U_i
    - Move $R_{\max}$ from U_i to K_i

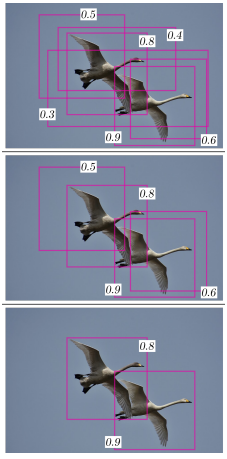


Figure 25: Top: Original. Middle: Too low c_0 removed. Bottom: $iou > 0.5$ removed. Image source: https://www.pixabay.com

Fast R-CNN

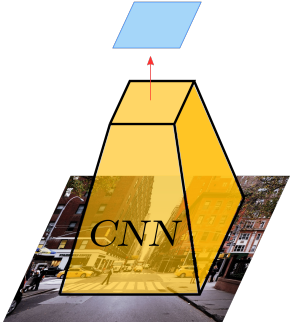
- Publication: [Girshick, 2015]
- The original R-CNN is slow and expencive:
  - Multi-stage pipeline
  - Classify each proposed region with a CNN
  - Intermediate feature maps are stored, takes a lot of space
- Fast R-CNN manages to save computation by first feeding the entire image into a CNN

Fast R-CNN

- Get region proposals (as in R-CNN)

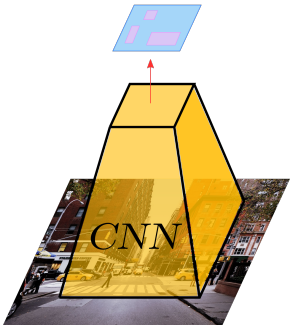
Fast R-CNN

- Get region proposals (as in R-CNN)
- Run a CNN on the entire image



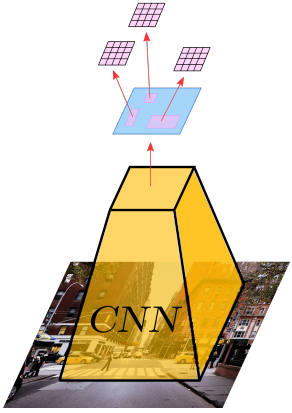
Fast R-CNN

- Get region proposals (as in R-CNN)
- Run a CNN on the entire image
- Project region proposals (ROIs) onto output CNN feature map



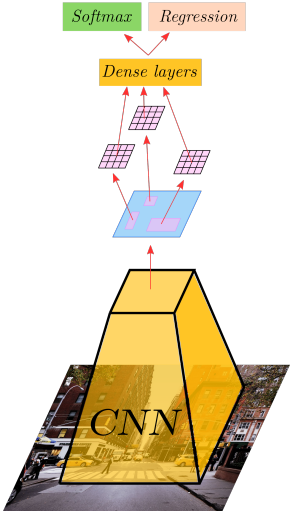
Fast R-CNN

- Get region proposals (as in R-CNN)
- Run a CNN on the entire image
- Project region proposals (ROIs) onto output CNN feature map
- ROI pooling layer



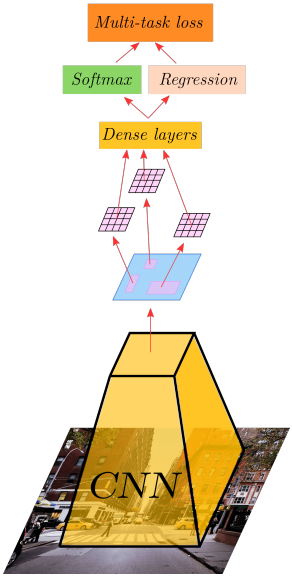
Fast R-CNN

- Get region proposals (as in R-CNN)
- Run a CNN on the entire image
- Project region proposals (ROIs) onto output CNN feature map
- ROI pooling layer
- Feed the fixed-sized pooled region to fully connected layers
- One softmax output for class prediction
- One regression output for the bounding box prediction



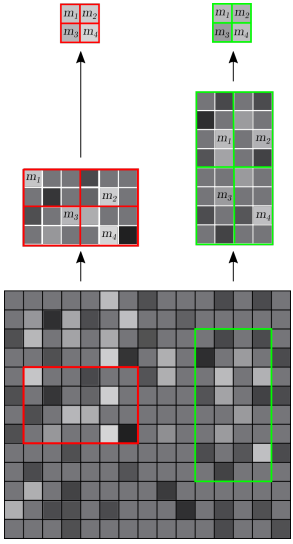
Fast R-CNN

- Get region proposals (as in R-CNN)
- Run a CNN on the entire image
- Project region proposals (ROIs) onto output CNN feature map
- ROI pooling layer
- Feed the fixed-sized pooled region to fully connected layers
- One softmax output for class prediction
- One regression output for the bounding box prediction
- Multi-task loss



Region of Interest (ROI) pooling

- Max-pooling on input of non-uniform size
- Often used when extracting features for regions
- One list of region descriptors: (b_r, b_c, b_h, b_w) for every region
- One, shared feature map
- ROI-pooling partitions the box region into a $S \times S$ grid
- Extracts the max in each grid cell
- This results in a $S \times S$ matrix, no matter the input shape

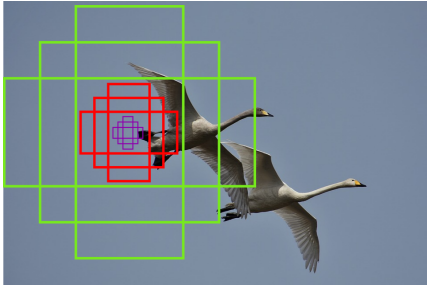


Faster R-CNN

- Publication: [Ren et al., 2016]
- With Fast R-CNN, the region proposal is the bottleneck
- Faster R-CNN introduces a Region Proposal Network (RPN)
- Consists of two modules:
  - A fully convolutional RPN
  - The Fast R-CNN detection network
- The RPN “tells Fast R-CNN where to look”
- The RPN and the detection network share layers

Anchor boxes

- A location have multiple boxes in different sizes and shapes
- Spread these anchor locations around an image
- Anchor shapes can be task-specific
- Training/labeling: Every anchor box is labeled as containing an object or not
- The overlap between the anchor box and reference box determines the label

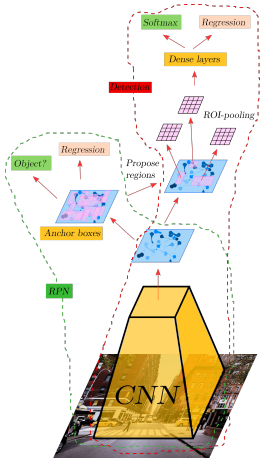


Region proposal network (RPN)

- A small network that scans the anchor boxes
- Predicts if an anchor box contain an object
- Refines the anchor box region boundary
- The regions with highest confidence of containing an object is kept
- Non-maximum suppression handles overlap
- These proposal regions go on to the next stage

Faster R-CNN — Complete picture

- The CNN extracts features
- Features are used by the RPN
- The RPN proposes regions to a detection network (like Fast-RCN)
- The detection network uses the same features as the RPN
- Training:
  - Alternate between RPN and detection network training
  - Or, train them jointly



YOLO

- Publication: [Redmon et al., 2016]
- Very fast (about $100\times$ faster than Fast R-CNN)
- Not as accurate as the most accurate methods

YOLO — GRID

- Partition image into $S \times S$ grid
- A grid cell is responsible for detecting an object with center point in that grid cell
- Each grid predicts B bounding boxes, and a confidence score for each box
- Confidence:
  $c_0 \cdot iou(\text{reference box}, \text{predicted box})$
  (zero if there is no reference box in that cell)

Figure 26: Image source: https://www.pixabay.com

YOLO — PREDICTION VECTOR

- The target vector has then the shape $(S, S, 5 \cdot B + N_c)$
- For each cell:
  - One set of parameters $(c_0, b_r, b_c, b_h, b_w)$, for every bounding box
  - A set of class parameters $(c_1, c_2, \ldots, n_c)$
- Can train against the target vector with a conventional CNN
- Limitation: One cell can predict B objects of the *same* class
- Extension: Use multiple anchor boxes, each with an associated class

Figure 27: Image source: https://www.pixabay.com

Object detection — summary

- A challenging problem
- Impressive progress the last few years
- More recent methods
  - R-FCN
  - SSD

SEMANTIC SEGMENTATION

Semantic segmentation

- Classify every pixel in an image
- Differentiate between classes
- Do not differentiate between multiple instances of the same class

Figure 28: Top: Original. Bottom: Segmented. Image source: https://www.pexels.com

Sliding window classification

- Select a small window
- Classify this image
- Assign the center pixel of this window the most probable class
- Repeat for all pixels in the image
- Very inefficient
- Misses larger image context

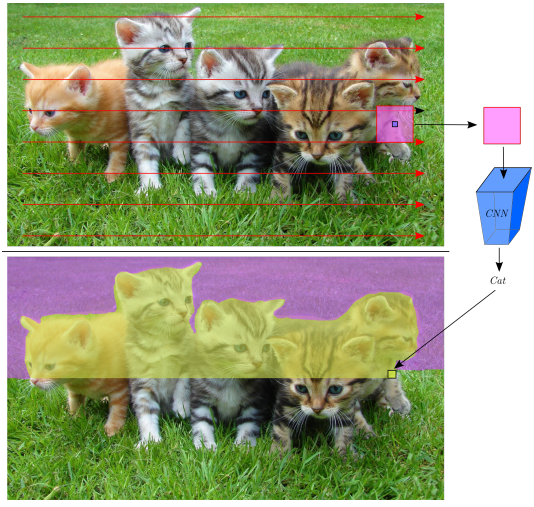


Figure 29: Top: Original. Bottom: Partially segmented. Image source: https://www.pexels.com

CNN multiple pixel classification

- Segment image all at once
- Input image shape: $H \times W \times C$
- Output layer shape: $H \times W \times N_c$, where N_c: number of classes
- Pixel-wise cross entropy loss
  - Softmax over channels at a pixel location
  - Repeat, and average over all pixels
- Very expensive on computation and memory

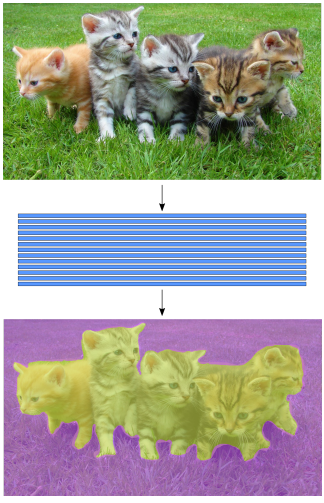


Figure 30: Top: Original. Bottom: Segmented. Image source: https://www.pexels.com

Downsampling and upsampling

- Segment image all at once
- Input image shape: $H \times W \times C$
- Output layer shape: $H \times W \times N_c$, where N_c: number of classes
- Spatial downsampling followed by upsampling (encoding, decoding)
- Pixel-wise cross entropy loss
  - Softmax over channels at a pixel location
  - Repeat, and average over all pixels
- Different upsampling techniques

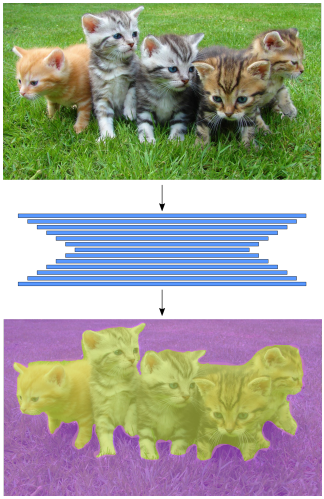
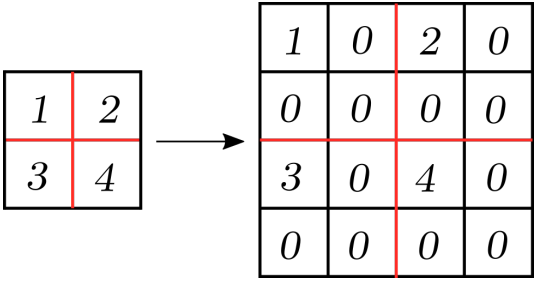


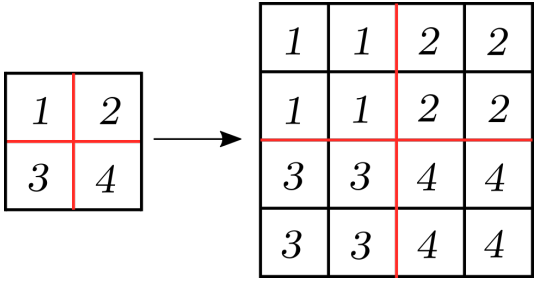
Figure 31: Top: Original. Bottom: Segmented. Image source: https://www.pexels.com

Simple unpooling

Fill with zeros

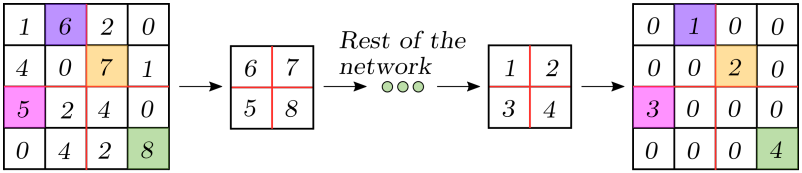


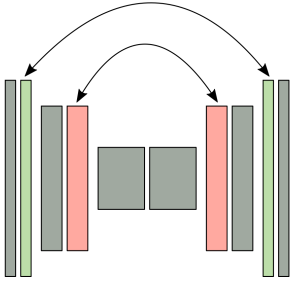
Fill with same



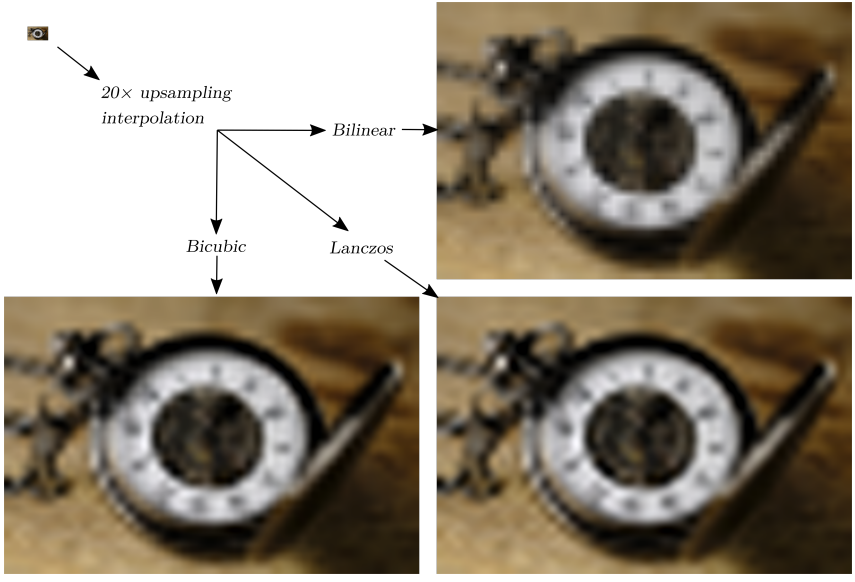
Max unpooling

Remember max locations from max pool downsampling. Reverse this on the “opposite” layer





Interpolation upsampling

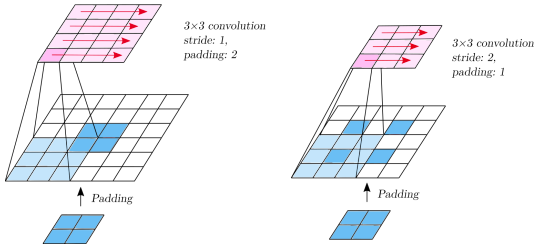


Trainable upsampling

- Upsampling convolution
- In depth convolution tutorial here
- Can learn kernel parameters as with regular convolution

Transposed convolution

- Can view convolution as a matrix-matrix multiplication
- Transposed convolution gets its name by transposing this operation
- Also called
  - fractionally strided convolution
  - econvolution (this is a misnomer)



FCN

- Publication: [Long et al., 2014]
- Early adaptor of segmentation with end-to-end trained CNNs
- Uses learnable transposed convolution in upsampling

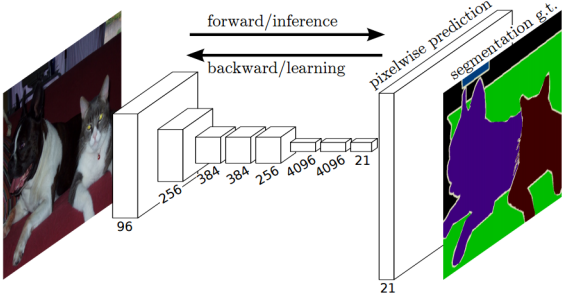


Figure 32: FCN architecture. Image source: [Long et al., 2014]

FCN — SKIP CONNECTIONS

- Aggressive upsampling leads to course segmentation result
- Combine upsampling from different parts in the layer
- Each with different upscaling

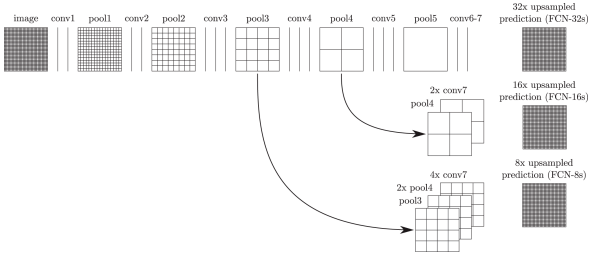
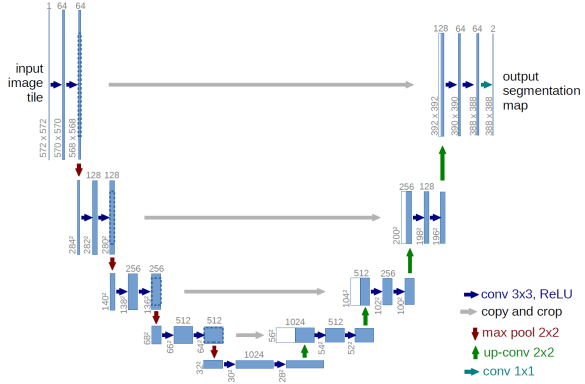


Figure 33: FCN architecture with different upsampling strategies. Image source: [Long et al., 2014]

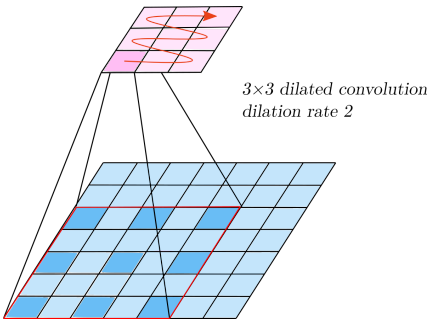
U-net

- Publication: [Ronneberger et al., 2015]
- Contraction: Ordinary convolution and pooling layers
- Expansion: Concatenation of
  - Cropped feature maps from contraction phase (gray arrows)
  - Transposed convolution from previous layer



Dilated convolution

- Insert spacing between convolution kernel cells (dilation rate)
- Also called
  - convolution with holes
  - A-trous convolution (a trous is french for with holes)



- Publication: [Chen et al., 2016]
- VGG16 or ResNet as base networks
- Employs dilated convolution in upsampling
- v3 currently holds top position on the PASCAL VOC segmentation leaderboard

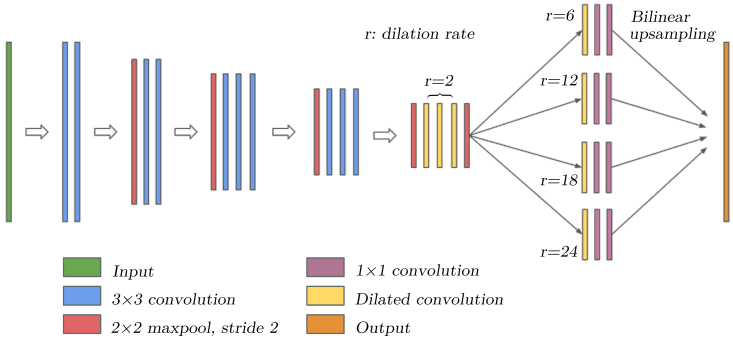
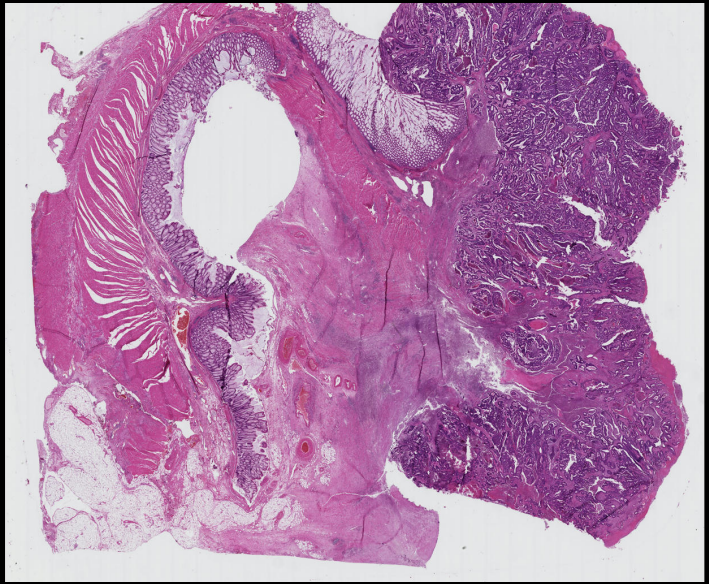


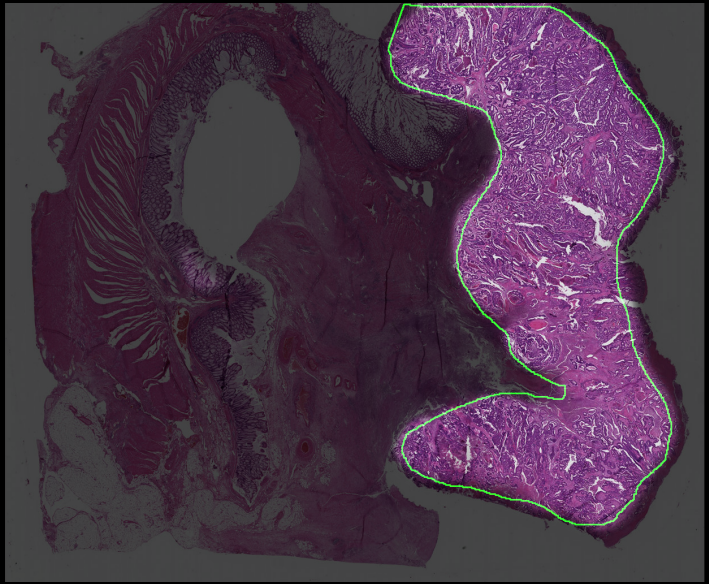
Figure 34: VGG16 version with atrous spatial pyramid pooling (ASPP)

Example: Tumor delineation — input



Example: Tumor delineation — prediction

Example: Tumor delineation — prediction and reference



Segmentation — argmax

- Normally, convnet-segmentation methods outputs a probability map for each class
- Need to assign *one* label to each pixel (segment the image)
- Could use arg-max over classes for each pixel
- Pros:
  - Simple
  - Fast
- Cons:
  - Ugly
  - Inaccurate
- Very common to use *Dense Conditional Random Fields*

- Fast implementation: Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials
- Use information from the output probability map
- Also takes into account features from the input image, such as color in neighbouring pixels

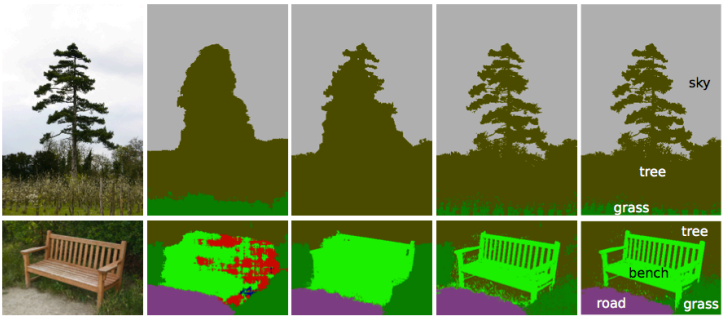
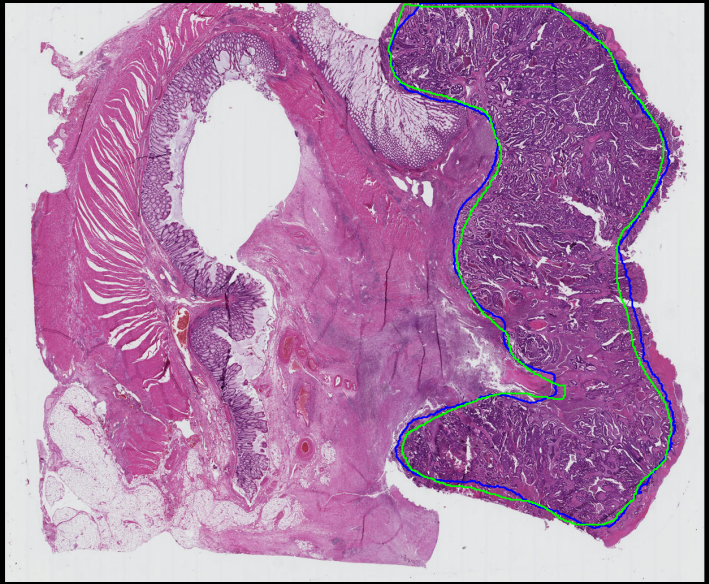


Figure 35: Source: [Krähenbühl and Koltun, 2011]

Example: Tumor delineation — segmentation and reference



INSTANCE SEGMENTATION

Instance segmentation

- Image segmentation with discrimination between instances of the same class
- Combines object detection and semantic segmentation
- More difficult than standard semantic segmentation

Figure 36: Top: Original. Bottom: Segmented. Image source: https://www.pexels.com

Mask R-CNN

- Publication: [He and Girshick, 2017]
- Extends Faster R-CNN
- Faster R-CNN outputs a class label and a bounding box offset for each detected region
- Mask R-CNN in addition outputs one object mask for each class
- The object mask is produced by a small segmentation network (e.g. FCN)
- The segmentation is performed independently in each class
- Substitutes the ROI pooling with a location-preserving ROI alignment layer

Mask R-CNN

Figure 37: Mask R-CNN results [He and Girshick, 2017]

References I

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint*, pages 1–10, 2016.

Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The P ASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88:303–338, 2009. doi: 10.1007/s11263-009-0275-4.

Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015.

Ross Girshick. Fast R-CNN. *arXiv preprint*, pages 1–9, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR2014*, pages 1–8, 2014.

References II

Junwei Han, Dingwen Zhang, Gong Cheng, Nian Liu, and Dong Xu. Advanced deep-learning techniques for salient and category-specific object detection. *IEEE Signal Processing Magazine*, 35:84–100, 2018.

Kaiming He and Ross Girshick. Mask R-CNN. *arXiv preprint*, pages 1–10, 2017.

P Krähenbühl and Vladlen Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials Supplementary Material. *arXiv preprint*, pages 1–4, 2011.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint*, pages 1–10, 2014.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv preprint*, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint*, pages 1–14, 2016.

References III

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint*, pages 1–8, 2015.

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

Questions?