Module 8: Supervised learning

5LSM0: Convolutional neural networks for computer vision

Fons van der Sommen

Electrical Engineering / VCA research group

Administrative

We're half-way!

Assignments

You are here!

- Assignment 1 due
- Assignment 2 online [due Mar-25]
- Start on final assignment as soon as you've finished assignment 2

Staff changes

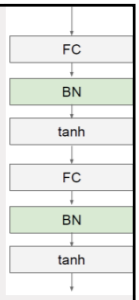
- Joost → Chengyang
  - *Pytorch expert & smart guy*

WK	Date		Module
6	Tue	Feb-5	1. Introduction
	Thu	Feb-7	2. Data-driven image classification
7	Tue	Feb-12	3. Loss functions and optimization
	Thu	Feb-14	4. Neural networks and backpropagation
8	Tue	Feb-19	5. Convolutional neural networks
	Thu	Feb-21	6. CNN architetures
9	Tue	Feb-26	7. Training neural networks (part I)
	Thu	Feb-28	7. Training neural networks (part II)
10			NO LECTURES
11	Tue	Mar-12	8. Supervised learning
	Thu	Mar-14	9. Unsupervised learning
12	Tue	Mar-19	10. Semi/self-supervised learning
	Thu	Mar-21	11. Temporal modeling & reinforcement learning
13	Tue	Mar-26	12. Visualization and understanding
	Thu	Mar-28	PAPER SESSIONS
14	Tue	Apr-2	13. Efficient deep learning
	Thu	Apr-4	Recap + question hours
15+16			EXAMS

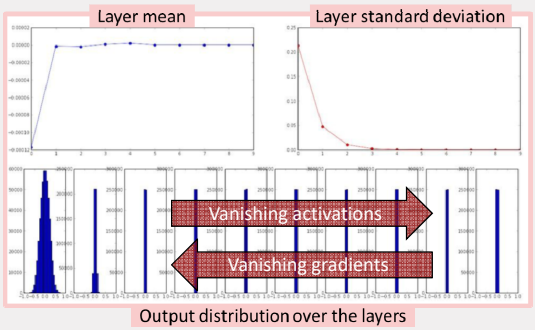
Last time(s): training neural networks

Batch norm

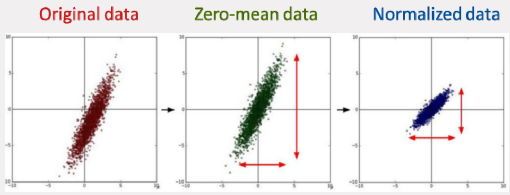
$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$



Weight initialization



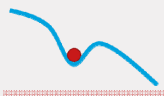
Output distribution over the layers



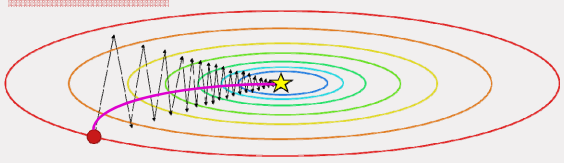
Data normalization

Last time(s): training neural networks

Saddle points



Local minima



Poor conditioning

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

SGD+ momentum

$$x_{t+1} = x_t - \alpha v_{t+1}$$
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

AdaGrad

$$x_{t+1} = x_t - \frac{\alpha \nabla f(x_t)}{g_{.}^{-1/2}}$$
$$g_{t+1,i} = \gamma g_{t,i} + (1-\gamma)\left(\frac{\partial L}{\partial w_i}\right)_t^2$$

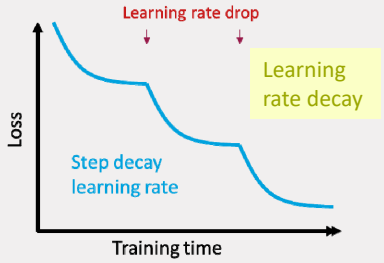
ADAM

$$x_{t+1} = x_t - \alpha \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon}$$

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla f(x_t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\left(\nabla f(x_t)\right)^2$$

Optimization

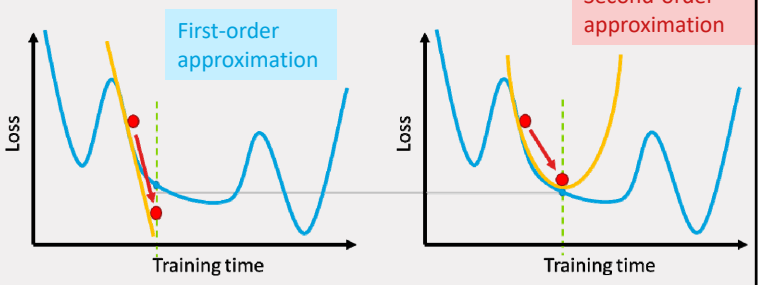
Last time(s): training neural networks



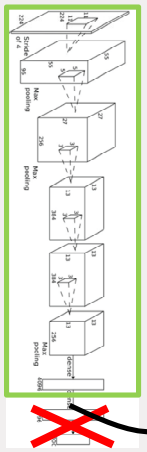
More optimization…

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \boldsymbol{H} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$



Last time(s): training neural networks



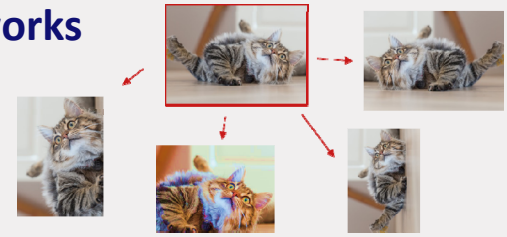
Re-use

Transfer learning

CNN codes

Conventional classifier (e.g. SVM)

Regularization



Data augmentation

Dropout

This time

Supervised learning!

Classification (recap)

- What image belongs to what category?

Segmentation

- What pixels belong to what category?

Detection

- What pixels belong to what instance of what category?

Supervised learning

So far: just classification

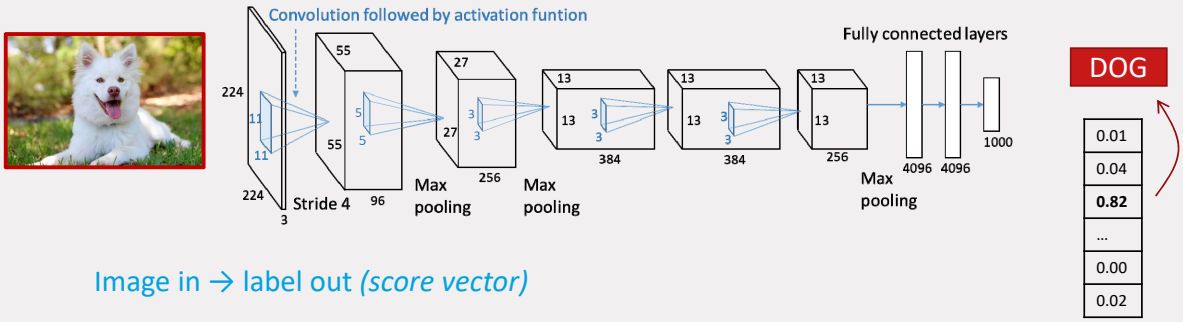
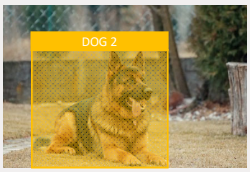


Image in → label out *(score vector)*

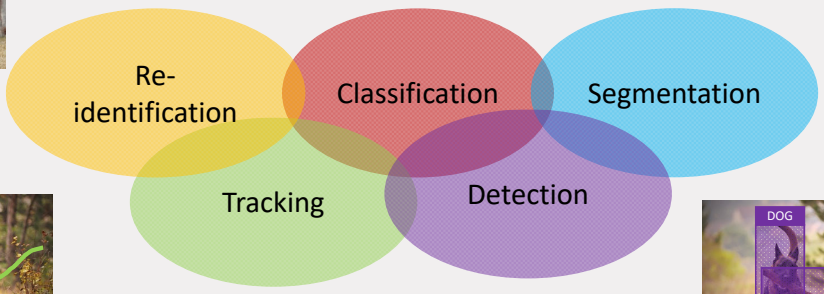
Typically an image shows more than a single class though…

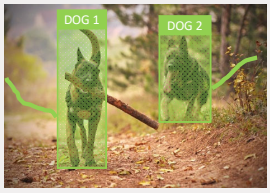
Images license free (www.pexels.com)

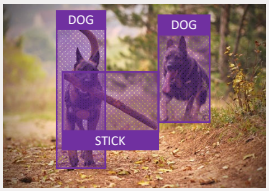
Supervised learning



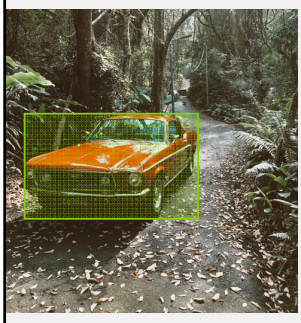
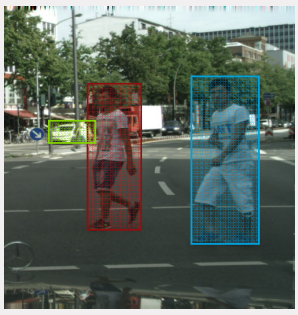
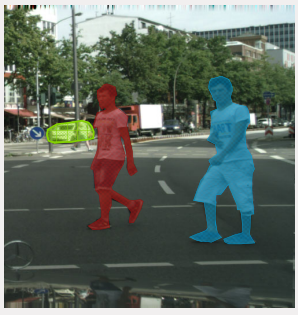
DOG







Supervised learning

Classification+localization	Object detection	Instance segmentation	Segmentic segmentation
			
Car	Pedestrian Car Pedestrian	Pedestrian Car Pedestrian	Truck Road Vegetation Pedestrian Sidewalk Car
Single object	Multiple objects		No objects, just pixels

Semantic segmentation

- Assign a class to each pixel in an image
- Do not distinguish between several instances of the same category

Pedestrian

Road

Truck

Car

Vegetation

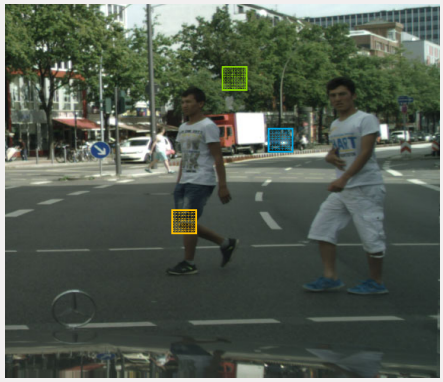
Sidewalk

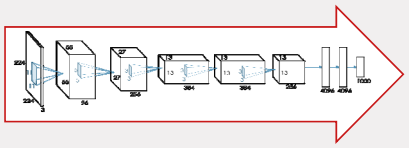
Q: Main difference with instance segmentation?

Semantic segmentation

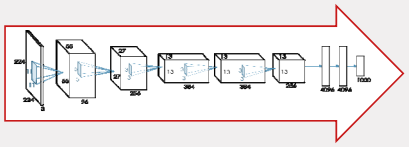
Q: Why is this not a good idea?

Idea: sliding window

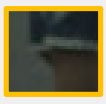


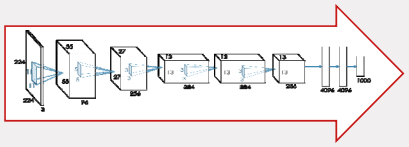


Car



Car





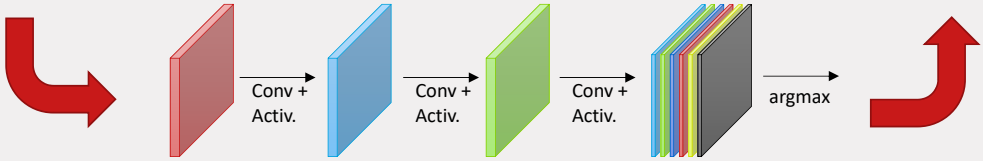
Pedestrian

Semantic segmentation

Q1: How could we characterize the loss in this case?

Q2: Problem in this case?

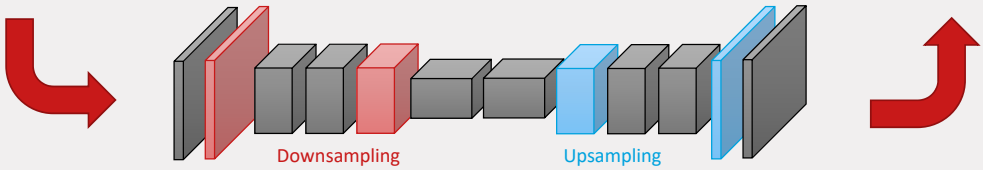
Idea: fully convolutional



TU/e

Semantic segmentation

Idea: fully convolutional

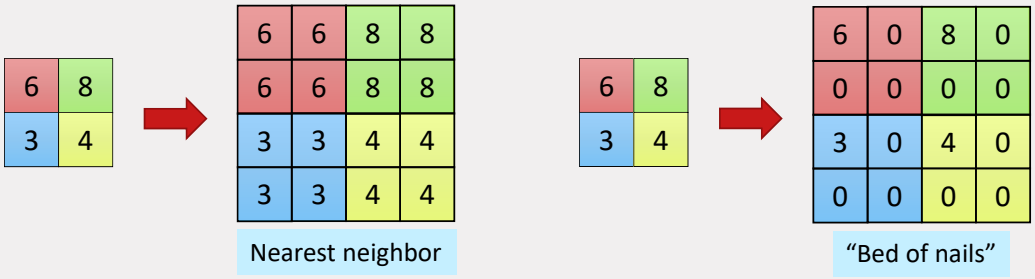


TU/e

Semantic segmentation

Upsampling… but how?

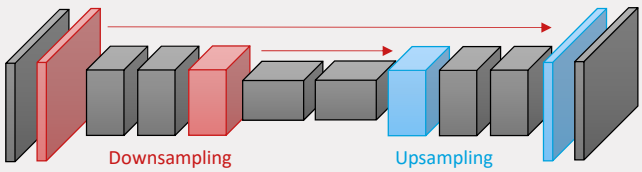
- Unpooling

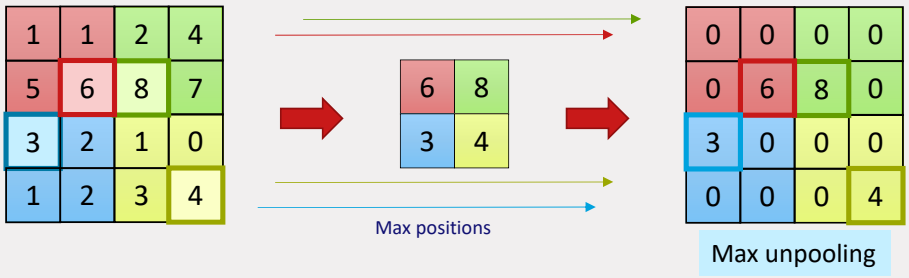


Semantic segmentation

Upsampling… but how?

- Unpooling



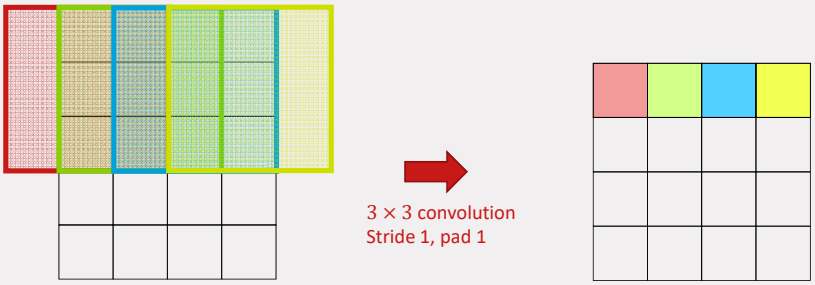


Semantic segmentation

Upsampling… but how?

- Unpooling
- Transpose convolution

Recall: normal convolution

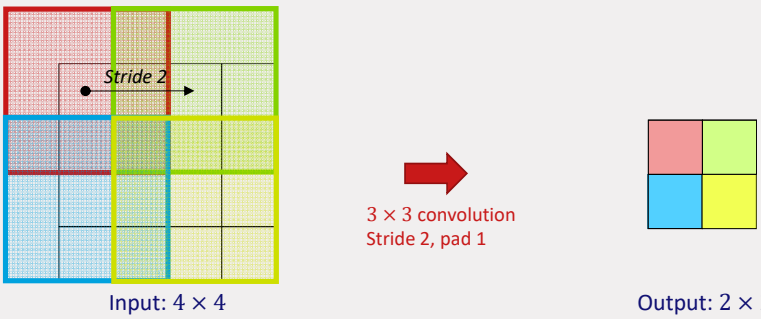


Semantic segmentation

Upsampling… but how?

- Unpooling
- Transpose convolution

Strided convolution



Semantic segmentation

Upsampling… but how?

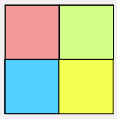
- Unpooling
- Transpose convolution

Q: Sometimes called *deconvolution*, why is this inadequate nomeclature?

Transpose convolution

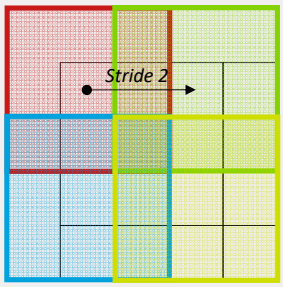
Multiply input value with filter kernel

Sum where overlaps



Output: 2 × 2

3 × 3 convolution
Stride 2, pad 1



Input: 4 × 4

Semantic segmentation

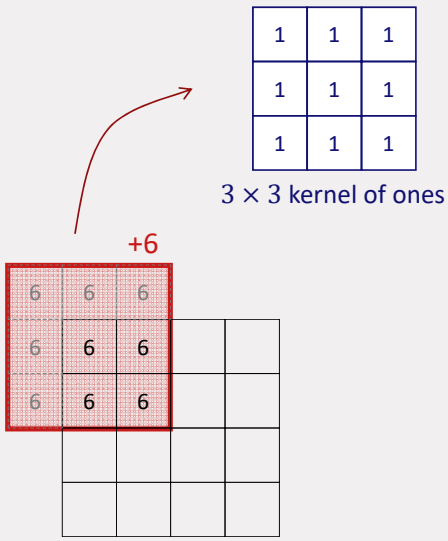
Upsampling… but how?

- Unpooling
- Transpose convolution

Transpose convolution

Output: 2 × 2

transpose convolution
stride 2, pad 1
3 × 3 kernel of ones



Input: 4 × 4

Semantic segmentation

Upsampling… but how?

- Unpooling
- Transpose convolution

Transpose convolution

1	1	1
1	1	1
1	1	1

3 × 3 kernel of ones

6	8
3	4

Output: 2 × 2

transpose convolution
stride 2, pad 1
3 × 3 kernel of ones

+8

6	6	14	8	8
6	6	14	8	8
6	6	14	8	8

Input: 4 × 4

Semantic segmentation

Upsampling… but how?

- Unpooling
- Transpose convolution

Transpose convolution

1	1	1
1	1	1
1	1	1

3 × 3 kernel of ones

6	8
3	4

Output: 2 × 2

transpose convolution
stride 2, pad 1
3 × 3 kernel of ones

+3

6	6	14	8	8
6	6	14	8	8
9	9	17	8	8
3	3	3		
3	3	3		

Input: 4 × 4

Semantic segmentation

Upsampling… but how?

- Unpooling
- Transpose convolution

Transpose convolution

1	1	1
1	1	1
1	1	1

3 × 3 kernel of ones

6	8
3	4

Output: 2 × 2

transpose convolution
stride 2, pad 1
3 × 3 kernel of ones

6	6	14	8	8
6	6	14	8	8
9	9	21	12	12
3	3	7	4	4
3	3	7	4	4

+4

Input: 4 × 4

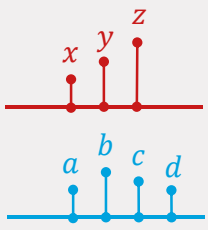
Semantic segmentation

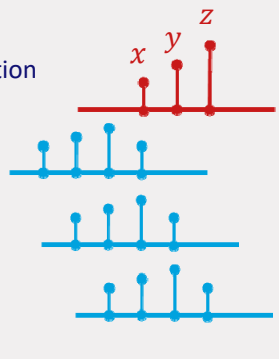
Q: Output size of this convolution?

Convolution as matrix multiplication

- We can write a convolution as a matrix multiplication

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \boldsymbol{q} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$





ax

$bx + ay$

$cx + by + az$

$\boldsymbol{y} = \boldsymbol{p} * \boldsymbol{q}$ $\quad y_n = \sum_{k=-\infty}^{\infty} p_k q_{n-k}$

dz

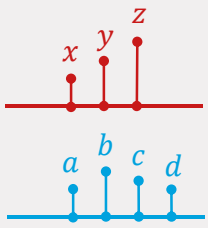
Semantic segmentation

Convolution as matrix multiplication

- We can write a convolution as a matrix multiplication

$$\boldsymbol{p} * \boldsymbol{q} = \boldsymbol{Q}\boldsymbol{p}_{00}$$ ← *Zero-padded*

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \boldsymbol{q} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$



$$\begin{bmatrix} d & c & b & a & 0 & 0 & 0 & 0 & 0 \\ 0 & d & c & b & a & 0 & 0 & 0 & 0 \\ 0 & 0 & d & c & b & a & 0 & 0 & 0 \\ 0 & 0 & 0 & d & c & b & a & 0 & 0 \\ 0 & 0 & 0 & 0 & d & c & b & a & 0 \\ 0 & 0 & 0 & 0 & 0 & d & c & b & a \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ x \\ y \\ z \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ bx + ay \\ cx + by + az \\ dx + cy + bz \\ dy + cz \\ dz \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{p} * \boldsymbol{q} \qquad y_n = \sum_{k=-\infty}^{\infty} p_k q_{n-k}$$

Semantic segmentation

Transpose convolution as matrix multiplication

$$\boldsymbol{p} *^T \boldsymbol{q} = \boldsymbol{Q}^T \boldsymbol{p}_{00} \qquad = \boldsymbol{p} \star \boldsymbol{q}$$

For stride = 1

$$\begin{bmatrix} d & 0 & 0 & 0 & 0 & 0 \\ c & d & 0 & 0 & 0 & 0 \\ b & c & d & 0 & 0 & 0 \\ a & b & c & d & 0 & 0 \\ 0 & a & b & c & d & 0 \\ 0 & 0 & a & b & c & d \\ 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & 0 & a & b \\ 0 & 0 & 0 & 0 & 0 & a \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} dx \\ cx + dy \\ bx + cy + dz \\ ax + by + cz \\ ay + bz \\ az \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Q: Difference between convolution and cross-correlation?

Semantic segmentation

Q: How is this different from a convolution?

No convolution (cross-correlation) anymore!!

Transpose convolution as matrix multiplication

$$\begin{bmatrix} d & c & b & a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & d & c & b & a & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d & c & b & a & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ x \\ y \\ z \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ cx + by + az \\ dy + cz \end{bmatrix}$$

Strided convolution

$\boldsymbol{p} * \boldsymbol{q} = \boldsymbol{Q}\boldsymbol{p}_{00}$

Stride 2

Strided transpose convolution

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

$\boldsymbol{p} *^T \boldsymbol{q} = \boldsymbol{Q}^T\boldsymbol{p}_{00}$

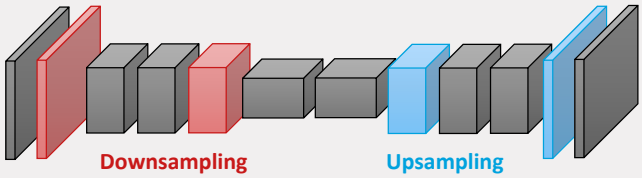
Stride 2

TU/e

Semantic segmentation

Q: Fundamental difference between unpooling and transpose convolution?

Fully convolutional network (FCN)



- Pooling
- Strided convolution
- Unpooling
- Strided transpose convolution

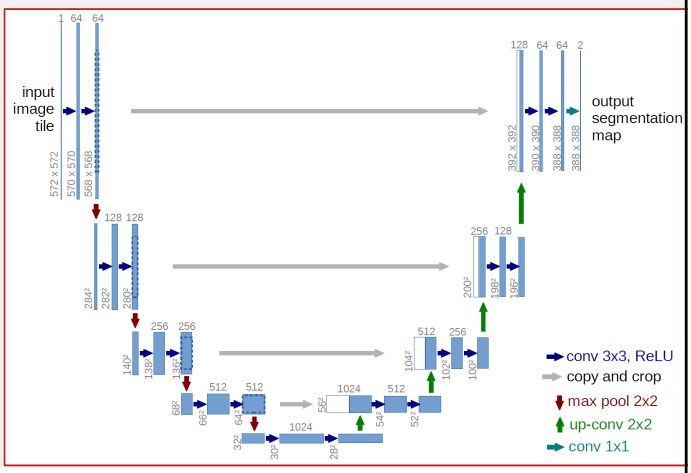
TU/e

Semantic segmentation

The U-net architecture

- Problem with conventional FCNs:
  - *Sharp edge information is lost*
- Solution:
  - *Append upstream convolution maps with downstream convolution maps of the same scale / size*
  - *Fine-grained edge information preserved*

Ronneberger et al., MICCAI 2015
-> Most cited MICCAI paper, runner up from 1998…
(4,568 vs 3,310 @ March 10, 2019)



Classification + localization

Image classification

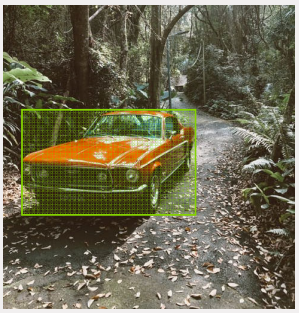
- Category label assigent to the image

Classification + localization

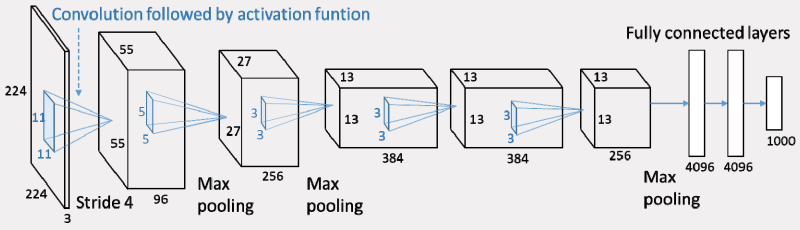
- Predict category lablel
- New: Where is the <category> in that image?

Example:

- Image labeled as CAR
- Draw a bounding box around the car within the image



Classification + localization



Output vector

1000

0.07 Boat
0.81 Car
0.02 Tree
... ...

(x, y, w, h)

Class scores → Softmax loss
Given: correct label

Box coordinates → L2 loss
Given: correct position

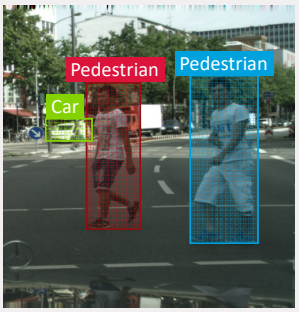
Q: What happens if we just add these losses?

Object detection

Fixed set of categories that we're interested in

Every time that one of those categories appears in an input image:

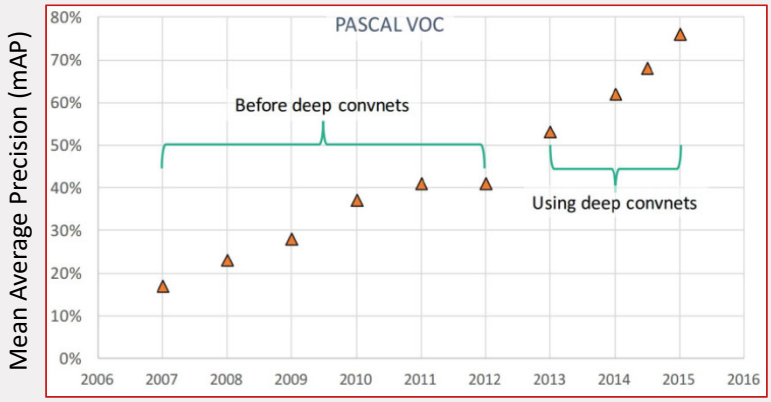
- put a bounding box around it and classify it.



Why can't we use the classification + localication approach from the previous slides?

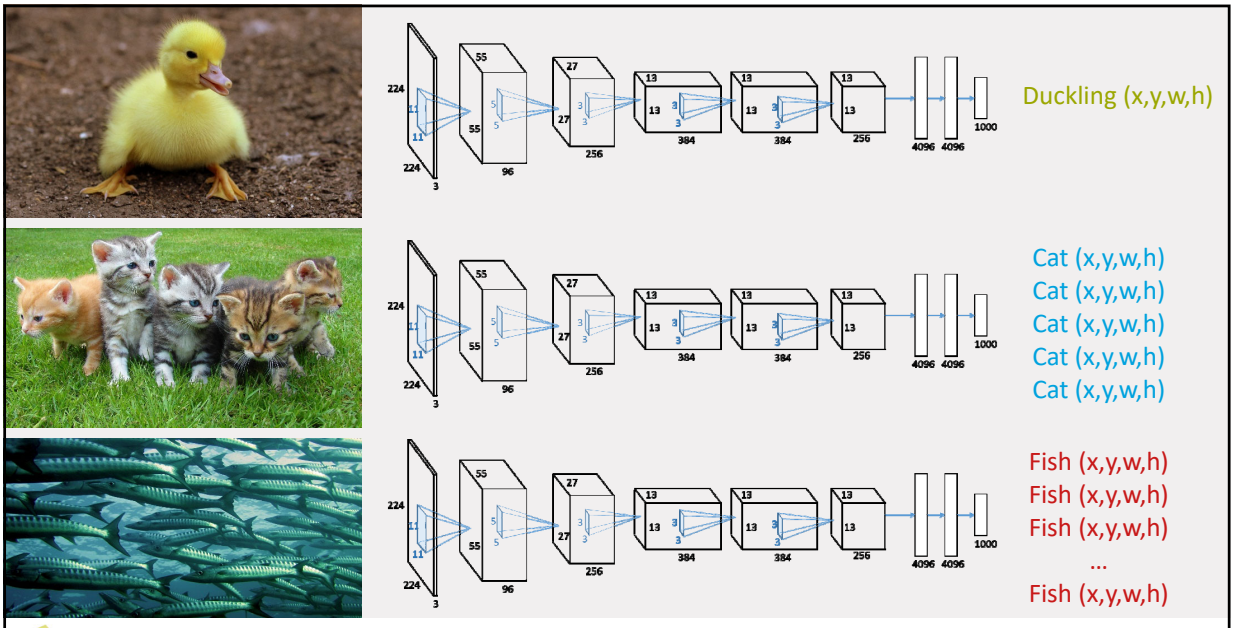
- Problem: we don't know how many objects of interest to expect in the image
- Hence, we don't know how many bounding boxes to estimate and classify…

Object detection



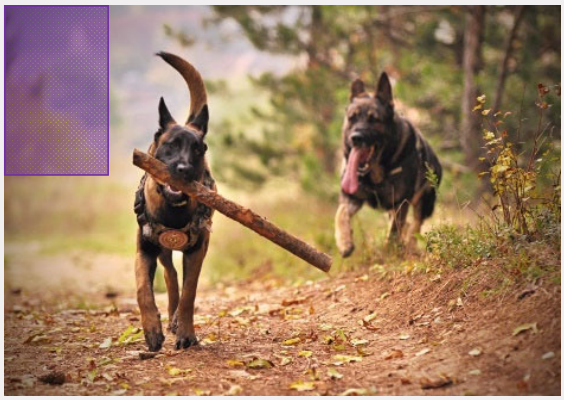
Current state-of-the-art well over 80%

*Example from cs231n (2017): lecture 11 – slide 54



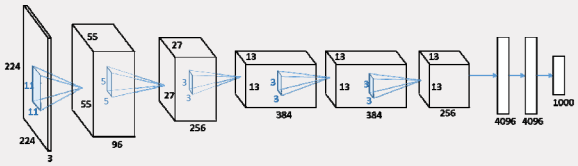
Object detection

Other approach: sliding window



Move a window over the image and classify each window independently:

- + Add category for background



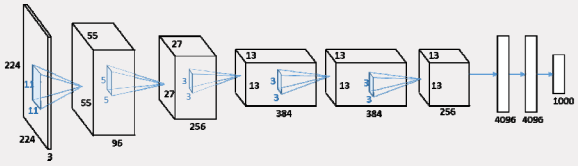
Cat? NO
Dog? NO
Stick? NO
Background? YES

Object detection

Other approach: sliding window

Move a window over the image and classify each window independently:

- + Add category for background



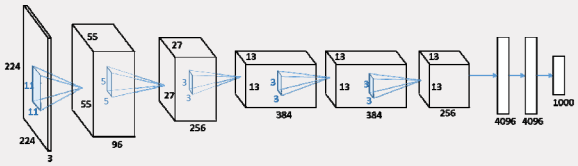
Cat? NO
Dog? YES
Stick? NO
Background? NO

Object detection

Other approach: sliding window

Move a window over the image and classify each window independently:

- + Add category for background



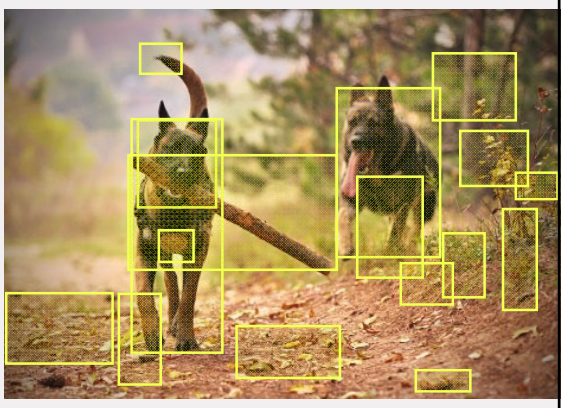
Cat?	NO
Dog?	YES
Stick?	MAYBE?
Background?	NO

Q: Problem(s) with this approach?

Object detection

Region proposals

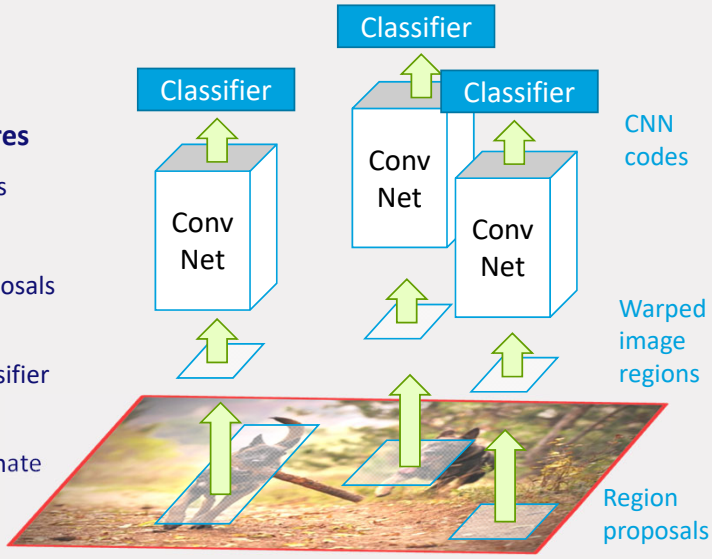
- Use conventional (fast) image processing techniques to find potential locations of objects
  - *Regions in which an object may be present*
  - *Candidate proposal regions*
- Typically a lot of proposals are generated
  - *The majority of them may not actually fully contain an object*
  - *Very high recall → if there are objects in the image, some proposals will contain them*



Object detection

R-CNN: Regions with CNN features

- Find regions using region proposals
- Warp images to normalize size
- Compute features of warped proposals with pre-trained CNN
- Feed features to conventional classifier (e.g. SVM) to classify the region
- In addition, use regression to estimate a correction for the bounding box



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

Object detection

Problems with R-CNN

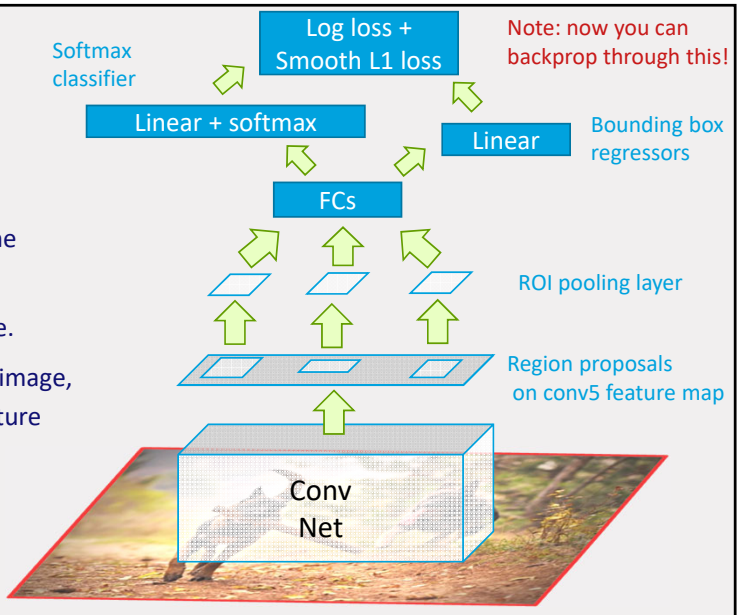
- Training is very slow and memory heavy…
  - *Features of region candidates typically dumped to disk*
- Inference is slow…
  - *A lot of candidate regions which have to be classified*
- Different types of losses for training
  - *E.g. SVM loss for classification, Least squares regression loss, fine-tune network with softmax*
- Convolution results are not shared/re-used
- Region proposals are not learned…

Object detection

Fast R-CNN!

- First run entire image through some convolutional layer(s).
- Region proposals for original image.
- Rather than taking crops from the image, project the proposals onto the feature map and take crops from there.

Q: Why would this strategy fail for the FC layers?



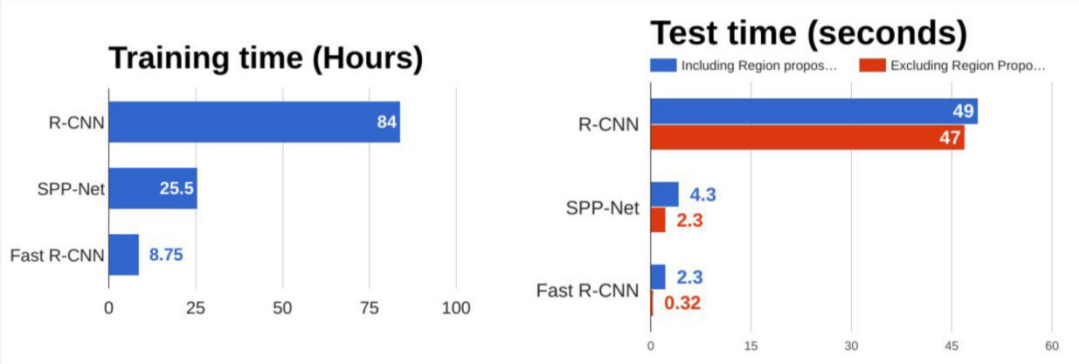
Object detection

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

He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014

Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN: how fast?



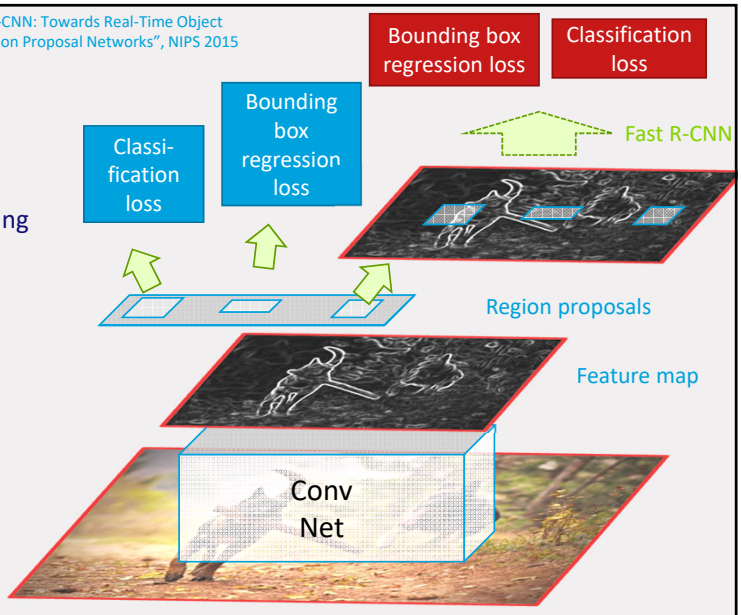
*Example from cs231n (2017): lecture 11 – slide 79

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

Object detection

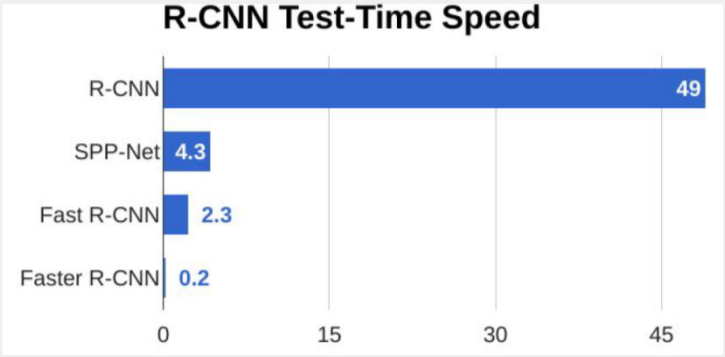
Faster R-CNN!!

- Bottleneck of Fast R-CNN: computing region proposals with external function
- Solution: insert Region Proposal Network (RPN) to come up with region proposals
- Jointly train with four losses
  - *RPN predict object? [Y/N]*
  - *RPN box coordinates?*
  - *Final class scores?*
  - *Final box coordinates?*



Object detection

Faster R-CNN: how fast?



**Example from cs231n (2017): lecture 11 – slide 82*

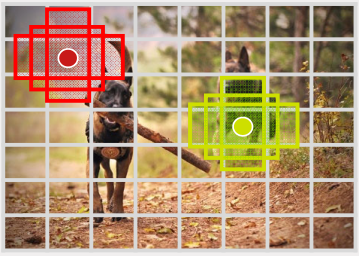
Object detection

So far: region based methods

- First propose a set of regions, then process each region independently

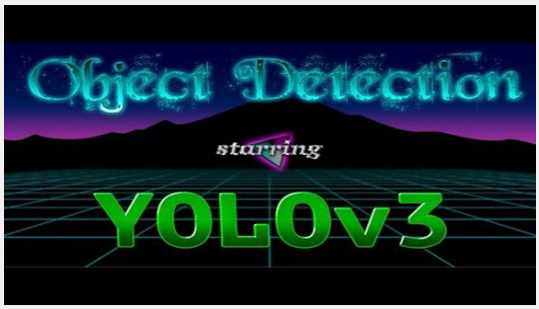
Alternative: all feed-forward in a single pass

- YOLO: You Only Look Once / SSD: Single Shot Detection

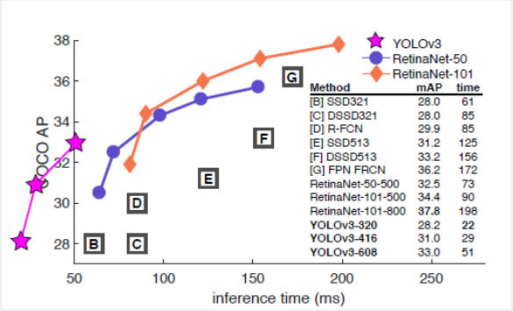


- Divide image into grid
- Use a base set of bounding boxes for each cell in the grid
- Solve one regression problem:
- For each bounding box predict:
  - *True location + confidence*
  - *Class scores*

Object detection



https://pjreddie.com/darknet/yolo/



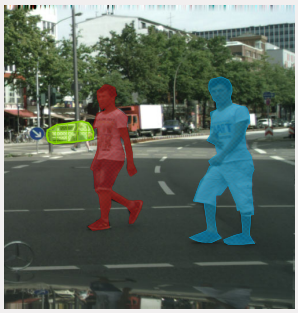
Redmon and Farhadi, *"Yolov3: An incremental improvement." arXiv preprint* arXiv:1804.02767 (2018).

Instance segmentation

Segmentic segmentation

Pedestrian

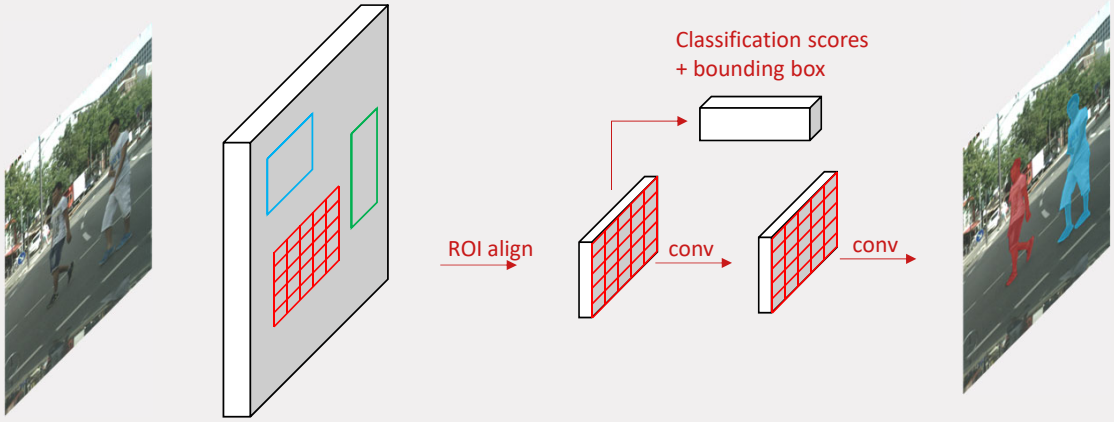
Instance segmentation



Pedestrian 1

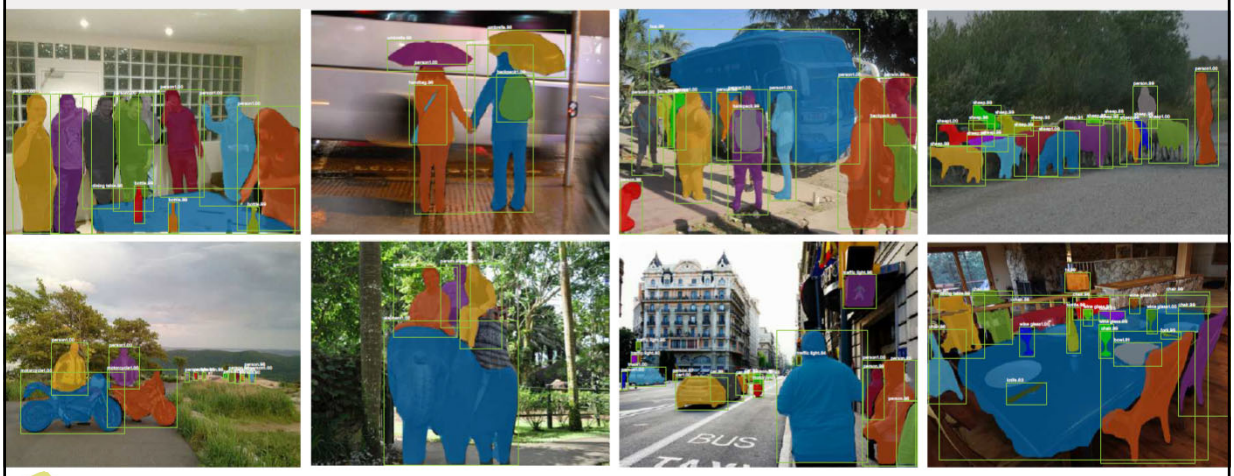
Pedestrian 2

Instance segmentation



He, Kaiming, et al. "Mask r-cnn." ICCV 2017.

Instance segmentation



Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, "Mask R-CNN", The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969 (link)

Summary

Supervised learning

- Classification
  - *Predict category of an image*
- Semantic segmentation
  - *Predict category of pixels within an image*
- Classification + localization
  - *Predict category of an image + bounding box of the location of the object within the image*
- Object detection
  - *Detect multiple instances of different categories within a single image*
- Instance segmentation
  - *Detect and segment multiple instances of different categories within a single image*

Next time

Wait a minute.. what if we ain't got no labels?

UNSUPERVISED LEARNING