Machine Intelligence & Deep Learning Workshop

Raymond Ptucha, Majid Rabbani, Mark Smith

The Kate Gleason COLLEGE OF ENGINEERING

Regional CNNs

Raymond Ptucha

June 27-29, 2018

Rochester Institute of Technology

www.rit.edu/kgcoe/cqas/machinelearning

Fair Use Agreement

This agreement covers the use of all slides in this document, please read carefully.

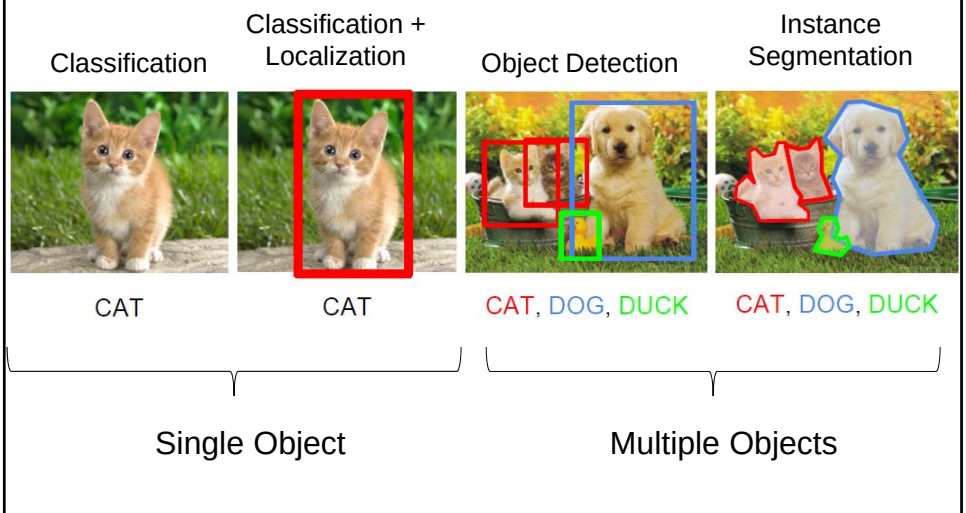
- You may freely use these slides for personal use, if:
  - My name (R. Ptucha) appears on each slide.
- You may freely use these slides externally, if:
  - You send me an email telling me the conference/venue/company name in advance, and which slides you wish to use.
  - You receive a positive confirmation email back from me.
  - My name (R. Ptucha) appears on each slide you use.

(c) Raymond Ptucha, rwpeec@rit.edu

Agenda

- Wed, June 27
  - 9-10:30am — Regression and Classification
  - 10:30-10:45pm — Break
  - 10:45-12:15pm — Boosting and SVM
  - 12:15-1:30pm — Lunch
  - 1:30-3:30pm — Neural Networks and Dimensionality Reduction
  - 3:30-5pm — Hands-on Python and Machine Learning
- Thur, June 28
  - 9-10:30am — Introduction to deep learning
  - 10:30-10:45pm — Break
  - 10:45-12:15pm — Convolutional Neural Networks
  - 12:15-1:30pm — Lunch
  - 1:30-3:30pm — **Region and pixel-level convolutions**
  - 3:30-5pm — Hands-on CNNs
- Fri, June 29
  - 9-10:30am — Recurrent neural networks
  - 10:30-10:45pm — Break
  - 10:45-12:15pm — Language and Vision
  - 12:15-1:30pm — Lunch
  - 1:30-3:30pm — Graph convolutional neural networks; Generative adversarial networks
  - 3:30-5pm — Hands-on regional CNNs, RNNs

Vision Tasks



Classification vs. Classification + Localization

Classification
- **Input**: Image
- **Output**: Class label
- **Evaluation metric**: Accuracy

→ CAT

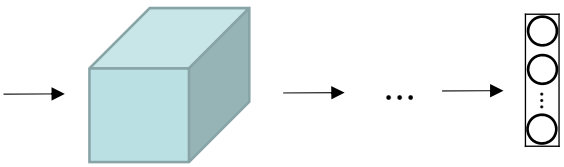
Classification + Localization
- **Input**: Image
- **Output**: Class label, Box coordinates
- **Evaluation metric**: Intersection over Union (IoU)

→ (CAT,x,y,w,h)

Classification with Localization

(0,0)

(1,1)



Lets allow a few classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

- For now, lets assume one object per image.
- Each object has {x, y, w, h}
- For this image, object location {x, y, w, h} = {0.3, 0.6, 0.4, 0.3}

Image from: deeplearning.ai, C4W3L01

Classification with Localization

Four classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

Localization {x, y, w, h}

Define *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

- P_c: Probability of an object
- b_x, b_y, b_w, b_h: Bounding box location
- C_1, C_2, C_3, C_4: 0/1 for each class

Image from: deeplearning.ai, C4W3L01

Classification with Localization

Four classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

Localization {x, y, w, h}

Cost function (squared error):

$$Loss = \sum_{i=1}^{9} (\hat{y}_i - y_i)^2$$ If y_1=1

$$Loss = (\hat{y}_1 - y_1)^2$$ If y_1=0

$$y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

? = don't care

Image from: deeplearning.ai, C4W3L01

Classification with Localization

Four classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

Localization {x, y, w, h}

Alternate cost function:
- y_1 can be logistic loss
- $y_2 \rightarrow y_5$ can be squared error
- $y_6 \rightarrow y_9$ can be softmax cross entropy

$$y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

? = don't care

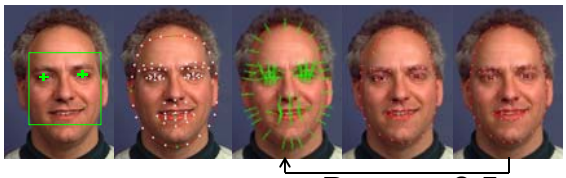
Image from: deeplearning.ai, C4W3L01

Snapchat Facewarp?

- Traditional approach:

Viola Jones Face Detection

Search for actual point locations using Mahalanobis distance



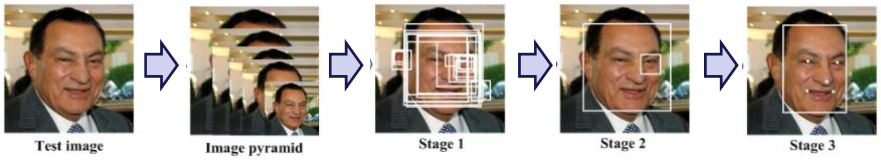
Repeat ~3-5×

Average eye and 82 facial feature points

Restrict based on PCA statistics

Snapchat Facewarp?

- Deep Learning approach:



Candidate faces

Refine face selection

Facial feature points

MT-CNN [Zhang et al. 2016]

Localization

- Facial feature

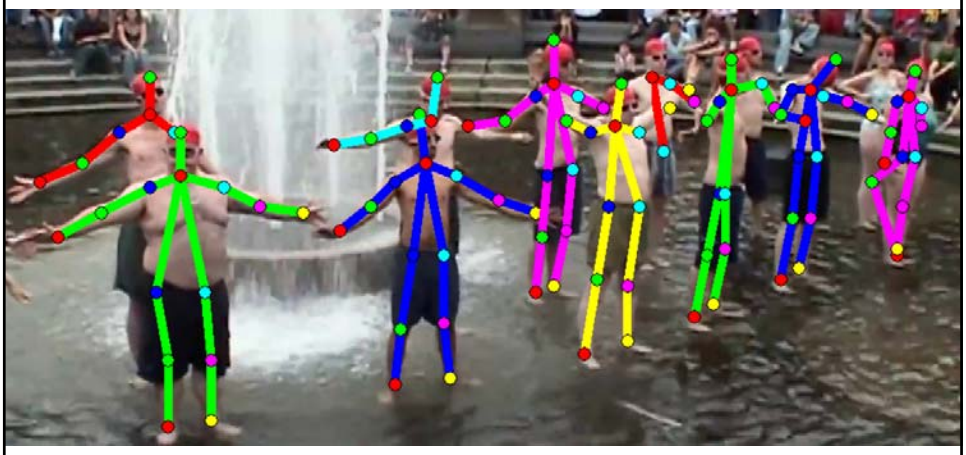
Each face has 68 points, so CNN would output:

Face?
pt1X
pt1Y
pt2X
pt2Y
.
.
.
pt68X
pt68Y

137 outputs

Of course, need GT for thousands of faces to train model.

Can do same with Body Pose…



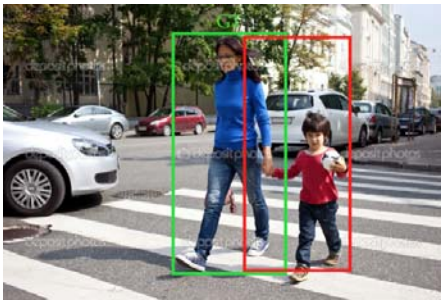
Pishchulin et al. CVPR'16

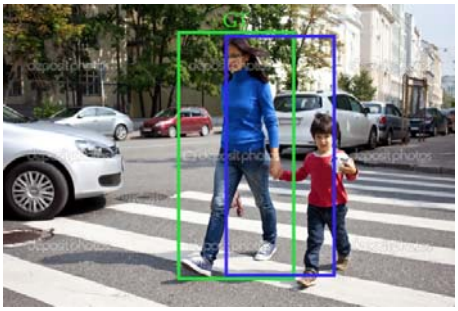
13

Intersection Over Union

- Intersection Over Union (IOU)

Task: Find adults, Green=GroundTruth, Red=FP, Blue=TP





But why is Red FP and Blue TP?

14

Intersection Over Union

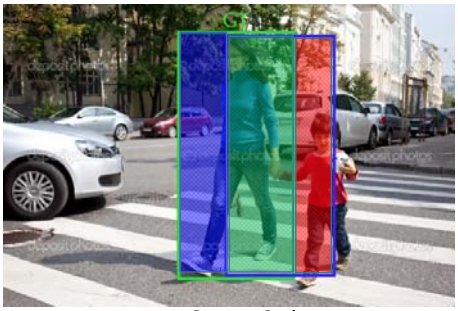
- Intersection Over Union (IOU): TP/(TP+FP+FN)

  TP= region of detected bbox that is part of GT

  FP = region of detected bbox that is not part of GT

  FN= GT region missed by detected bbox

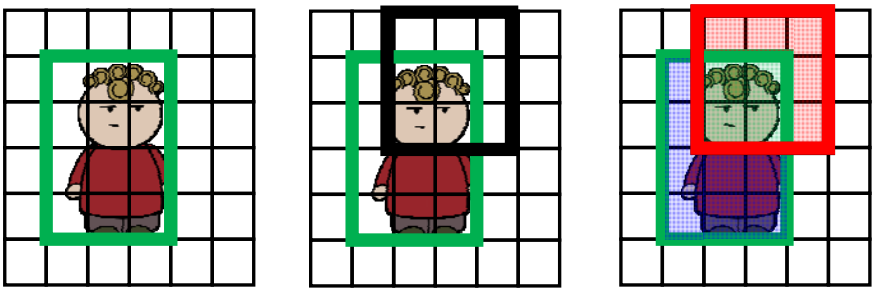
IOU=25%



IOU=50%

Bounding boxes with IOU> 50% are generally considered true positive.
Note: Any pixels marked void (two class, unsure class) are ignored

Intersection over Union



$$IOU = \frac{TP}{TP + FP + FN} = \frac{4}{4 + 5 + 8} = 23.5\%$$

Object Detection

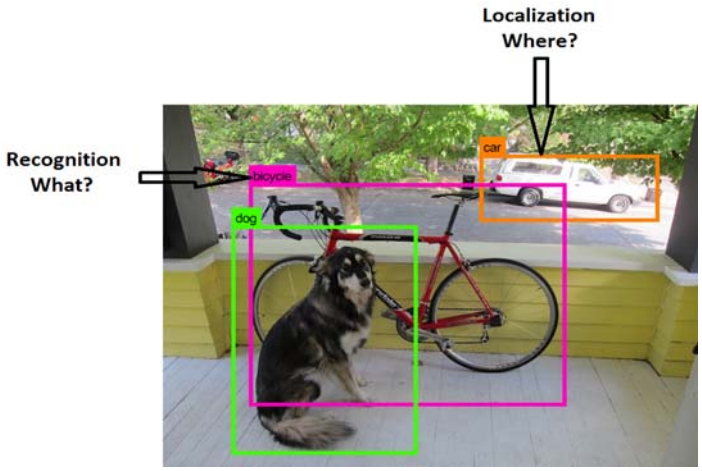


Image Credit: Redmon, Joseph, et al [4]

More than one object per image?

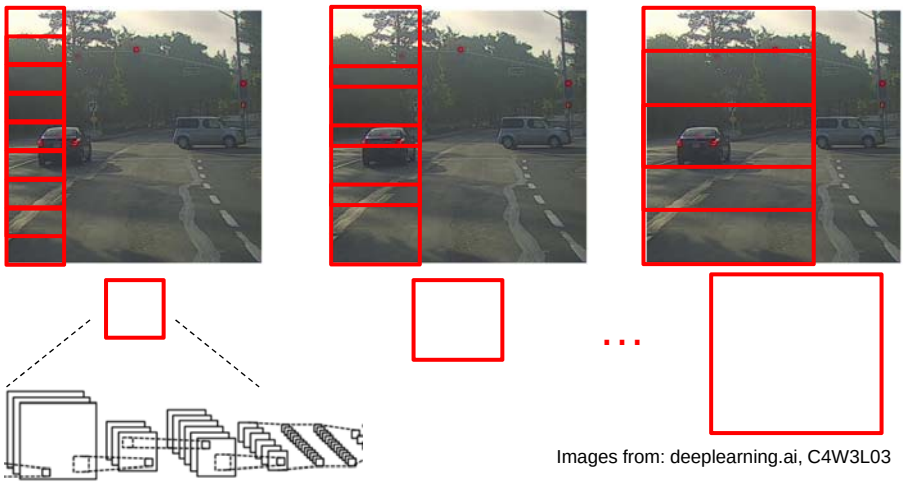
Training set:

x	y
	1
	1
	1
	0
	0

Car detection example

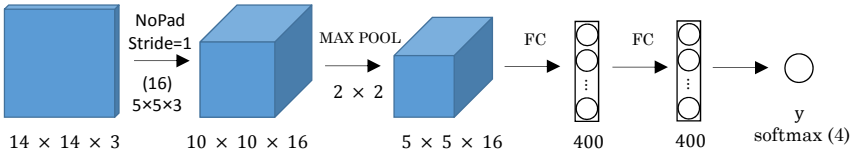
Images from: deeplearning.ai, C4W3L03

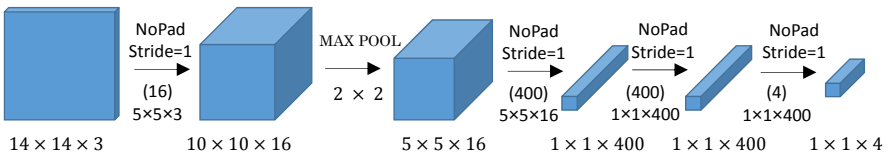
Sliding Window Detection



Images from: deeplearning.ai, C4W3L03

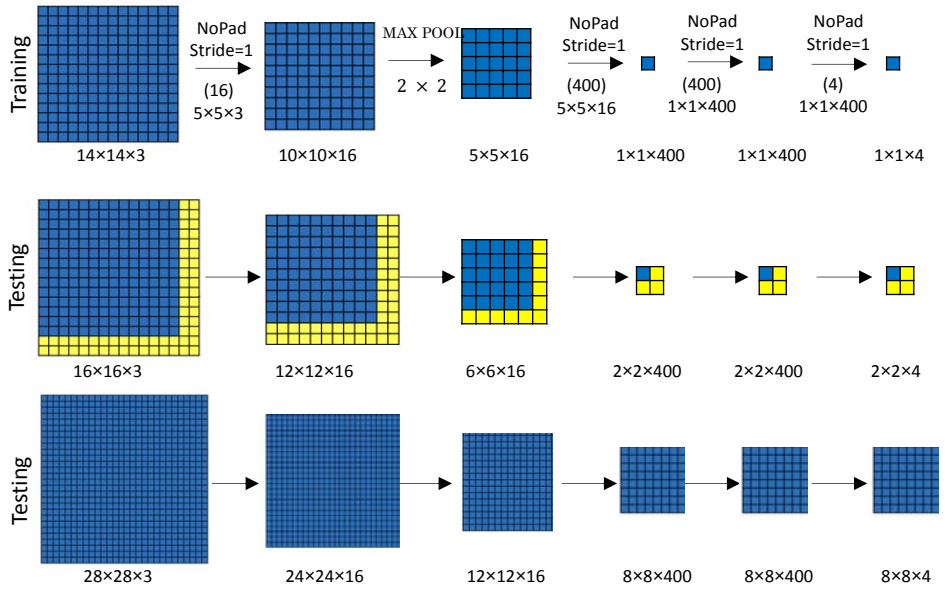
Computing FC layers with Convolution





Images from: deeplearning.ai, C4W3L04

Replacing Sliding Windows w/Fully Convolutional CNNs

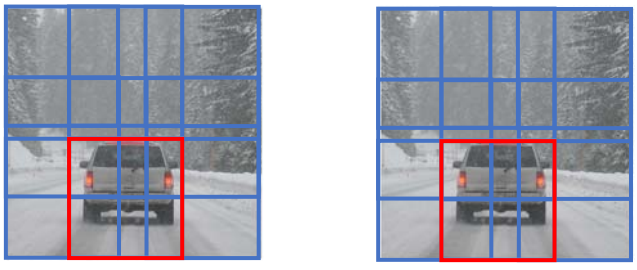


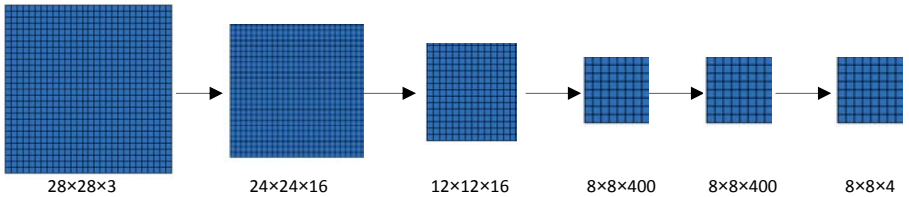
Images from: deeplearning.ai, C4W3L04

Replacing Sliding Windows w/Fully Convolutional CNNs

Sliding window approach:
Sequentially evaluate one window at a time

Fully convolutional approach:
Evaluate 64 regions at once

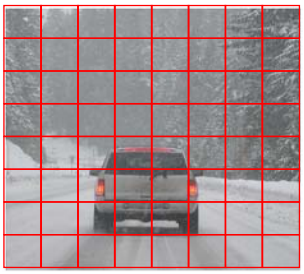




Images from: deeplearning.ai, C4W3L04

Replacing Sliding Windows w/Fully Convolutional CNNs

- Can think of this as evaluating 8×8 grid, where each of the 64 cells is independently checked for an object:

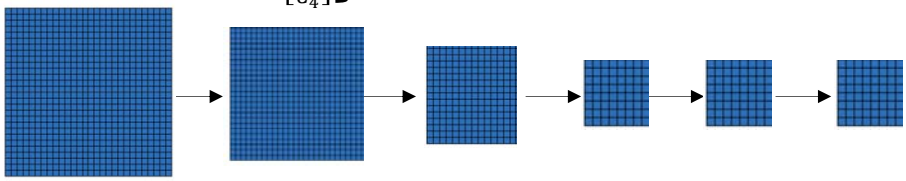


Each cell has a *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object

Object location

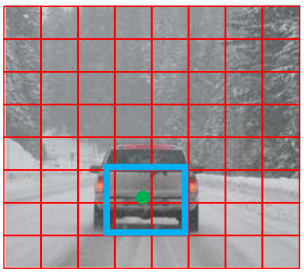
0/1 for each class
(Four classes in this example)



25

Replacing Sliding Windows w/Fully Convolutional CNNs

- Overlay GT of object
- Cell where centroid lie is responsible.

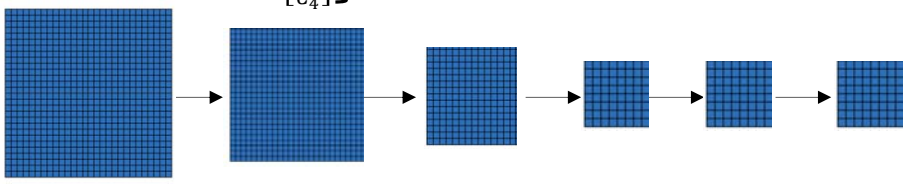


Each cell has a *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object

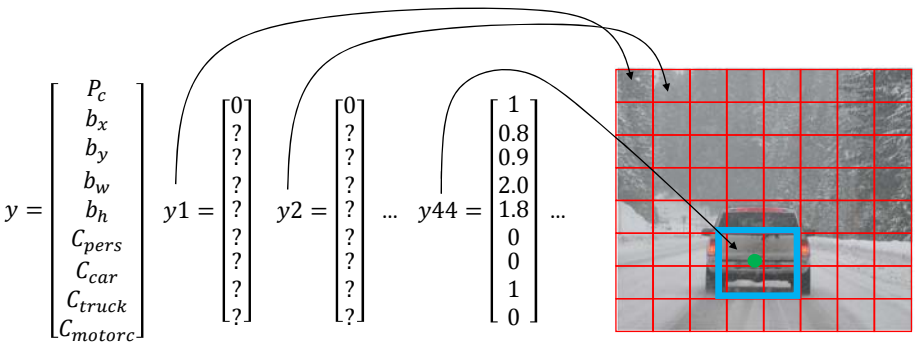
Object location

0/1 for each class
(Four classes in this example)

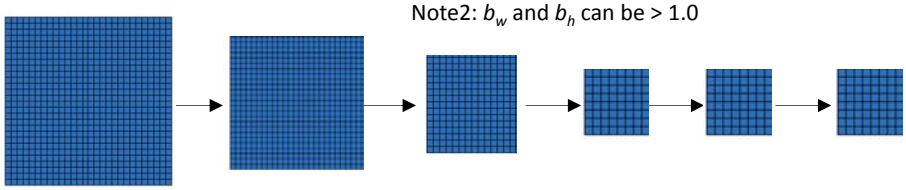


26

Replacing Sliding Windows w/Fully Convolutional CNNs



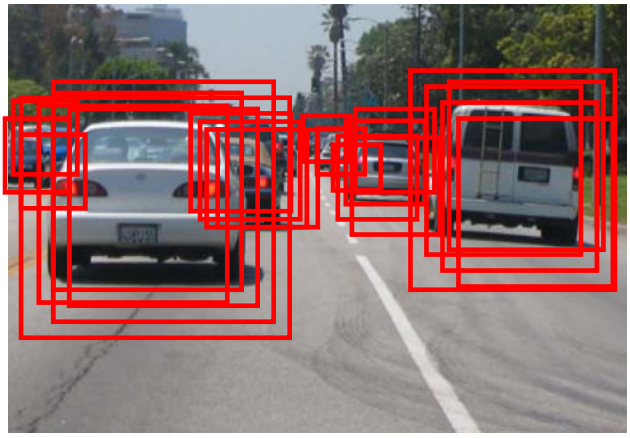
Note1: cell upper left (0,0); cell lower right (1,1)
Note2: b_w and b_h can be > 1.0



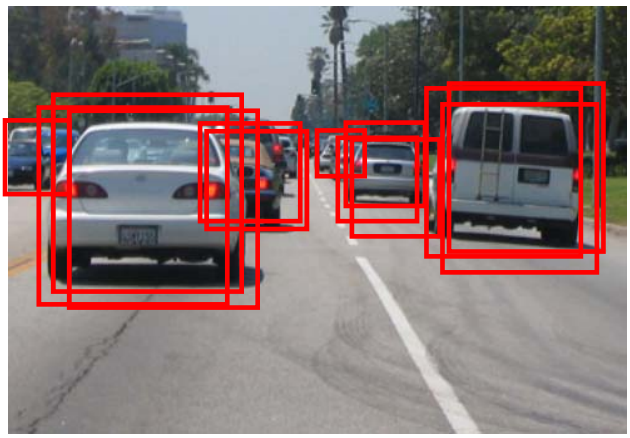
Non-max Suppression

- With sliding window or fully convolutional methods, we often see multiple detections for each object.
- For each object class:
  - Non-max suppression first discards objects with low detection probability;
  - Then, for remaining boxes:
    - Select box with maximum probability
    - Discard all other boxes with IoU>0.5 with selected box
    - Repeat (at Then, for …)

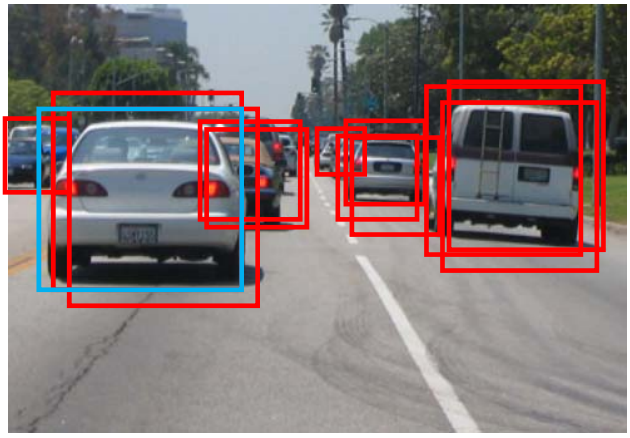
Non-max Suppression- Start



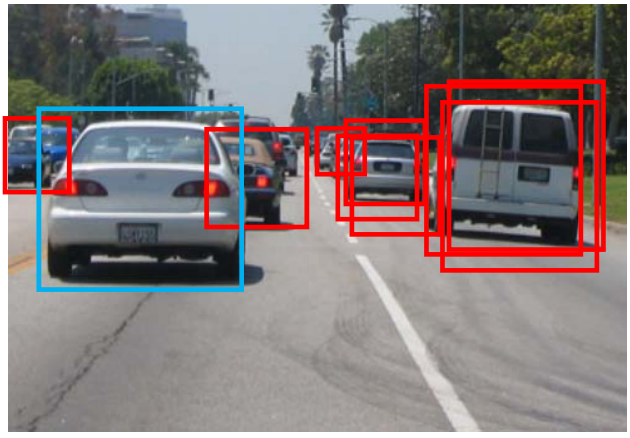
Non-max Suppression- Remove all boxes P()< 0.6



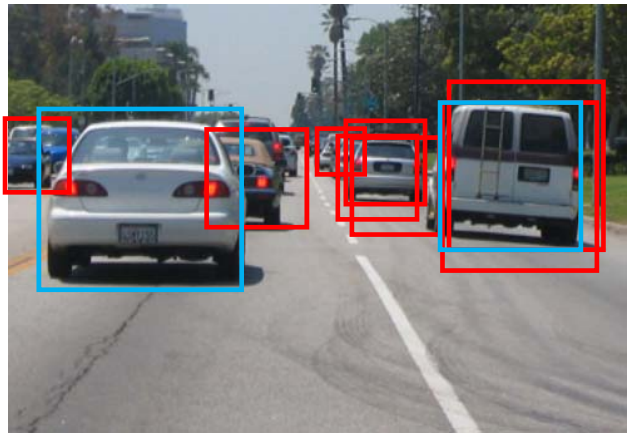
Non-max Suppression- Find Highest P()



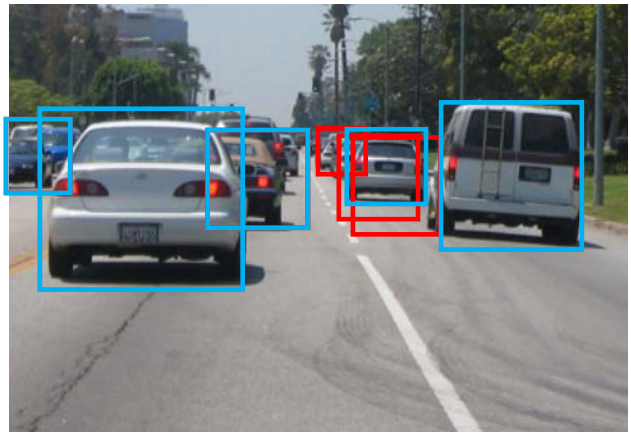
Non-max Suppression- Remove bounding boxes with IoU > 0.5 with blue box.



Non-max Suppression- Repeat.



Non-max Suppression- Repeat.



Region Proposals

- Fully convolutional great, but limits on max number of locations and objects per any location
- Image Blobs likely to contain objects
- Class-independent detector

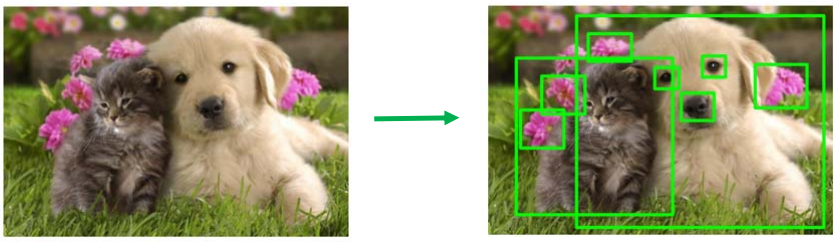
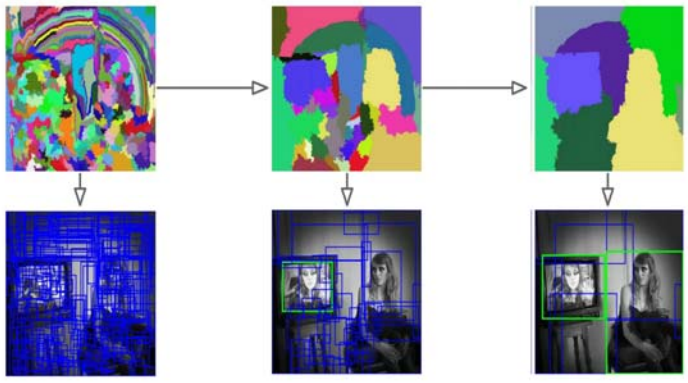


Image Credit: Dr. Fei-Fei Li, Andrej Karpathy

Region Proposals- Selective Search

Convert Regions to Boxes



Uijlings et al., "Selective search for Object Recognition", IJCV 2013.

Region Proposals

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea-tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	⋆⋆⋆	⋆	·
CPMC [19]	Grouping	✓	✓	✓	250	-	⋆⋆	⋆
EdgeBoxes [20]	Window scoring		✓	✓	0.3	⋆⋆	⋆⋆⋆	⋆⋆⋆
Endres [21]	Grouping	✓	✓	✓	100	-	⋆⋆⋆	⋆⋆
Geodesic [22]	Grouping	✓		✓	1	⋆	⋆⋆⋆	⋆⋆
MCG [23]	Grouping	✓	✓	✓	30	⋆	⋆⋆⋆	⋆⋆⋆
Objectness [24]	Window scoring		✓	✓	3	·	⋆	·
Rahtu [25]	Window scoring		✓	✓	3	·	·	⋆
RandomizedPrim's [26]	Grouping	✓		✓	1	⋆	⋆	⋆⋆
Rantalankila [27]	Grouping	✓		✓	10	⋆⋆	·	⋆⋆
Rigor [28]	Grouping	✓		✓	10	⋆	⋆⋆	⋆⋆
SelectiveSearch [29]	Grouping	✓	✓	✓	10	⋆⋆	⋆⋆⋆	⋆⋆⋆
Gaussian				✓	0	·	·	⋆
SlidingWindow				✓	0	⋆⋆⋆	·	·
Superpixels		✓			1	⋆	·	·
Uniform				✓	0	·	·	·

Hosang et al. , "What makes for effective detection proposals?", PAMI 2015

Summary of Region Methods

- Slide window of varying sizes across image
- Use fully convolutional method
- Use region proposals

- With those tools in our back pocket, lets tackle R-CNN, fast R-CNN, and faster R-CNN!

Regional CNN (R-CNN)

1. Pre-Training:
   Train classification model for ImageNet

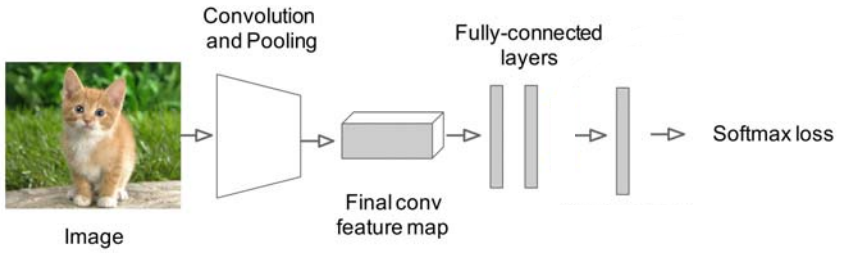


Image Credit: Dr. Fei-Fei Li, Andrej Karpathy

R-CNN

ImageNet: 1000 classes, one class per image.
PASCAL: 20 classes, each with [x,y,w,h] bounding box location in image.

2. Fine Tuning for PASCAL VOC dataset
   - Replace 1000 ImageNet classes with 21 classes (20 Object class and 1 Background)
   - Train using positive and negative regions

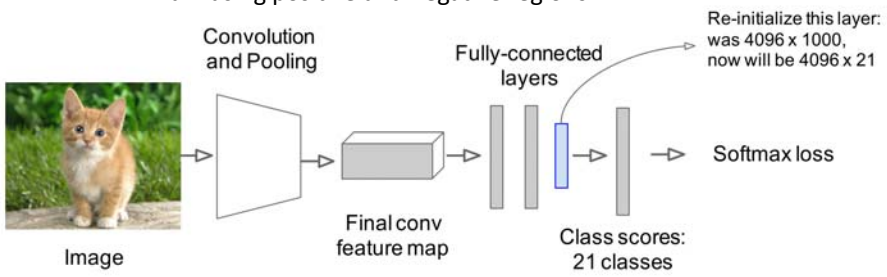
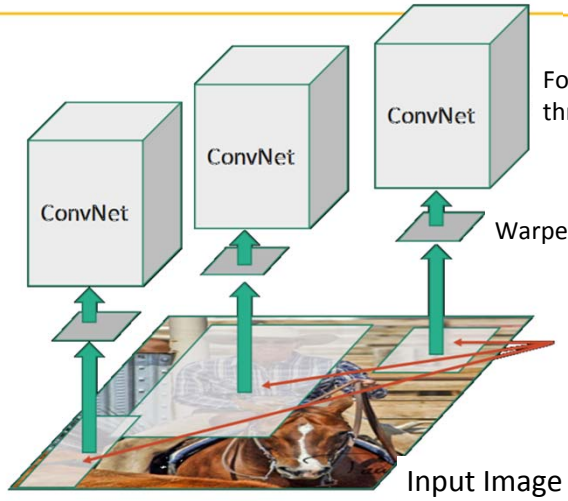


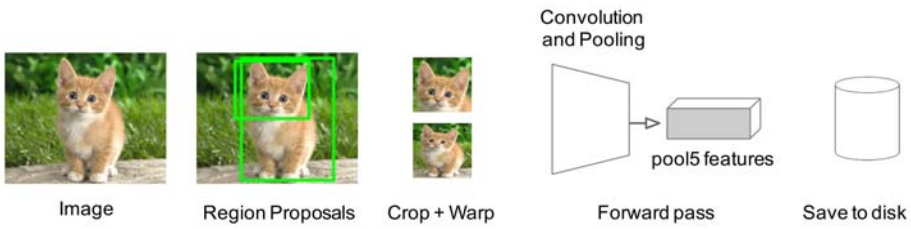
Image Credit: Dr. Fei-Fei Li, Andrej Karpathy



http://www.robots.ox.ac.uk/~tvg/publications/talks/fast-rcnn-slides.pdf

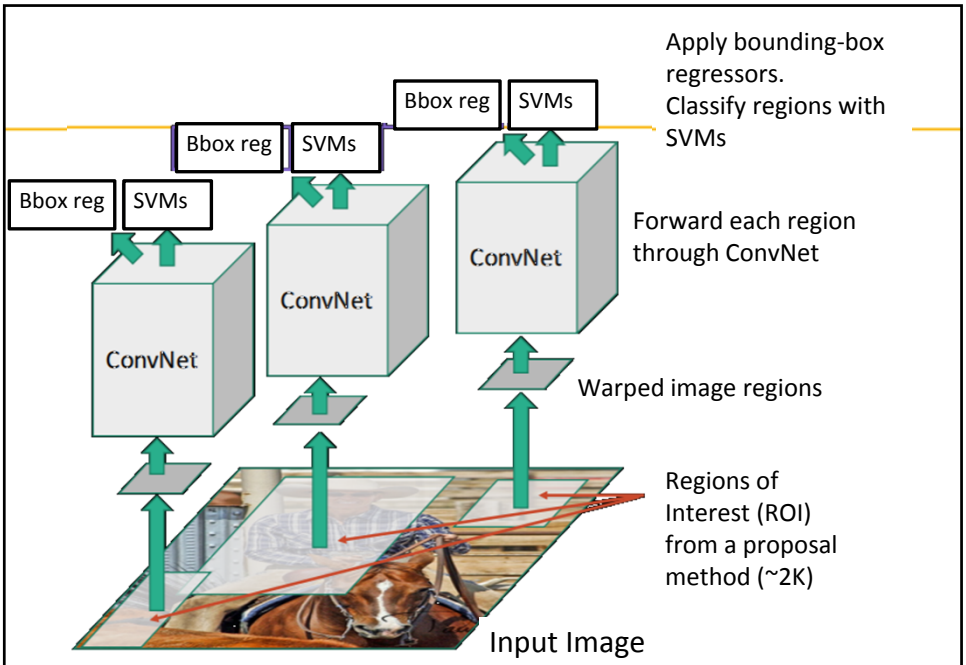
R-CNN

3. Extract Features from Proposals
   - Extract up to 2000 region proposals using Selective Search
   - Run through CNN and save Pool5 features for each region



~2000 proposals

Image Credit: Dr. Fei-Fei Li, Andrej Karpathy



http://www.robots.ox.ac.uk/~tvg/publications/talks/fast-rcnn-slides.pdf

R-CNN

4. Classification using SVM
   - Train Class Specific SVM for every object class in the dataset

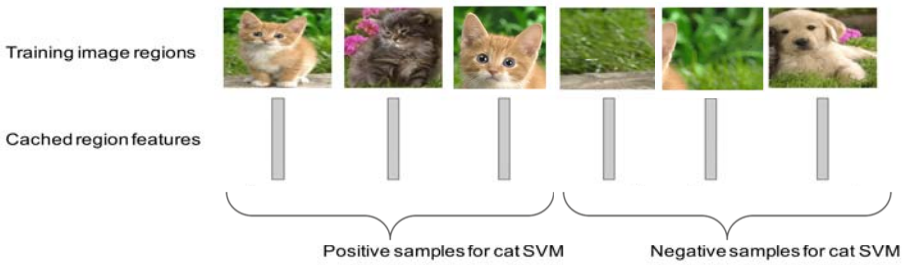


Image Credit: Dr. Fei-Fei Li, Andrej Karpathy

R-CNN

5. Bounding Box regression
   - Fine tune the proposals using the ground truth boxes

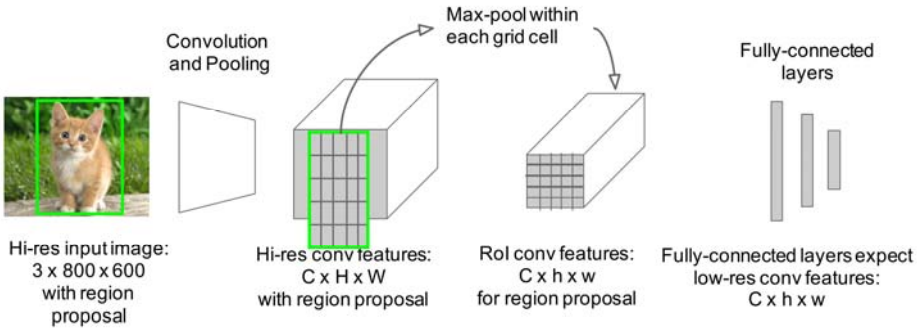
Image Credit: Dr. Fei-Fei Li, Andrej Karpathy

Limitations of R-CNN

1. Multistage training and inference
   1. Selective search
   2. Run each region through CNN
   3. Pass pool5 into SVM
   4. Fine tune bbox via regression model
2. CNN features are not updated based on SVM and regression responses.
3. Slow at test time as it needs to run forward pass on every region proposal

Fast R-CNN: Regions of Interest Pooling

For each region proposal, no matter what size or aspect ratio, max pool $H \times W$ to $h \times w$ before fully connected layers.



Actually two sibling output layers- one gives object confidence, other gives [x,y,w,h].

Image Credit: Dr. Fei-Fei Li, Andrej Karpathy

Faster R-CNN

Have the CNNs also generate the proposals!

- Selective search too slow
- Using EdgeBox instead would have done ~10× speedup alone!

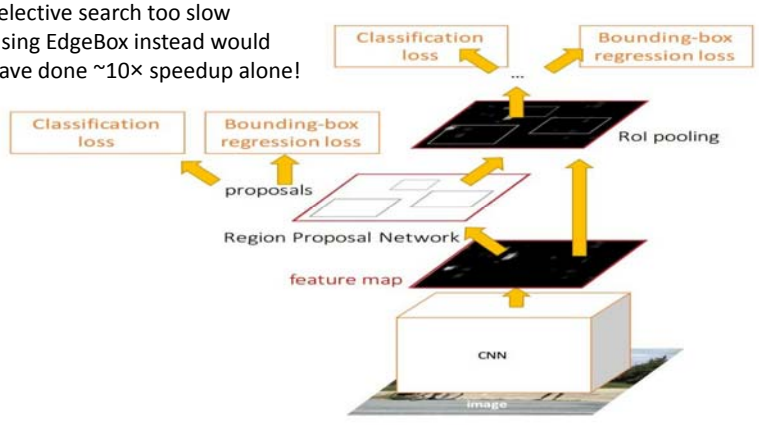
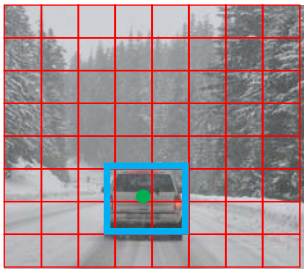


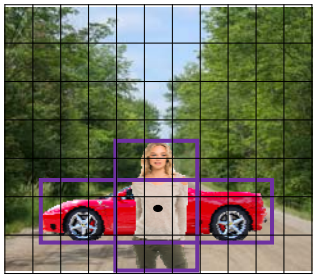
Image Credit: Ren, Shaoqing, et al.[3]

What if Lots of Objects?

- Using fully convolutional methods, the max number of objects is determined by the resolution of the last fully convolutional layer.
- For an 8×8 grid, we would detect up to 64 objects.
- Further, what if two object centroids land in same cell?

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \end{bmatrix}$$

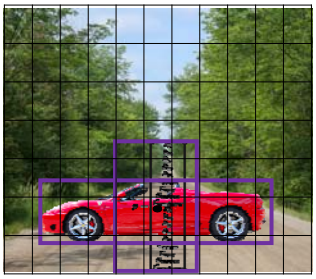




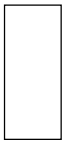
Anchor Boxes

If two anchor boxes:

- Allow multiple detections per cell.
- Define multiple anchor boxes per cell.
- Assign each GT object to closest Anchor shape (use max IoU).



Anchor 1:



Anchor 2:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \\ P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \end{bmatrix}$$

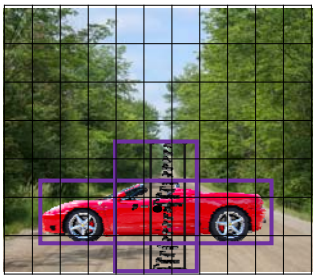
Anchor 1

Anchor 2

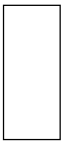
Anchor Boxes

If two anchor boxes:

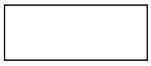
- Popular configuration is to evaluate 49×49=2401 locations. At each location, use 9 anchor boxes.
  - 3 aspect ratios 2:1,1:1,2:1
  - 3 scales, box areas of 128^2, 256^2, 512^2



Anchor 1:



Anchor 2:



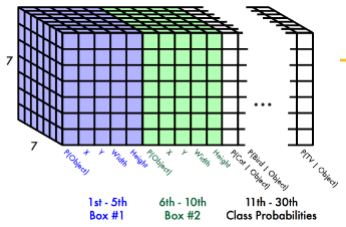
$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \\ P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \end{bmatrix}$$

Anchor 1

Anchor 2

YOLO and YOLOV2

- You Only Look Once (YOLO) is a popular bounding box utility as it is fast and almost as good quality as Faster R-CNN.
- YOLO uses 7×7 grid, each with 2 bounding boxes per grid cell, 20 classes.
- YOLO-v2 uses 19×19 grid, each with 5 bounding boxes, 80 classes.



Default params:

S=7 (grid size)

B=2 (# boxes/grid)

C=20 (#classes/box)

{P(object), x,y,w,h}

7×7 grid ×
(2 boxes per grid × 5 params/box + 20 class scores per box)
=7x7x30 tensor

Each grid cell can have *two* bboxes and *one* set of class probs.

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu

Abstract

Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Our key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet [20],

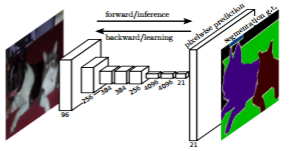
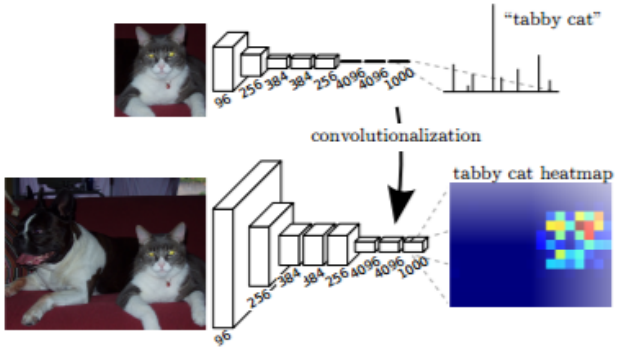


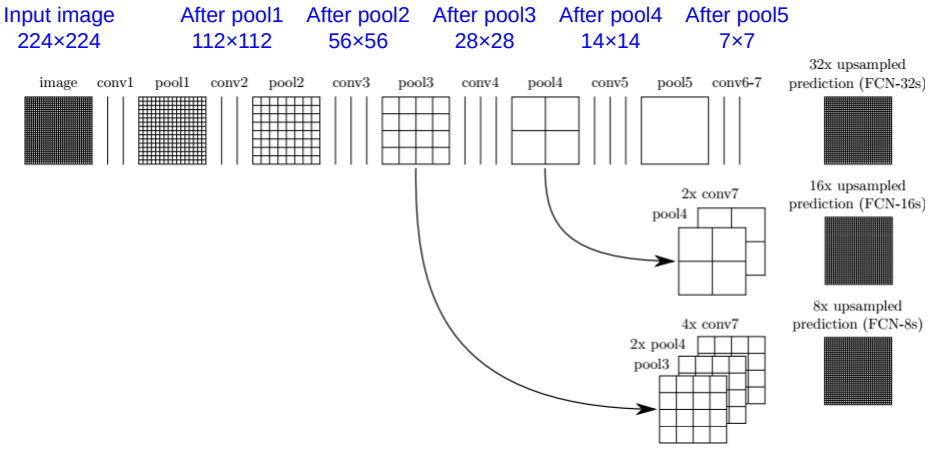
Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Why Fully Convolutional for Segmentation?

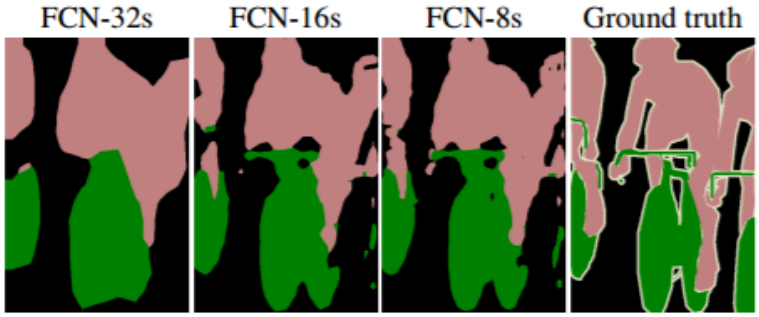


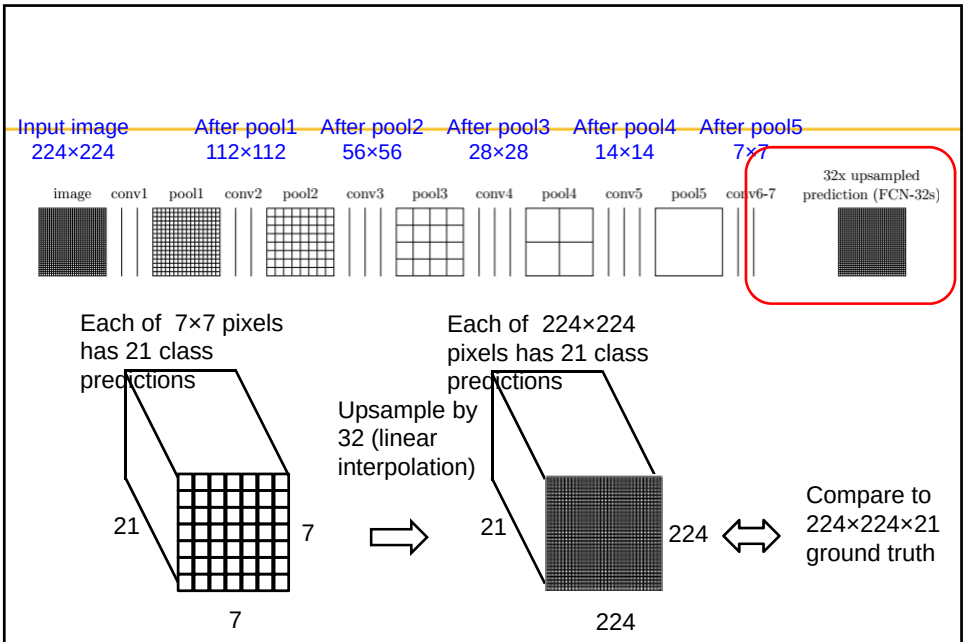
Long et al., "Fully Convolutional Networks for Semantic Segmentation", 2016.

- Up until this point, a 224×224 image, using FCN would give one output cla
- To get more (pixel-level classes), we need to increase resolution of input.
- Here is an idea which gives pixel-level classes, without increasing resoluti

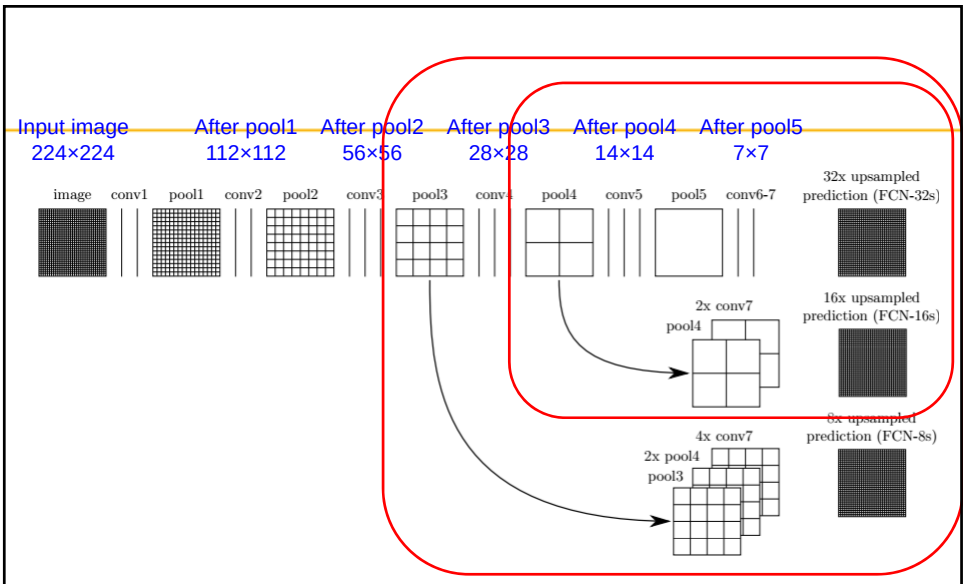


Long et al., "Fully Convolutional Networks for Semantic Segmentation", 2016.

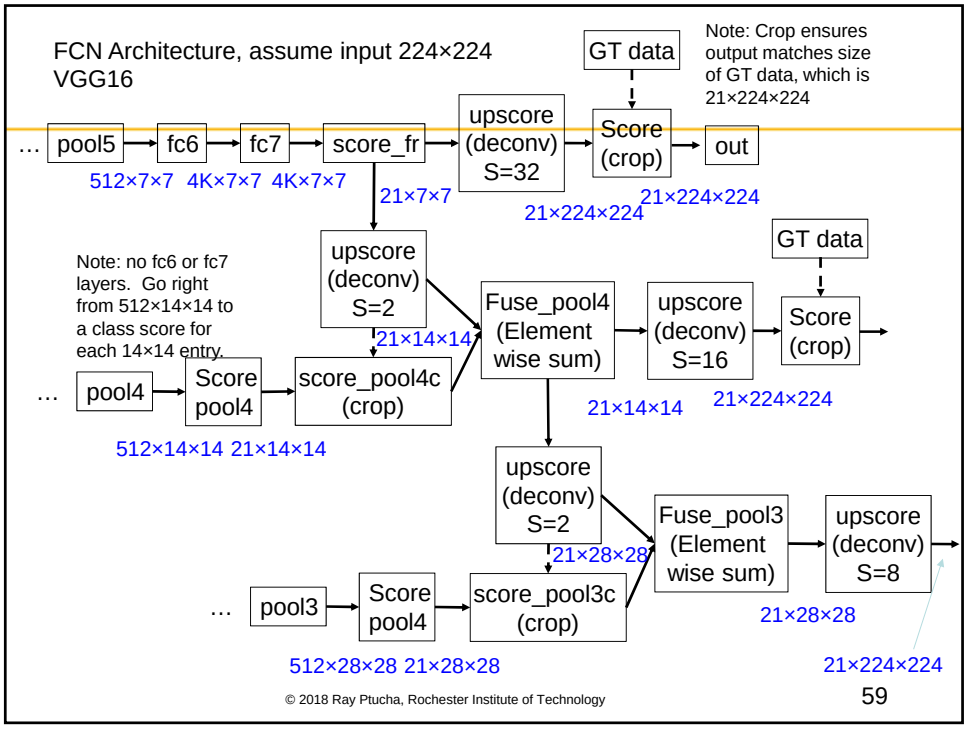


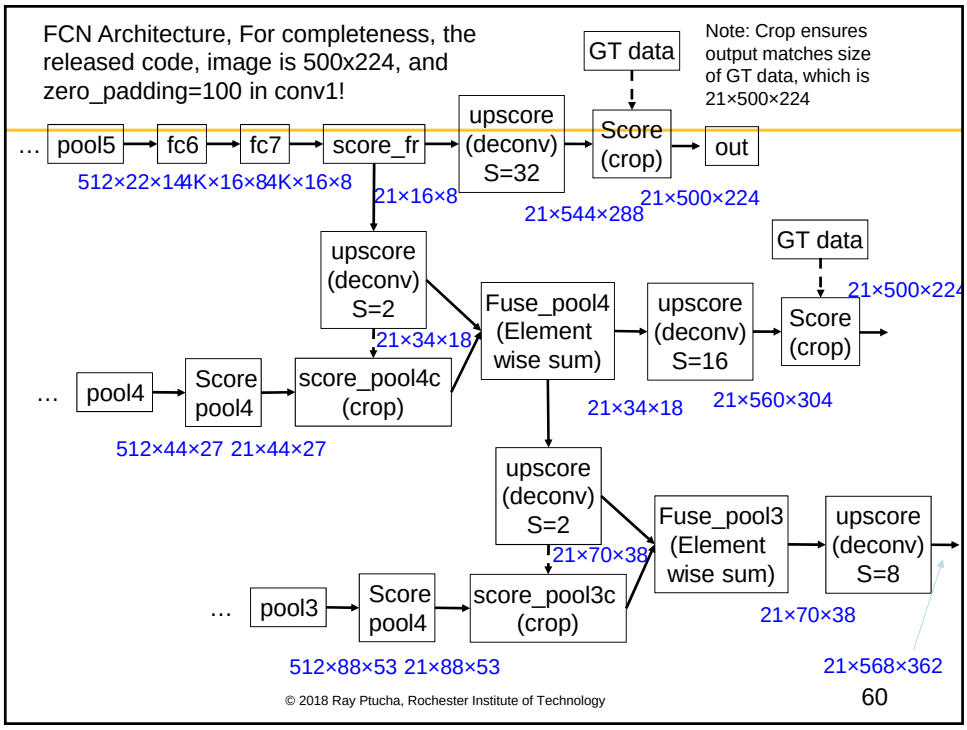


Long et al., "Fully Convolutional Networks for Semantic Segmentation", 2016.



Long et al., "Fully Convolutional Networks for Semantic Segmentation", 2016.


FCN Architecture, assume input 224×224
VGG16
GT data
Note: Crop ensures output matches size of GT data, which is 21×224×224
…
pool5
fc6
fc7
score_fr
upscore (deconv) S=32
Score (crop)
out
512×7×7
4K×7×7
4K×7×7
21×7×7
21×224×224
21×224×224
Note: no fc6 or fc7 layers. Go right from 512×14×14 to a class score for each 14×14 entry.
upscore (deconv) S=2
21×14×14
Fuse_pool4 (Element wise sum)
upscore (deconv) S=16
GT data
Score (crop)
…
pool4
Score pool4
score_pool4c (crop)
21×14×14
21×224×224
512×14×14
21×14×14
upscore (deconv) S=2
21×28×28
Fuse_pool3 (Element wise sum)
upscore (deconv) S=8
…
pool3
Score pool4
score_pool3c (crop)
21×28×28
512×28×28
21×28×28
21×224×224
© 2018 Ray Ptucha, Rochester Institute of Technology
59


FCN Architecture, For completeness, the released code, image is 500x224, and zero_padding=100 in conv1!
GT data
Note: Crop ensures output matches size of GT data, which is 21×500×224
…
pool5
fc6
fc7
score_fr
upscore (deconv) S=32
Score (crop)
out
512×22×14
4K×16×8
4K×16×8
21×16×8
21×544×288
21×500×224
upscore (deconv) S=2
21×34×18
Fuse_pool4 (Element wise sum)
upscore (deconv) S=16
GT data
21×500×224
Score (crop)
…
pool4
Score pool4
score_pool4c (crop)
21×34×18
21×560×304
512×44×27
21×44×27
upscore (deconv) S=2
21×70×38
Fuse_pool3 (Element wise sum)
upscore (deconv) S=8
…
pool3
Score pool4
score_pool3c (crop)
21×70×38
512×88×53
21×88×53
21×568×362
© 2018 Ray Ptucha, Rochester Institute of Technology
60

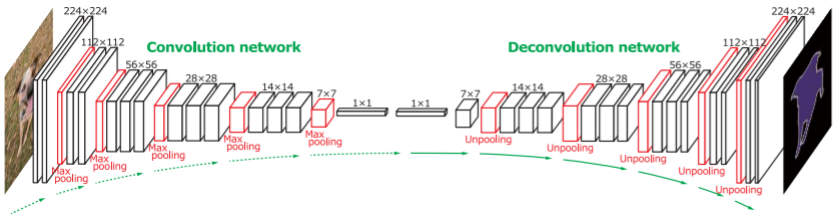
Learning Deconvolution Network for Semantic Segmentation

Hyeonwoo Noh Seunghoon Hong Bohyung Han
Department of Computer Science and Engineering, POSTECH, Korea
{hyeonwoonoh,,maga33,bhhan}@postech.ac.kr

Abstract

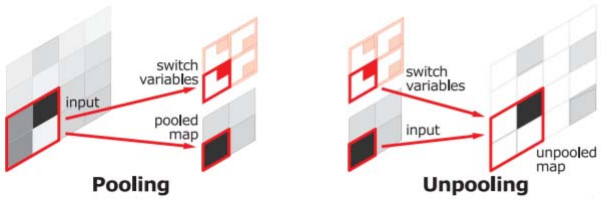
We propose a novel semantic segmentation algorithm by learning a deep deconvolution network. We learn the network on top of the convolutional layers adopted from VGG 16-layer net. The deconvolution network is composed of deconvolution and unpooling layers, which identify pixel-wise class labels and predict segmentation masks. We apply the trained network to each proposal in an input image, and construct the final semantic segmentation map by combining the results from all proposals in a simple manner. The proposed algorithm mitigates the limitations of the existing methods based on fully convolutional networks by integrating deep deconvolution network and proposal-wise prediction; our segmentation method typically identifies detailed structures and handles objects in multiple scales nat-

(a) Inconsistent labels due to large object size



- Left side is standard CNN…right side is a mirror of left
- Unpooling?
- Deconvolution?

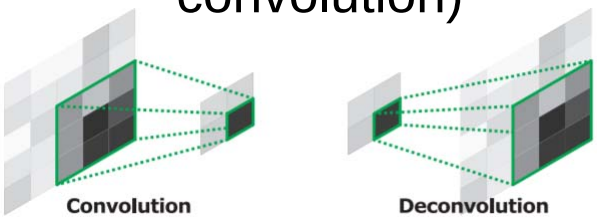
Pooling and Unpooling



Noh et al., "Learning Deconvolution Network for Semantic Segmentation", 2015.

- During max pooling, keep track of which pixel contributed to pooled pixel.
- Unpooling is reverse of pooling. Upsample, setting all values to zero, except for the pixel which contributed to the pooled pixel in the max pooling step.

Deconvolution (transposed convolution)



Noh et al., "Learning Deconvolution Network for Semantic Segmentation", 2015.

- The output of unpooling is a larger, but sparse map.
- Deconvolution densifies the activation map.
- Convolution filters take a receptive field from many activation maps and turn it into a single map.
- Deconvolution filters take a receptive field from one activation map and turn into many maps.

SegNet

https://arxiv.org/pdf/1511.00561.pdf

Vijay Badrinarayanan, Alex Kendall and Roberto Cipolla "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." PAMI, 2017

SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, *Senior Member, IEEE,*

Abstract—We present a novel and practical deep fully convolutional neural network architecture for semantic pixel-wise segmentation termed SegNet. This core trainable segmentation engine consists of an encoder network, a corresponding decoder network followed by a pixel-wise classification layer. The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network [1]. The role of the decoder network is to map the low resolution encoder feature maps to full input resolution feature maps for pixel-wise classification. The novelty of SegNet lies is in the manner in which the decoder upsamples its lower resolution input feature map(s). Specifically, the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. This eliminates the need for learning to upsample. The upsampled maps are sparse and are then convolved with trainable filters to produce dense feature maps. We compare our proposed architecture with the widely adopted FCN [2] and also with the well known DeepLab-LargeFOV [3], DeconvNet [4] architectures. This comparison reveals the memory versus accuracy trade-off involved in achieving good segmentation performance.
SegNet was primarily motivated by scene understanding applications. Hence, it is designed to be efficient both in terms of memory and computational time during inference. It is also significantly smaller in the number of trainable parameters than other competing architectures and can be trained end-to-end using stochastic gradient descent. We also performed a controlled benchmark of SegNet and other architectures on both road scenes and SUN RGB-D indoor scene segmentation tasks. These quantitative assessments show that SegNet provides good performance with competitive inference time and most efficient inference memory-wise as compared to other architectures. We also provide a Caffe implementation of SegNet and a web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

Index Terms—Deep Convolutional Neural Networks, Semantic Pixel-Wise Segmentation, Indoor Scenes, Road Scenes, Encoder, Decoder, Pooling, Upsampling.

SegNet

- 26 layer Fully Conv Nnet- trained end-to-end with SGD
- Fast: 65ms/image
- “light weight: 120MB for parameters.

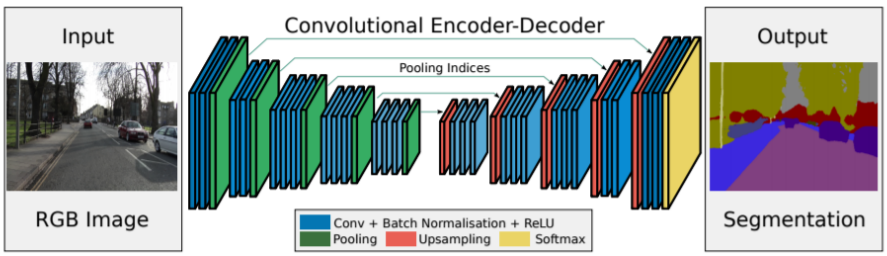


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

SegNet

- Like deconvnet, maxpooling indices used in decoder during upsampling.

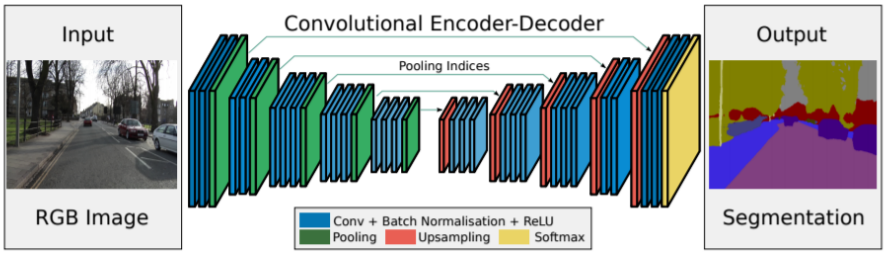


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

SegNet

- First 13 layers from VGG16 (throw away all fully connected layers)
- Second 13 are inverses of these
- Batch normalization used on all layers
- During 2x2 pooling, switch variables used to remember where pooled pixels came from.

- Note: DeconNet used switch variables, but went all the way down to 1x1 resolution, making harder to train.
- U-Net saved the entire lower res maps during decoding which consumes more memory. (U-Net also skipped last pooling and conv layer)

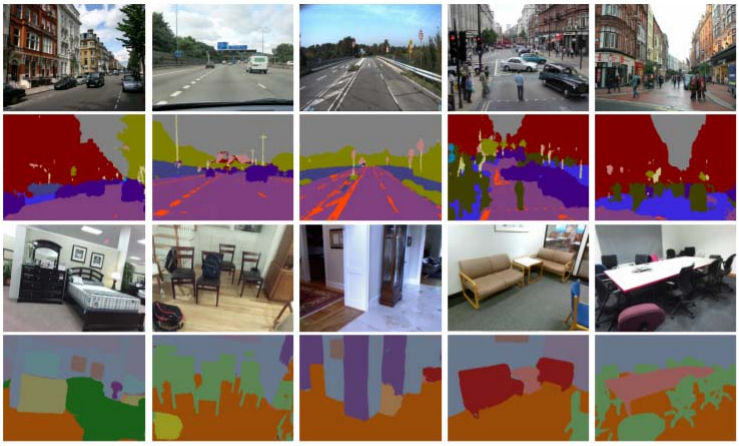


Fig. 1. SegNet predictions on road scenes and indoor scenes. To try our system yourself, please see our online web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick
Facebook AI Research (FAIR)

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel *with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box*

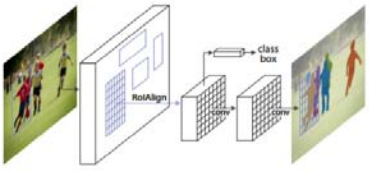


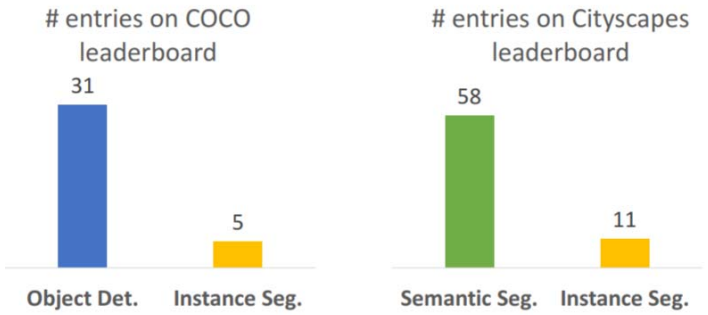
Figure 1. The **Mask R-CNN** framework for instance segmentation.

a fixed set of categories without differentiating object instances.[1] Given this, one might expect a complex method is required to achieve good results. However, we show that

https://arxiv.org/pdf/1703.06870.pdf

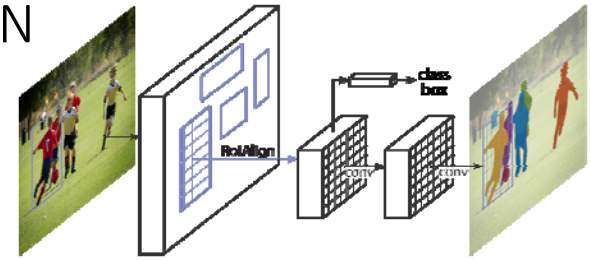
Instance Segmentation a Tough Problem

Object Detection | Semantic Segmentation | **Instance Segmentation**



http://kaiminghe.com/iccv17tutorial/maskrcnn_iccv2017_tutorial_kaiminghe.pdf

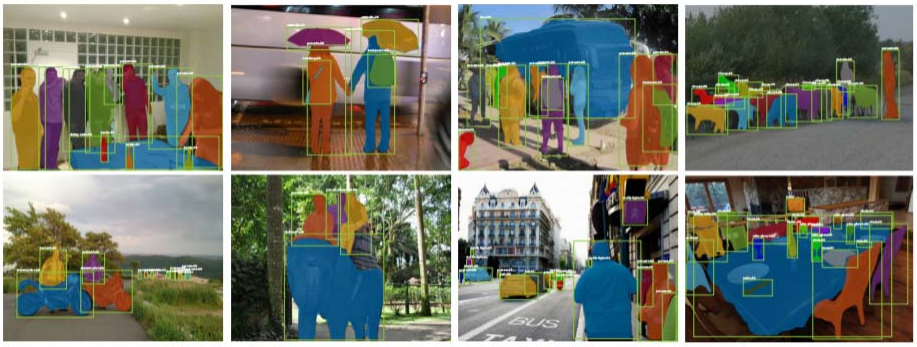
Mask R-CNN



- Extension of Faster R-CNN
- Faster R-CNN used region proposal network (3×3 sliding window over conv5) predicts *n* anchor boxes at each location. Each anchor box has [P(object),x,y,w,h]
- Each anchor box then used for final network to predict [P(object),x,y,w,h,C_1,C_2,…,C_C]
- Extend to also predict 28×28 pixel mask of object

Mask R-CNN

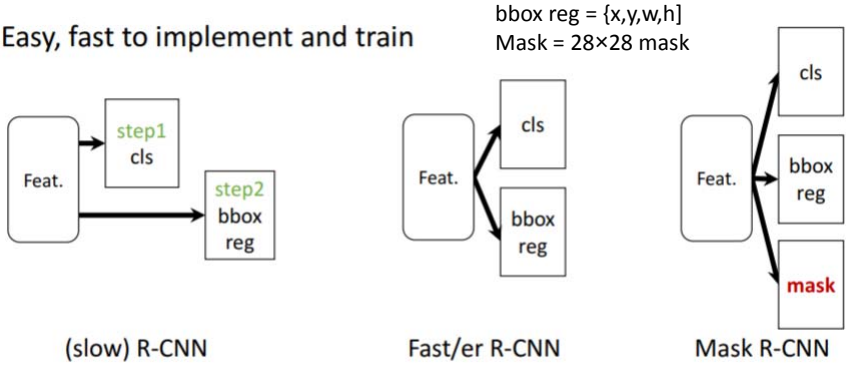
- Extension of Faster R-CNN
- Simultaneously predict bounding box along with object mask.
- Shown to work well on instance segmentation, bounding-box object detection, and person keypoint (body joint) detection.



Evolution of Mask R-CNN

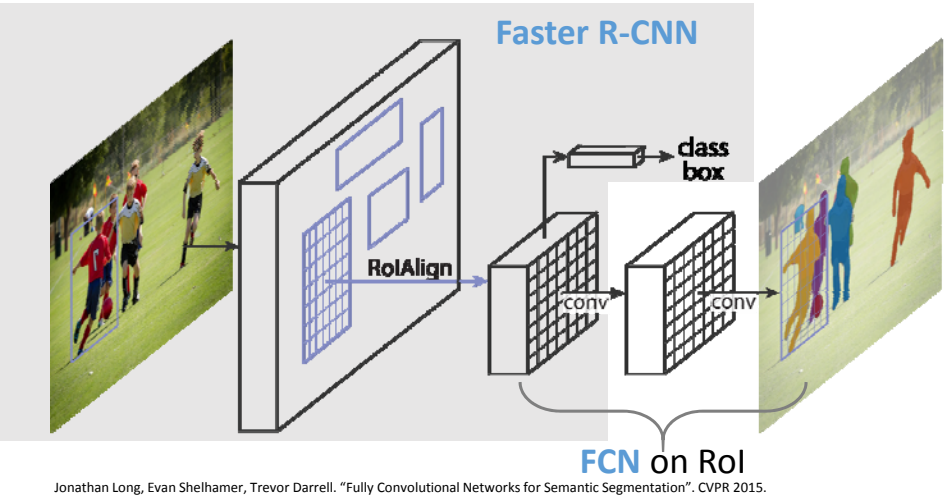
cls= class score $C_1, C_2, \ldots C_C$
bbox reg = {x,y,w,h]
Mask = 28×28 mask

- Easy, fast to implement and train



http://kaiminghe.com/iccv17tutorial/maskrcnn_iccv2017_tutorial_kaiminghe.pdf

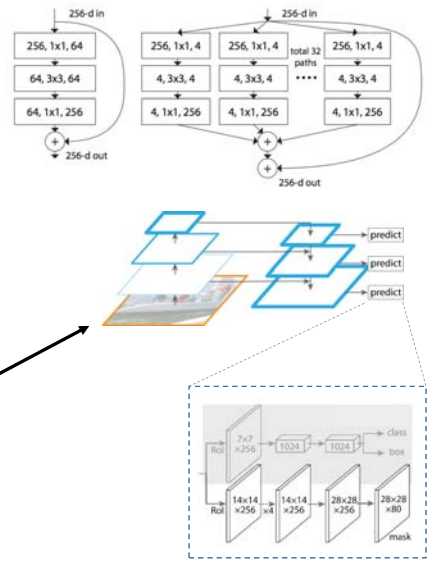
Mask R-CNN



Jonathan Long, Evan Shelhamer, Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". CVPR 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.
Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. "Mask R-CNN". ICCV 2017.

Mask R-CNN

- Meta-algorithm: algorithm learns its own hyperparameters…
- Built on:
- ResNext [Xie et al., CVPR'17]
- Feature Pyramid Net [Lin eta al., CVPR'17]



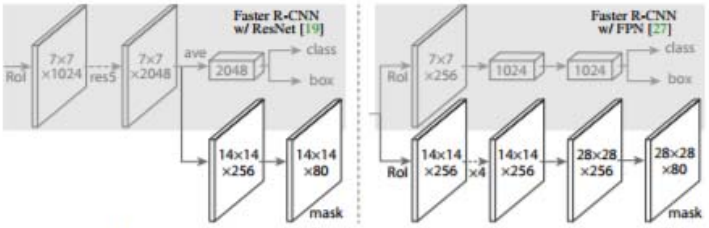
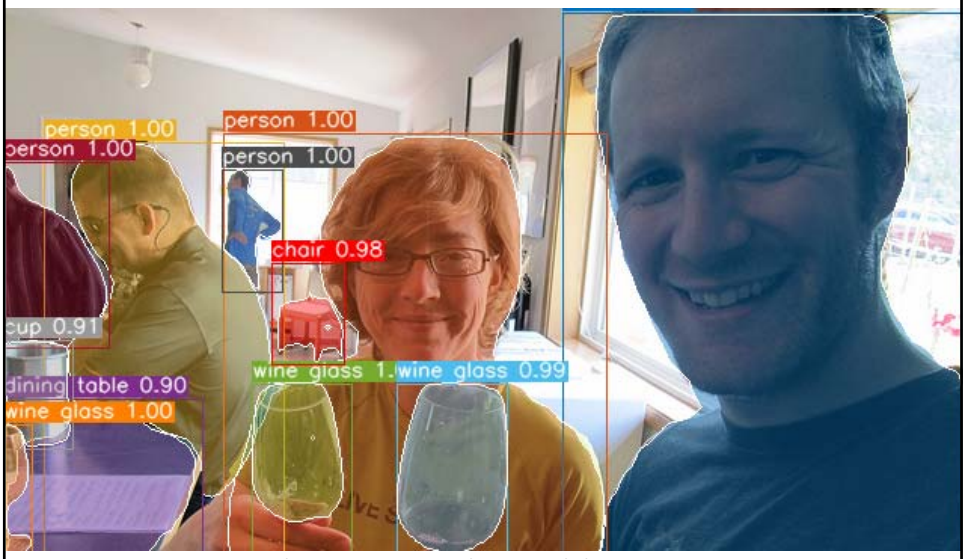


Figure 3. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, 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 [30] 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.

77

FAIR Mask R-CNN, COCO + Places Workshop, ICCV 2017

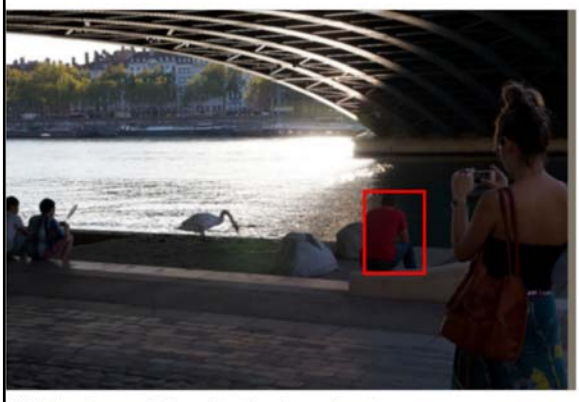
78



FAIR Mask R-CNN, COCO + Places Workshop, ICCV 2017

How get such nice masks with only 28×28 pix per mask?

Mask Prediction



Validation image with box detection shown in red

28x28 soft prediction from Mask R-CNN
(enlarged)

Soft prediction resampled to image coordinate
(bilinear and bicubic interpolation work equally well)

Final prediction (threshold at 0.5)

http://deeplearning.csail.mit.edu/instance_ross.pdf

Mask Prediction

28x28 soft prediction

Resized soft prediction

Final mask

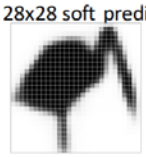
Validation image with box detection shown in red

http://deeplearning.csail.mit.edu/instance_ross.pdf

81

Mask Prediction

28x28 soft prediction



Resized Soft prediction

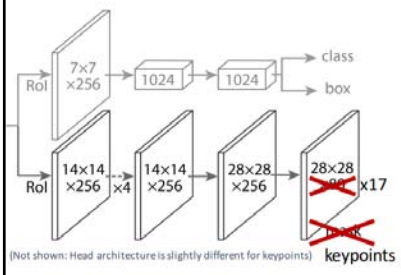
Final mask

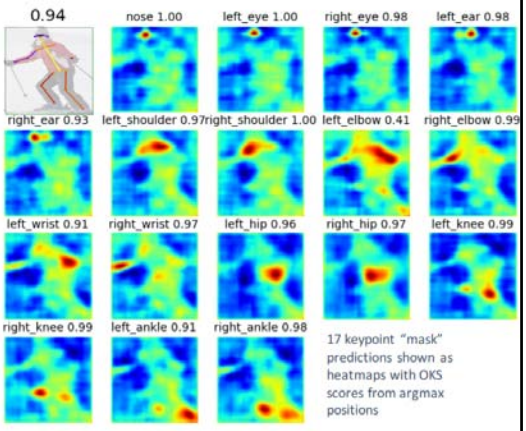
Validation image with box detection shown in red

http://deeplearning.csail.mit.edu/instance_ross.pdf

82

Human Pose

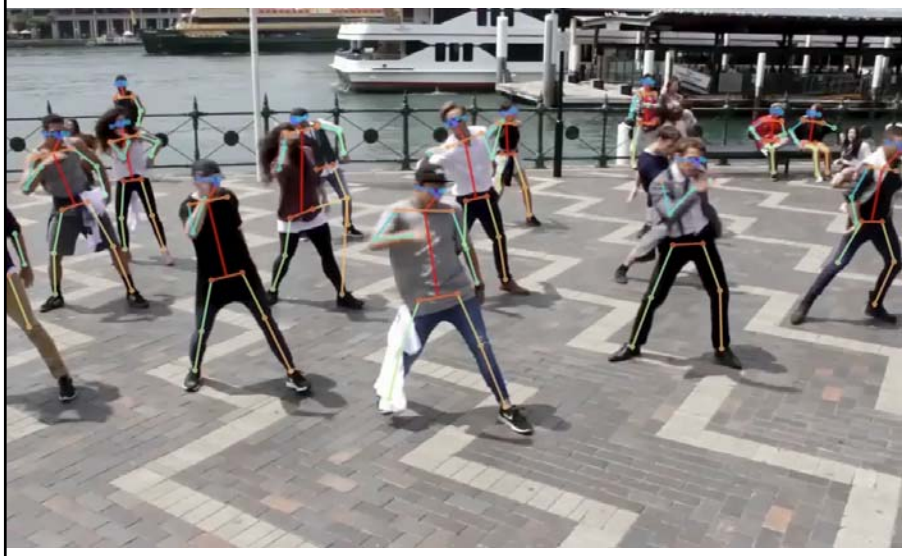




- Add keypoint head (28x28x17)
- Predict one "mask" for each keypoint (keypoint==bodyjoint)
- Softmax over spatial locations (encodes one keypoint per mask "prior")

http://deeplearning.csail.mit.edu/instance_ross.pdf

FAIR Mask R-CNN, COCO + Places Workshop, ICCV 2017



FAIR Mask R-CNN, COCO + Places Workshop, ICCV 2017

Figure 6. Keypoint detection results on COCO `test` using Mask R-CNN (ResNet-50-FPN), with person segmentation masks predicted from the same model. This model has a keypoint AP of 63.1 and runs at 5 fps.

Figure 7. Mask R-CNN results on Cityscapes `test` (32.0 AP). The bottom-right image shows a failure prediction.

Thank you!!

Ray Ptucha
rwpeec@rit.edu

https://www.rit.edu/mil