Detection and Segmentation

CS60010: Deep Learning

Abir Das

IIT Kharagpur

March 04 and 05, 2020

Agenda

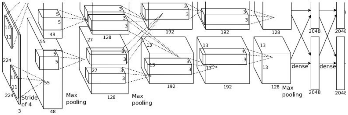
To get introduced to two important tasks of computer vision - detection and segmentation along with deep neural network's application in these areas in recent years.

Detection as Regression

- In detection you don't know the number of objects present
- So, it is problematic to address detection as regression
- How many output neurons to put?

Detection as Classification

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

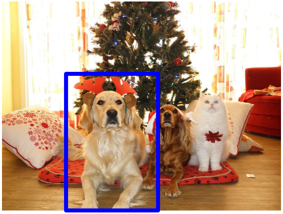


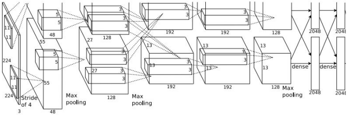
Dog? NO
Cat? NO
Background? YES

CS231n course, Stanford University

Detection as Classification

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



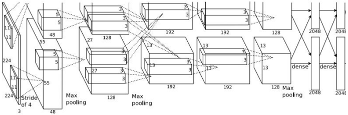


Dog? YES
Cat? NO
Background? NO

CS231n course, Stanford University

Detection as Classification

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

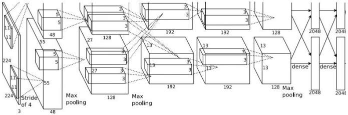


Dog? YES
Cat? NO
Background? NO

CS231n course, Stanford University

Detection as Classification

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

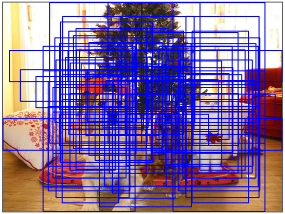


Dog? NO
Cat? YES
Background? NO

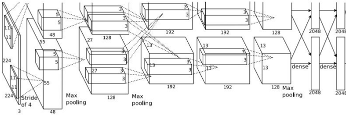
CS231n course, Stanford University

Detection as Classification

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



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



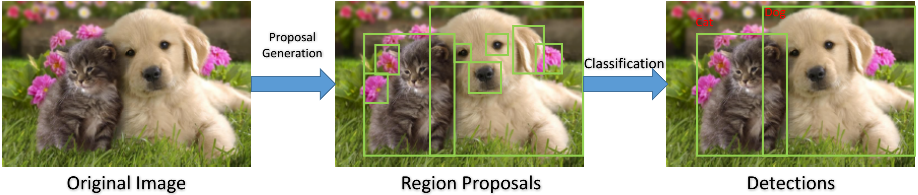
Dog? NO
Cat? YES
Background? NO

CS231n course, Stanford University

Detection as Classification

- Need to apply CNN to huge number of locations, scales and aspect ratios
- If the classifier is fast enough, this is done. Pre Deep Learning approach.
- Deep learning classifiers, first get a tiny subset of possible positions. Only these are passed through the deep classifiers.
- The possible positions are called ‘candidate proposals’ or ‘region proposals’.

Detection with Region Proposals



- Generate and evaluate a few (much less than exhaustive search) region proposals
- Proposal mechanism can take advantage of low-level cues (*e.g.*, edges or connected components)
- Classifier can be slower but more powerful

Selective Search

J Uijlings, K van de Sande, T Gevers, and A Smeulders, 'Selective Search for Object Recognition', IJCV 2013

Selective Search

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$
  $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
  Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
  Calculate similarity set S_t between r_t and its neighbours
  $S = S \cup S_t$
  $R = R \cup r_t$

Extract object location boxes L from all regions in R

J Uijlings, K van de Sande, T Gevers, and A Smeulders, 'Selective Search for Object Recognition', IJCV 2013

EdgeBoxes

- Edgeboxes depend on a fast scoring/evaluating method for bounding boxes.
- First edges are extracted for the whole image and they are grouped according to their similarity
- The main idea of scoring boxes builds on the fact that edges tend to correspond to object boundaries and bounding boxes that tightly enclose a set of edges are likely to contain an object.
- Gets 75% recall with 800 boxes (vs 1400 for Selective Search) and is 40 times faster

C Zitnick and P Dollar, 'Edge Boxes: Locating Object Proposals from Edges', ECCV 2014

Many Region Proposal Methods

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	·	·	·

J Hosang, R Benenson, P Dollar and B Schiele, 'What makes for effective detection proposals?', IEEE TPAMI 2016

R-CNN: Region Proposals + CNN Features

§ R Girshick, J Donahue, T Darrell and J Malik, 'R-CNN: Region-based Convolutional Neural Networks', CVPR 2014

CS231n course, Stanford University

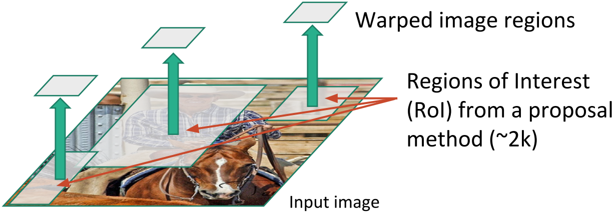
R-CNN: Region Proposals + CNN Features

R-CNN

CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features

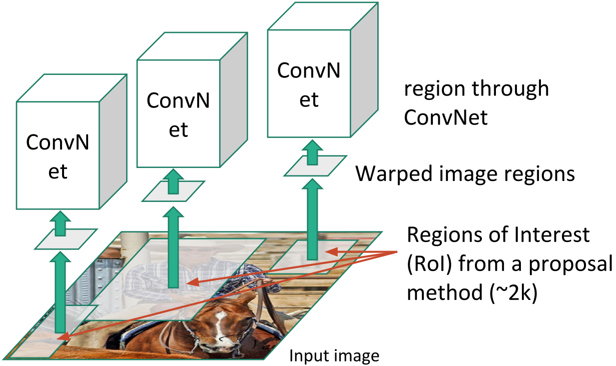
R-CNN



CS231n course, Stanford University

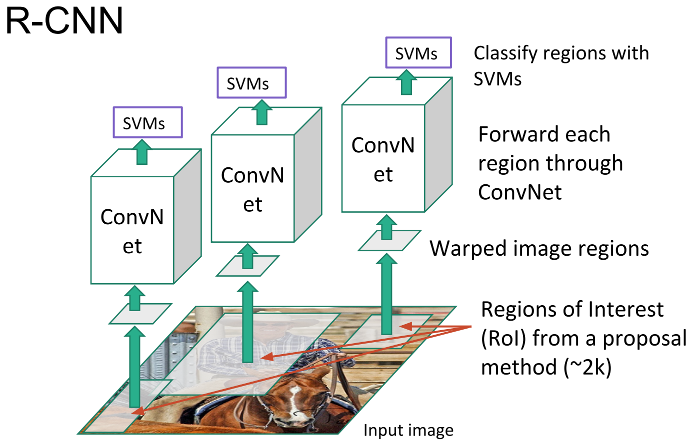
R-CNN: Region Proposals + CNN Features

R-CNN



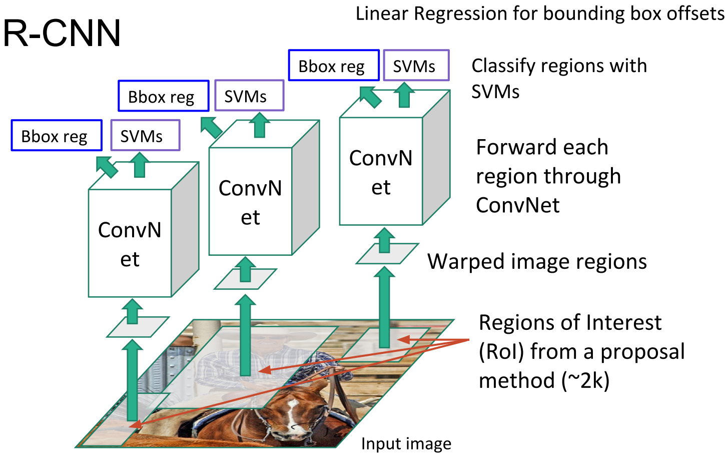
CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features



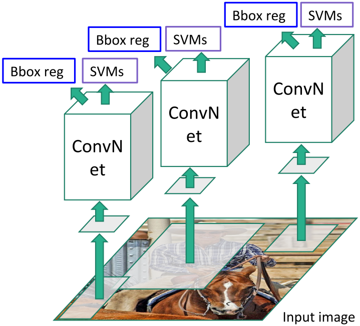
CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features



CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features



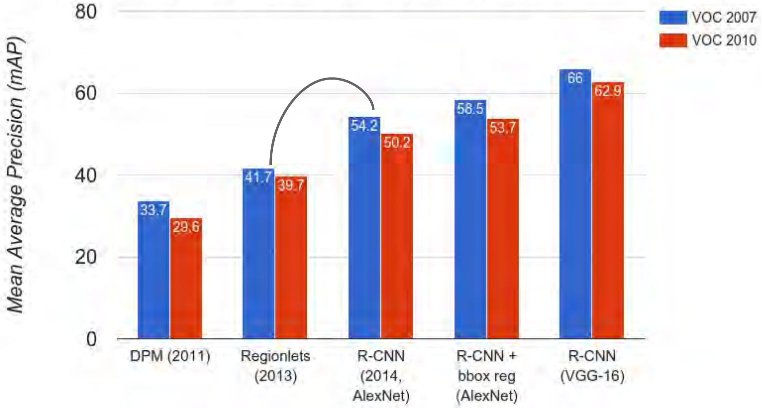
The parameters learned for this pipeline are: ConvNet, SVM Classifier and Bounding-Box regressors

CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features

R-CNN Results

Big improvement compared to pre-CNN methods

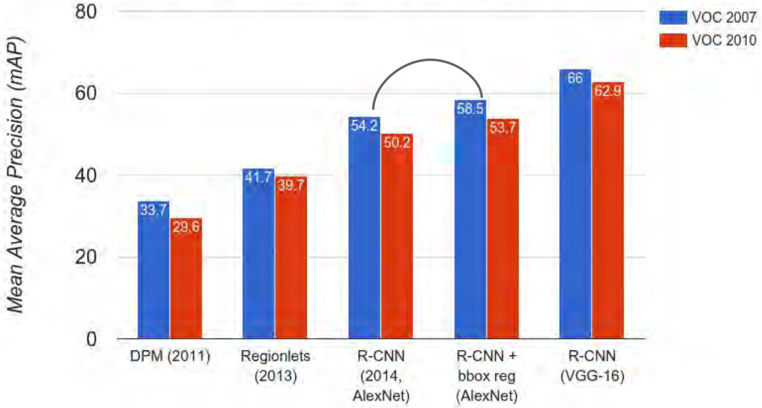


CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features

R-CNN Results

Bounding box regression helps a bit

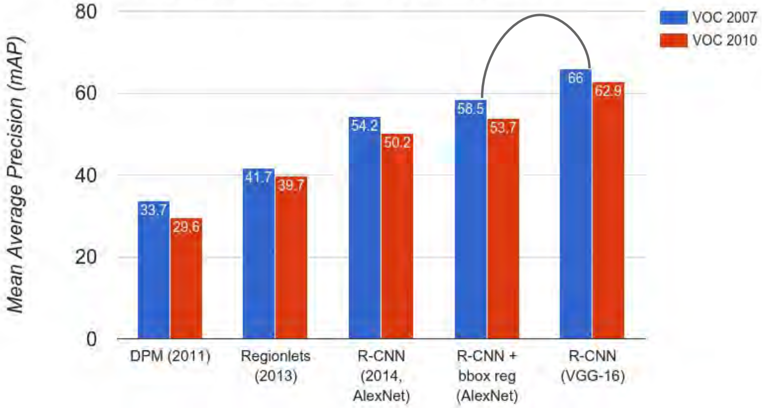


CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features

R-CNN Results

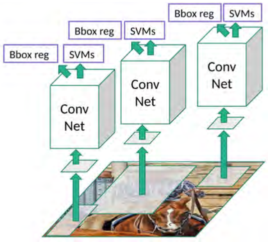
Features from a deeper network help a lot



CS231n course, Stanford University

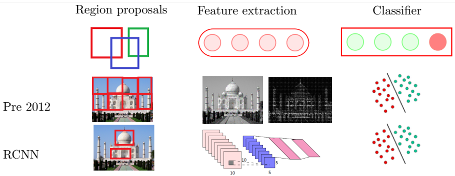
R-CNN: Region Proposals + CNN Features

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



CS231n course, Stanford University

R-CNN: Region Proposals + CNN Features

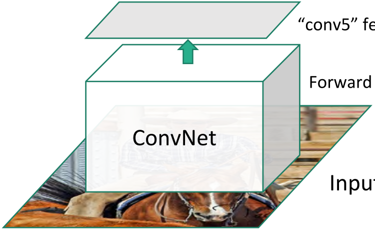


- **Region Proposals:** Selective Search
- **Feature Extraction:** CNNs
- **Classifier:** Linear

CS7015 course, IIT Madras

Fast R-CNN

Fast R-CNN

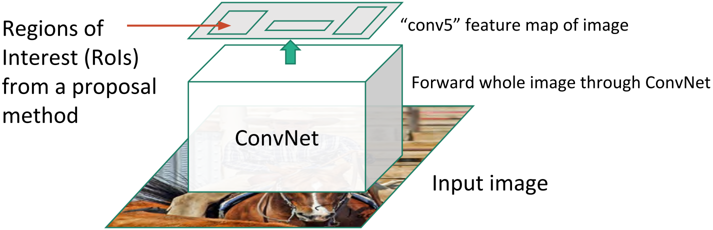


§ R Girshick, 'Fast R-CNN', ICCV 2015

CS231n course, Stanford University

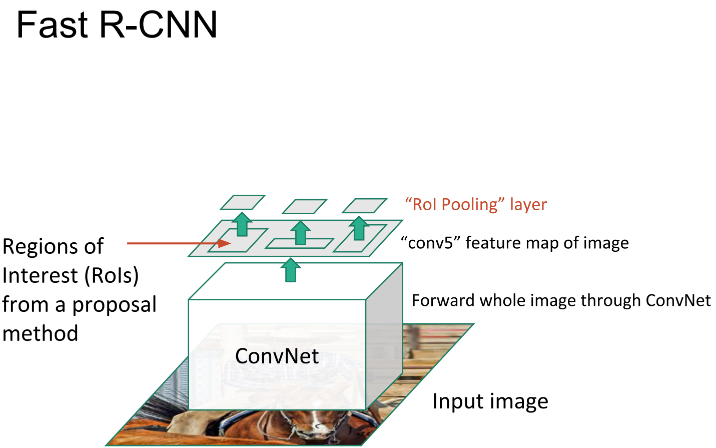
Fast R-CNN

Fast R-CNN



CS231n course, Stanford University

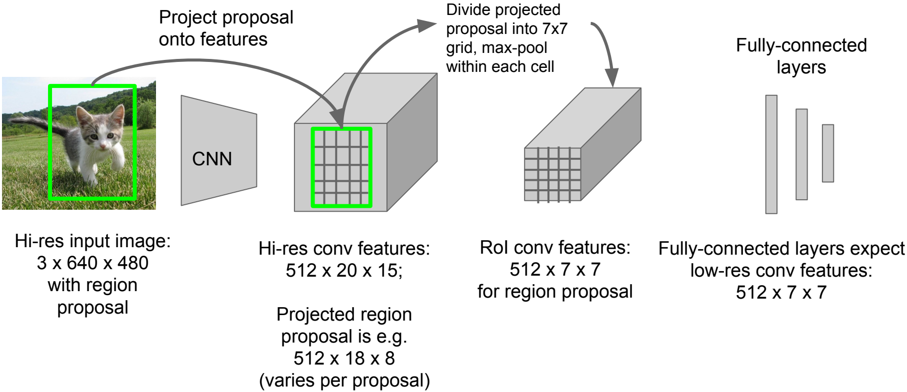
Fast R-CNN



CS231n course, Stanford University

Fast R-CNN

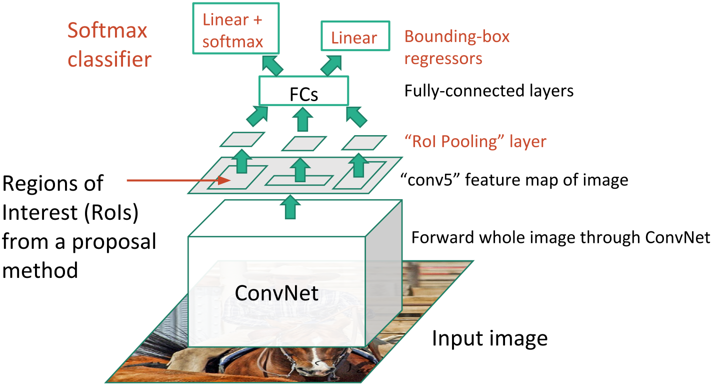
Fast R-CNN: RoI Pooling



CS231n course, Stanford University

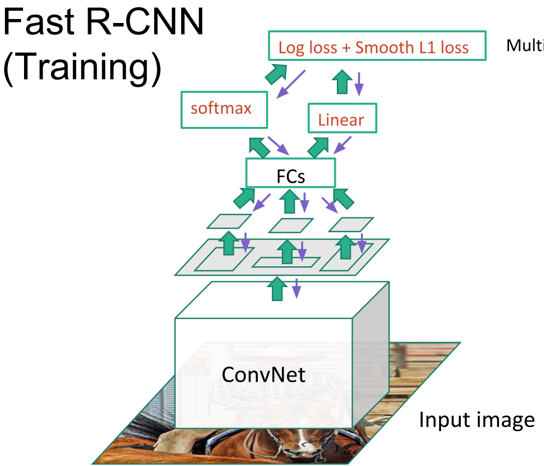
Fast R-CNN

Fast R-CNN



CS231n course, Stanford University

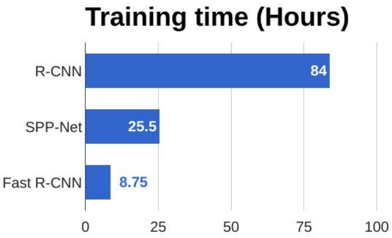
Fast R-CNN

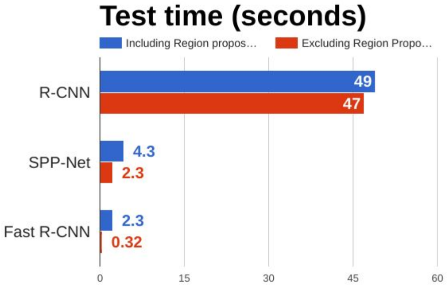


CS231n course, Stanford University

Fast R-CNN

R-CNN vs SPP vs Fast R-CNN



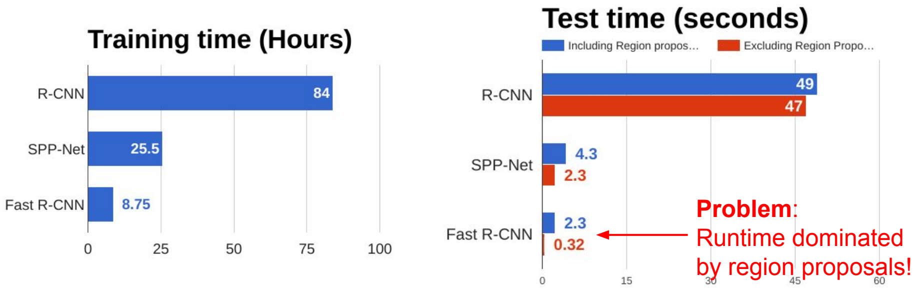


Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

CS231n course, Stanford University

Fast R-CNN

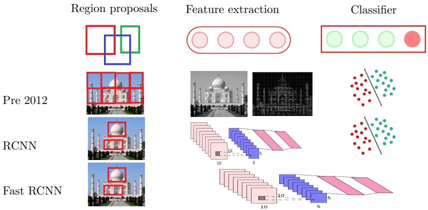
R-CNN vs SPP vs Fast R-CNN



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

CS231n course, Stanford University

Fast R-CNN



- **Region Proposals:** Selective Search
- **Feature Extraction:** CNN
- **Classifier:** CNN

Faster R-CNN

- The bulk of the time at test time of Fast RCNN is dominated by the region proposal generation.
- As Fast RCNN saved computation by sharing the feature generation for all proposals, can some sort of sharing of computation be done for generating region proposals?
- The solution is to use the same CNN for region proposal generation too.

Faster R-CNN

- The bulk of the time at test time of Fast RCNN is dominated by the region proposal generation.
- As Fast RCNN saved computation by sharing the feature generation for all proposals, can some sort of sharing of computation be done for generating region proposals?
- The solution is to use the same CNN for region proposal generation too.
- S Ren and K He and R Girshick and J Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks', NIPS 2015

Faster R-CNN

- The bulk of the time at test time of Fast RCNN is dominated by the region proposal generation.
- As Fast RCNN saved computation by sharing the feature generation for all proposals, can some sort of sharing of computation be done for generating region proposals?
- The solution is to use the same CNN for region proposal generation too.
- S Ren and K He and R Girshick and J Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks', NIPS 2015
- The region proposal generation part is termed as the Region Proposal Network (RPN)

Faster R-CNN

- The bulk of the time at test time of Fast RCNN is dominated by the region proposal generation.
- As Fast RCNN saved computation by sharing the feature generation for all proposals, can some sort of sharing of computation be done for generating region proposals?
- The solution is to use the same CNN for region proposal generation too.
- S Ren and K He and R Girshick and J Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks', NIPS 2015
- The region proposal generation part is termed as the Region Proposal Network (RPN)

Faster R-CNN

- The bulk of the time at test time of Fast RCNN is dominated by the region proposal generation.
- As Fast RCNN saved computation by sharing the feature generation for all proposals, can some sort of sharing of computation be done for generating region proposals?
- The solution is to use the same CNN for region proposal generation too.
- S Ren and K He and R Girshick and J Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks', NIPS 2015
- The region proposal generation part is termed as the Region Proposal Network (RPN)

Faster R-CNN

- The RPN works as follows:
  - A small 3x3 conv layer is applied on the last layer of the base conv-net
  - it produces activation feature map of the same size as the base conv-net last layer feature map (7x7x512 in case of VGG base)
  - At each of the feature positions (7x7=49 for VGG base), a set of bounding boxes (with different scale and aspect ratio) are evaluated for the following two questions
    - given the 512d feature at that position, what is the probability that each of the bounding boxes centered at the position contains an object? (Classification)
    - Given the same 512d feature can you predict the correct bounding box? (Regression)
  - These boxes are called '*anchor boxes*'

Faster R-CNN

§ The RPN works as follows:

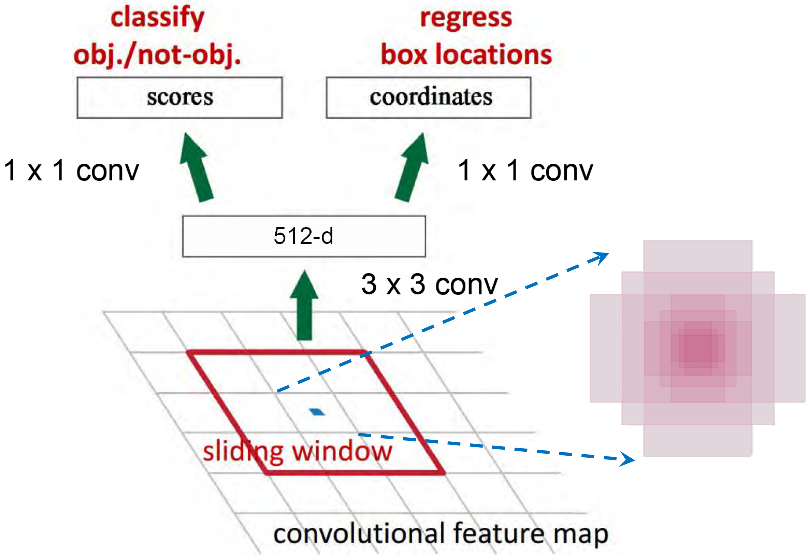
- A small 3x3 conv layer is applied on the last layer of the base conv-net
- it produces activation feature map of the same size as the base conv-net last layer feature map (7x7x512 in case of VGG base)
- At each of the feature positions (7x7=49 for VGG base), a set of bounding boxes (with different scale and aspect ratio) are evaluated for the following two questions
  - given the 512d feature at that position, what is the probability that each of the bounding boxes centered at the position contains an object? (Classification)
  - Given the same 512d feature can you predict the correct bounding box? (Regression)
- These boxes are called '*anchor boxes*'

Faster R-CNN

§ The RPN works as follows:

- A small 3x3 conv layer is applied on the last layer of the base conv-net
- it produces activation feature map of the same size as the base conv-net last layer feature map (7x7x512 in case of VGG base)
- At each of the feature positions (7x7=49 for VGG base), a set of bounding boxes (with different scale and aspect ratio) are evaluated for the following two questions
  - given the 512d feature at that position, what is the probability that each of the bounding boxes centered at the position contains an object? (Classification)
  - Given the same 512d feature can you predict the correct bounding box? (Regression)
- These boxes are called '*anchor boxes*'

Faster R-CNN


classify
obj./not-obj.
scores
regress
box locations
coordinates
1 x 1 conv
1 x 1 conv
512-d
3 x 3 conv
sliding window
convolutional feature map

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.

Consider a ground truth object and its corresponding bounding box

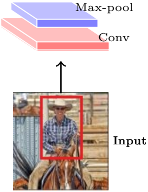
Input

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.

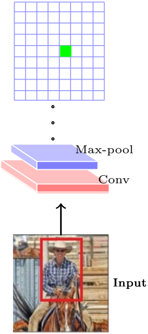
Consider a ground truth object and its corresponding bounding box

Consider the projection of this image onto the conv5 layer



Faster R-CNN

§ But how do we get the ground truth data to train the RPN.



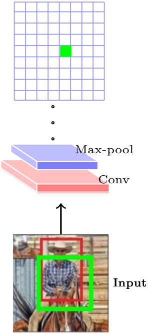
Consider a ground truth object and its corresponding bounding box

Consider the projection of this image onto the conv5 layer

Consider one such cell in the output

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.



Consider a ground truth object and its corresponding bounding box

Consider the projection of this image onto the conv5 layer

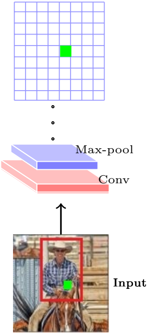
Consider one such cell in the output

This cell corresponds to a patch in the original image

CS7015 course, IIT Madras

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.



Consider a ground truth object and its corresponding bounding box

Consider the projection of this image onto the conv5 layer

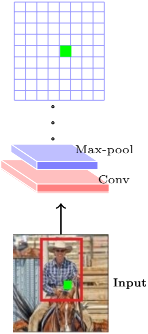
Consider one such cell in the output

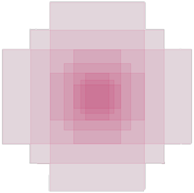
This cell corresponds to a patch in the original image

Consider the center of this patch

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.





Consider a ground truth object and its corresponding bounding box

Consider the projection of this image onto the conv5 layer

Consider one such cell in the output

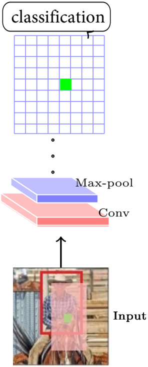
This cell corresponds to a patch in the original image

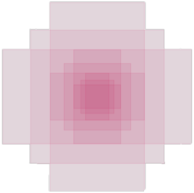
Consider the center of this patch

We consider anchor boxes of different sizes

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.



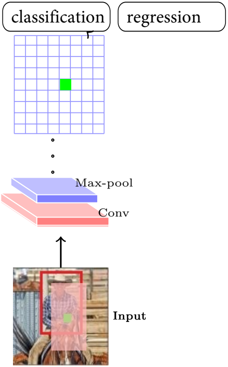


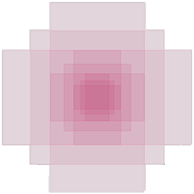
For each of these anchor boxes, we would want the classifier to predict 1 if this anchor box has a reason-able overlap (IoU $>$ 0.7) with the true grounding box

CS7015 course, IIT Madras

Faster R-CNN

§ But how do we get the ground truth data to train the RPN.





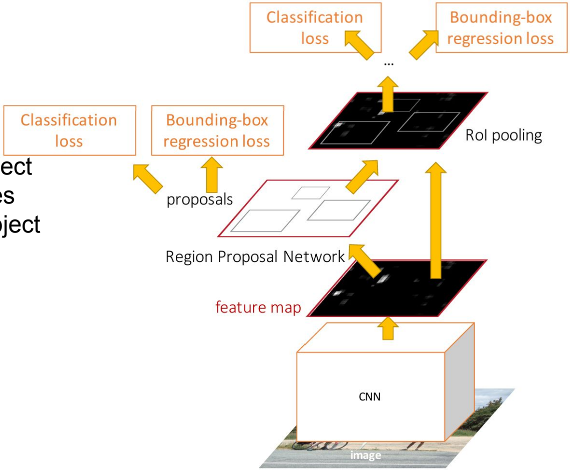
For each of these anchor boxes, we would want the classifier to predict 1 if this anchor box has a reasonable overlap (IoU > 0.7) with the true grounding box

Similarly we would want the regression model to predict the true box (red) from the anchor box (pink)

Faster R-CNN

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



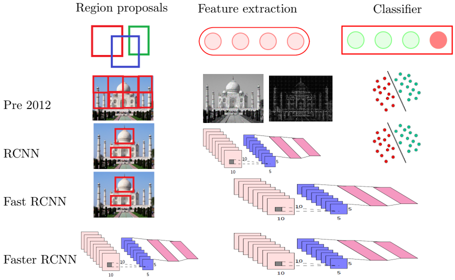
CS231n course, Stanford University

Faster R-CNN

§ Faster R-CNN based architectures won a lot of challenges including:

- Imagenet Detection
- Imagenet Localization
- COCO Detection
- COCO Segmentation

Faster R-CNN



- **Region Proposals:** CNN
- **Feature Extraction:** CNN
- **Classifier:** CNN

YOLO

§ The R-CNN pipelines separate proposal generation and proposal classification into two separate stages.

§ Can we have an end-to-end architecture which does both proposal generation and clasification simultaneously?

§ The solution gives the **YOLO (You Only Look Once)** architectures.

- J Redmon, S Divvala, R Girshick and A Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', CVPR 2016 - YOLO v1
- J Redmon and A Farhadi, 'YOLO9000: Better, Faster, Stronger', CVPR 2017 - YOLO v2
- J Redmon and A Farhadi, 'YOLOv3: An Incremental Improvement' arXiv preprint 2018 - YOLO v3

YOLO

- § The R-CNN pipelines separate proposal generation and proposal classification into two separate stages.
- § Can we have an end-to-end architecture which does both proposal generation and clasification simultaneously?
- § The solution gives the **YOLO (You Only Look Once)** architectures.
  - J Redmon, S Divvala, R Girshick and A Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', CVPR 2016 - YOLO v1
  - J Redmon and A Farhadi, 'YOLO9000: Better, Faster, Stronger', CVPR 2017 - YOLO v2
  - J Redmon and A Farhadi, 'YOLOv3: An Incremental Improvement' arXiv preprint 2018 - YOLO v3

YOLO

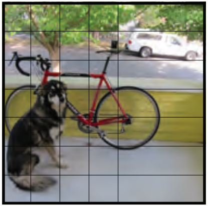
- The R-CNN pipelines separate proposal generation and proposal classification into two separate stages.
- Can we have an end-to-end architecture which does both proposal generation and clasification simultaneously?
- The solution gives the **YOLO (You Only Look Once)** architectures.
  - J Redmon, S Divvala, R Girshick and A Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', CVPR 2016 - YOLO v1
  - J Redmon and A Farhadi, 'YOLO9000: Better, Faster, Stronger', CVPR 2017 - YOLO v2
  - J Redmon and A Farhadi, 'YOLOv3: An Incremental Improvement' arXiv preprint 2018 - YOLO v3

YOLO

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

$P(dog)$



S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell

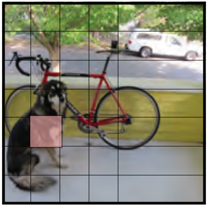
CS7015 course, IIT Madras

YOLO

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

$P(dog)$



S × S grid on input

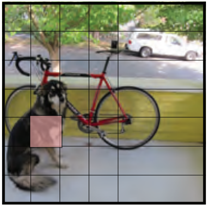
- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities

YOLO

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

$P(dog)$



S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object

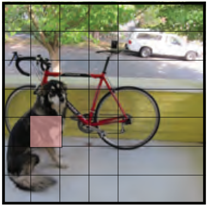
CS7015 course, IIT Madras

YOLO

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

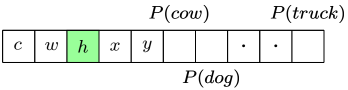
$P(dog)$

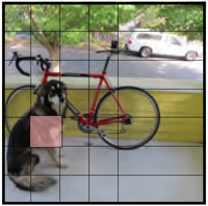


S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object

YOLO





S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object

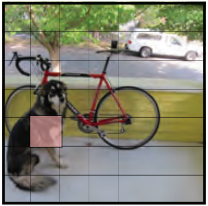
CS7015 course, IIT Madras

YOLO

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

$P(dog)$

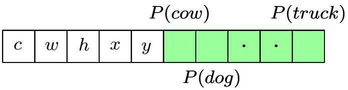


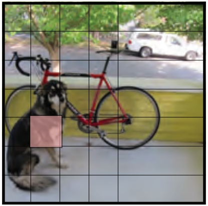
S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box

CS7015 course, IIT Madras

YOLO



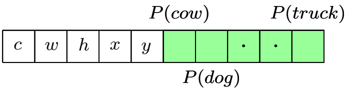


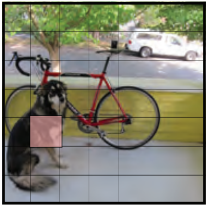
S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the K^{th} class (C values)

CS7015 course, IIT Madras

YOLO

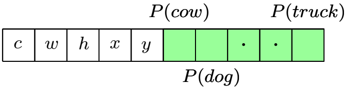


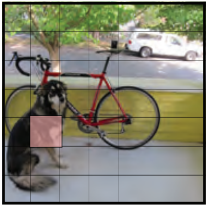


S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the Kth class (C values)
- The output layer should contain SxSxBx(5+C) elements

YOLO

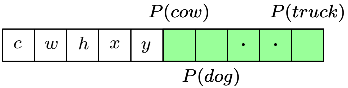


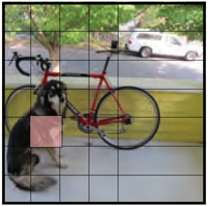


S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the Kth class (C values)
- The output layer should contain SxSxBx(5+C) elements
- However, each grid cell in YOLO predicts only one object even if there are B anchor boxes per cell

YOLO

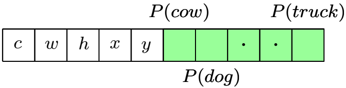


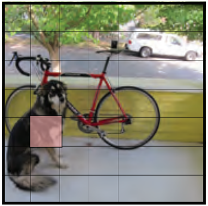


S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the Kth class (C values)
- The output layer should contain SxSxBx(5+C) elements
- The idea is each grid cell tries to make two boundary box predictions to locate a single object

YOLO





S × S grid on input

- Divide an image into S × S grids (S=7) and consider B (=2) anchor boxes per grid cell
- For each such anchor box in each cell we are interested in predicting 5 + C quantities
- Probability (confidence) that this anchor box contains a true object
- Width of the bounding box containing the true object
- Height of the bounding box containing the true object
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the Kth class (C values)
- The output layer should contain SxSxBx(5+C) elements
- Thus the output layer contains SxSx(Bx5+C) elements

YOLO

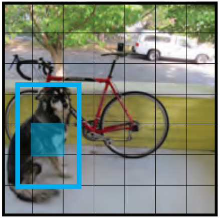
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object

S × S grid on input

CS7015 course, IIT Madras

YOLO

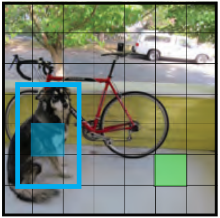
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object



S × S grid on input

YOLO

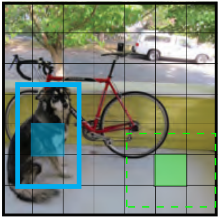
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object



S × S grid on input

YOLO

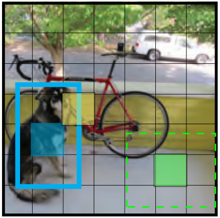
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object



S × S grid on input

YOLO

- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object

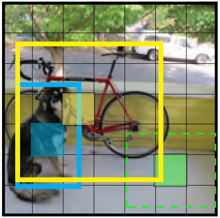


S × S grid on input

CS7015 course, IIT Madras

YOLO

- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object

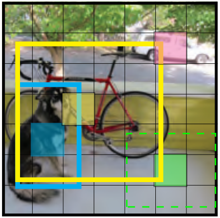


S × S grid on input

CS7015 course, IIT Madras

YOLO

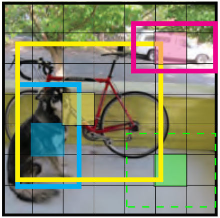
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object



S × S grid on input

YOLO

- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object

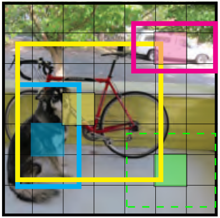


S × S grid on input

CS7015 course, IIT Madras

YOLO

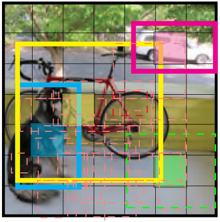
- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object



S × S grid on input

YOLO

- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object in it and the type of the object



S × S grid on input

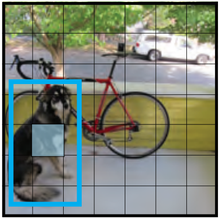
YOLO

- During inference/test phase, how do we interpret these $S \times S \times (B \times 5 + C)$ outputs?
- For each cell we compute the bounding box, its confidence about having any object it and the type of the object
- NMS is then applied to retain the most confident boxes

CS7015 course, IIT Madras

Training YOLO

- How do we train this network
- Consider a cell such that a true bounding box corresponds to this cell



- Initially the network with random weights will produce some values for these $(5 + C)$ values
- YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The following losses are computed
  - Classification Loss
  - Localization Loss
  - Confidence Loss

Training YOLO

Classification Loss

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

where, $\mathbb{1}_i^{\text{obj}} = 1$, if a ground truth object is in cell i, otherwise 0.
$\hat{p}_i(c)$ is the predicted probability of an object of class c in the i^{th} cell.
$p_i(c)$ is the ground truth label.

Training YOLO

Localization Loss: It measures the errors in the predicted bounding box locations and size. The loss is computed for the one box that is responsible for detecting the object.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(x_i - \hat{x}_i \right)^2 + \left(x_i - \hat{x}_i \right)^2 \right]$$

$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where, $\mathbb{1}_{ij}^{\text{obj}} = 1$, if j^{th} bounding box is responsible for detecting the ground truth object in cell i, otherwise 0.
By square rooting the box dimensions some parity is maintained for different size boxes. Absolute errors in large boxes and small boxes are not treated same.

Training YOLO

Confidence Loss: For a box responsible for predicting an object

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

where, $\mathbb{1}_{ij}^{\text{obj}} = 1$, if j^{th} bounding box is responsible for detecting the ground truth object in cell i, otherwise 0.
$\hat{C}_i$ is the predicted probability that there is an object in the i^{th} cell.
C_i is the ground truth label (of whether an object is there).

Training YOLO

Confidence Loss: For a box that predicts 'no object' inside

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

where, $\mathbb{1}_i^{\text{obj}} = 1$, if j^{th} bounding box is responsible for predicting 'no object' in cell i, otherwise 0.
$\hat{C}_i$ is the predicted probability that there is an object in the i^{th} cell.
C_i is the ground truth label (of whether an object is there).
The total loss is the sum of all the above losses

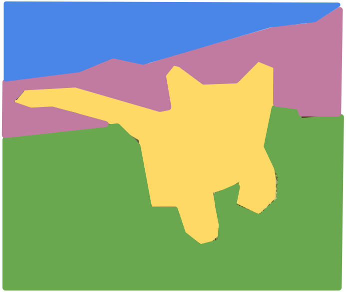
Training YOLO

Method	Pascal 2007 mAP	Speed
DPM v5	33.7	0.07 FPS — 14 sec/ image
RCNN	66.0	0.05 FPS — 20 sec/ image
Fast RCNN	70.0	0.5 FPS — 2 sec/ image
Faster RCNN	73.2	7 FPS — 140 msec/ image
YOLO	69.0	45 FPS — 22 msec/ image

CS7015 course, IIT Madras

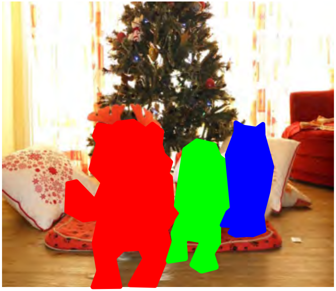
Segmentation

Semantic Segmentation



GRASS, CAT, TREE, SKY

Instance Segmentation

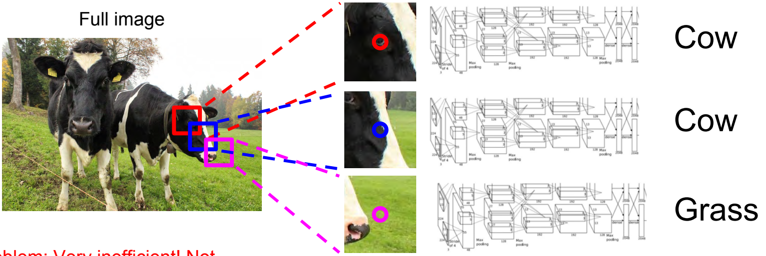


DOG, DOG, CAT

Source: cs231n course, Stanford University

Segmentation

Semantic Segmentation Idea: Sliding Window



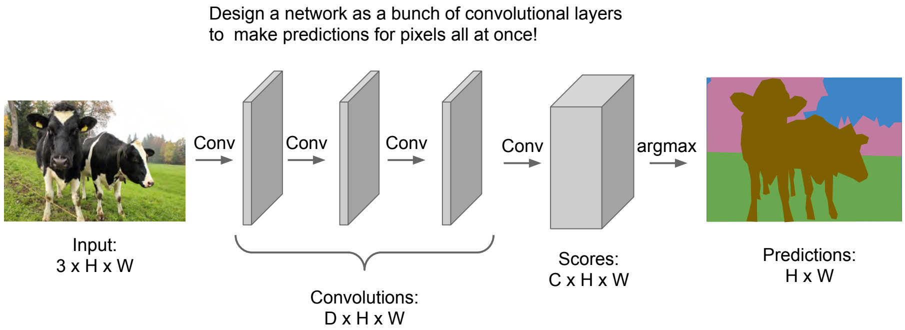
Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Source: cs231n course, Stanford University

Segmentation

Semantic Segmentation Idea: Fully Convolutional

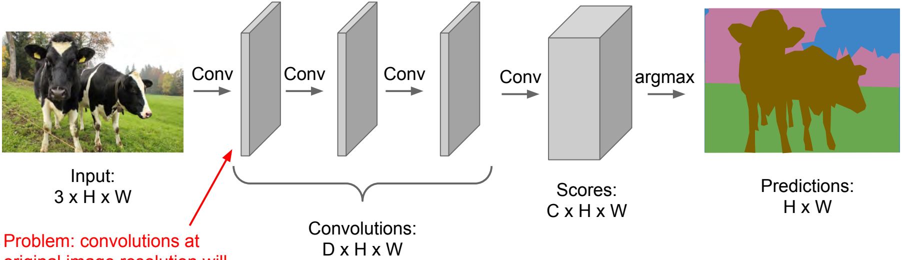


Source: cs231n course, Stanford University

Segmentation

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Problem: convolutions at original image resolution will be very expensive ...

Source: cs231n course, Stanford University

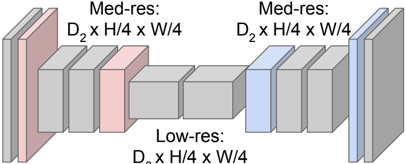
Segmentation

Downsampling: Pooling, strided convolution

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

Upsampling: ???

Input: 3 x H x W



Med-res: D_2 x H/4 x W/4

Med-res: D_2 x H/4 x W/4

Low-res: D_3 x H/4 x W/4

High-res: D_1 x H/2 x W/2

High-res: D_1 x H/2 x W/2

Predictions: H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Source: cs231n course, Stanford University

Segmentation

In-Network upsampling: “Unpooling”

Nearest Neighbor

1	2
3	4

Input: 2 x 2

→

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

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

→

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

Output: 4 x 4

Source: cs231n course, Stanford University

Segmentation

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

→ … →
Rest of the network

Max Unpooling
Use positions from pooling layer

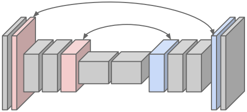
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

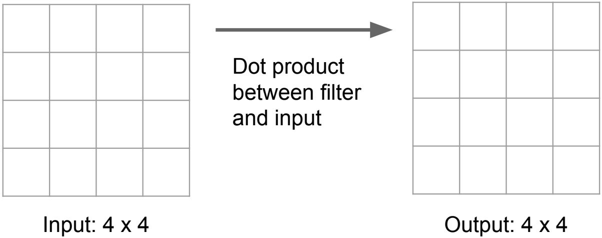


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

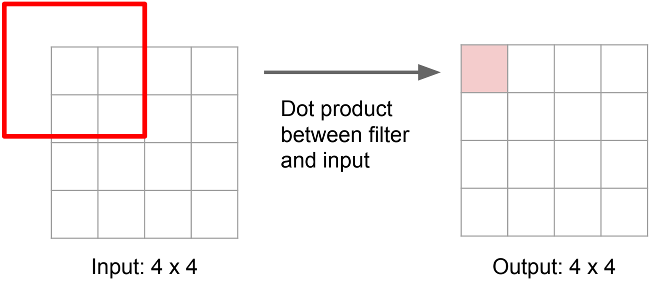


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

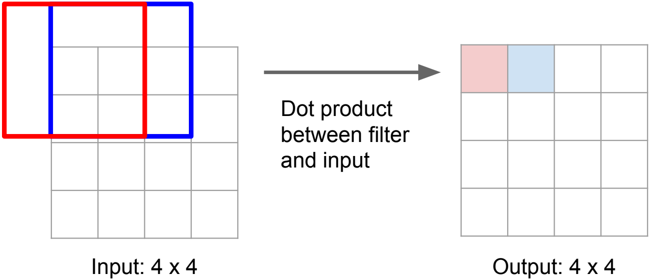


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

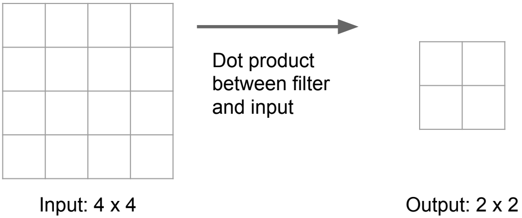


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

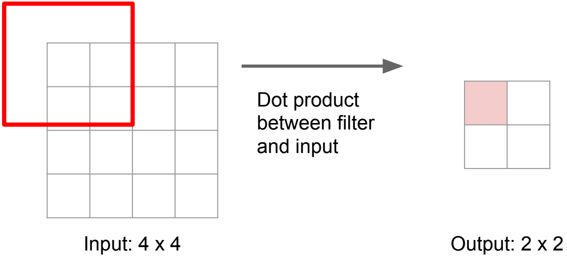


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

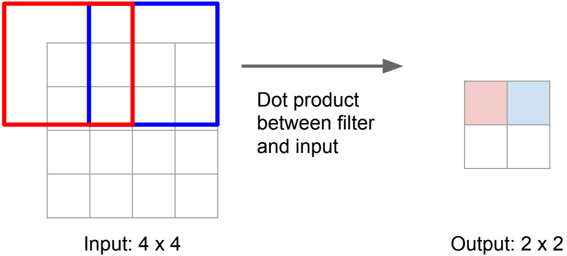


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

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

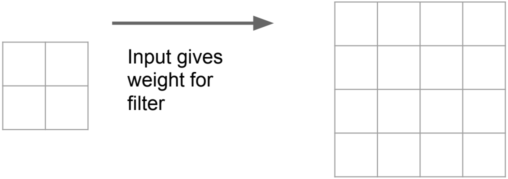


Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 1 pad 0



Input: 2 x 2

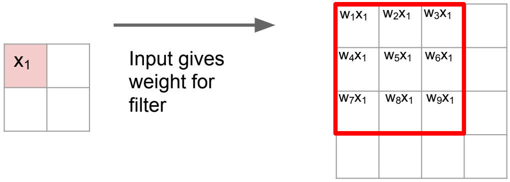
Output: 4 x 4

Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 1 pad 0



Input: 2 x 2

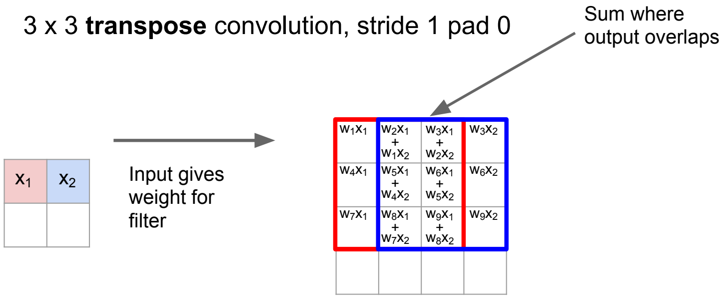
Output: 4 x 4

Source: cs231n course, Stanford University

Segmentation

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 1 pad 0



Input: 2 x 2

Output: 4 x 4

Source: cs231n course, Stanford University

Segmentation

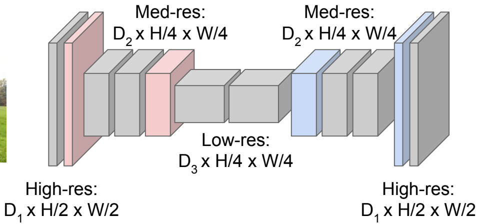
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: Unpooling or strided transpose convolution

Input: 3 x H x W

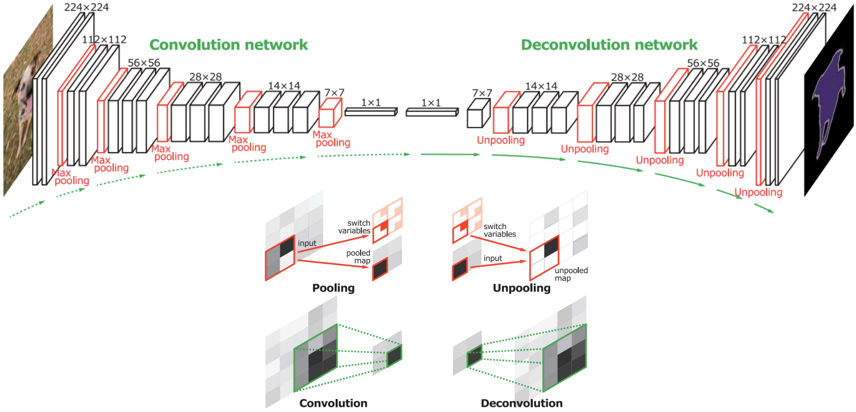


Predictions: H x W

J Long, E Shelhamer and T Darrell, 'Fully Convolutional Networks for Semantic Segmentation', CVPR 2015

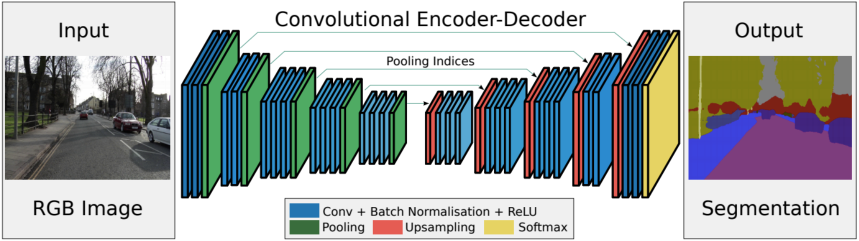
Source: cs231n course, Stanford University

Segmentation: Deconvolutional Network



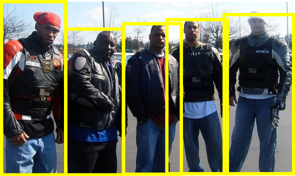
H Noh, S Hong and B Han, 'Learning Deconvolution Network for Semantic Segmentation', ICCV 2015

Segmentation: SegNet



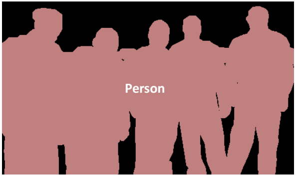
H Noh, S Hong and B Han, 'SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation', PAMI 2017

Instance Segmentation



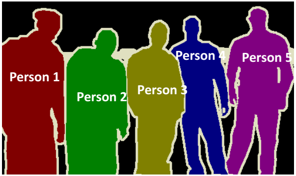
Object Detection

✓



Semantic Segmentation

✓



Instance Segmentation

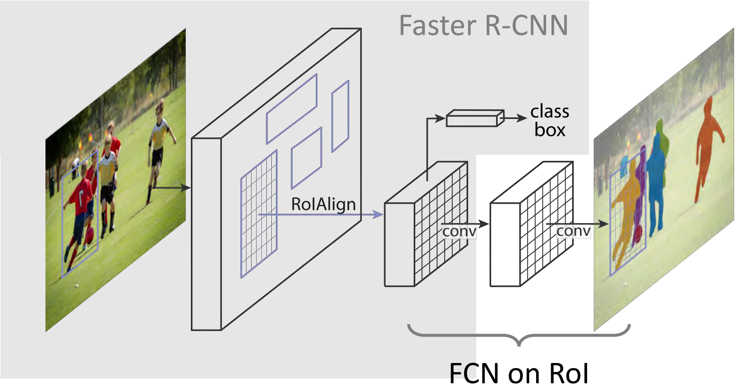
?

- Instance segmentation not only wants to detect individual object instances but also wants to have a segmentation mask of each instance
- What can be a naive idea?

Source: Kaiming He, ICCV 2017

Instance Segmentation

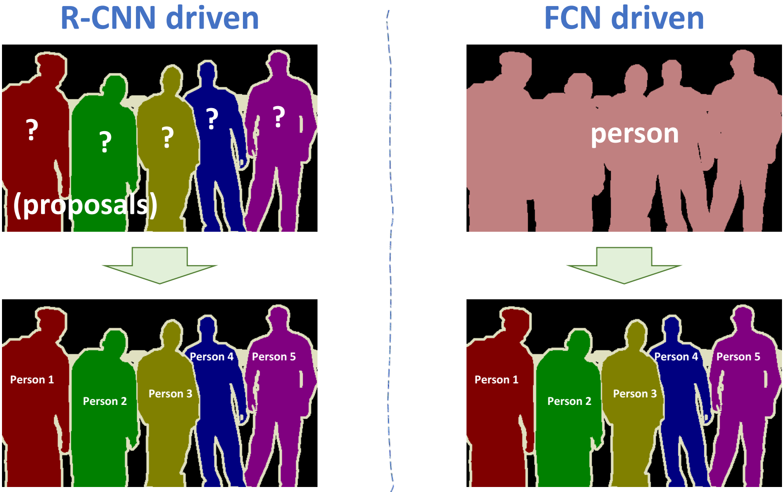
- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



Source: Kaiming He, ICCV 2017

Instance Segmentation: Broad Strategies

Instance Segmentation Methods

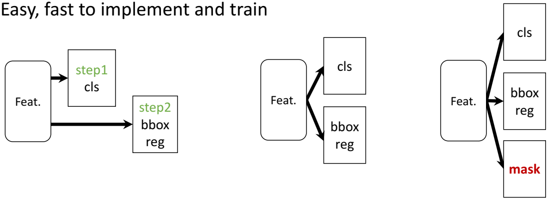


Source: Kaiming He, ICCV 2017

Instance Segmentation: Mask-RCNN

Parallel Heads

- Easy, fast to implement and train



(slow) R-CNN Fast/er R-CNN Mask R-CNN

Mask R-CNN is conceptually simple: Faster R-CNN has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask. Mask R-CNN is thus a natural and intuitive idea. But the additional mask output is distinct from the class and box outputs, requiring extraction of much *finer* spatial layout of an object. Next, we introduce the key elements of Mask R-CNN, including pixel-to-pixel alignment, which is the main missing piece of Fast/Faster R-CNN.

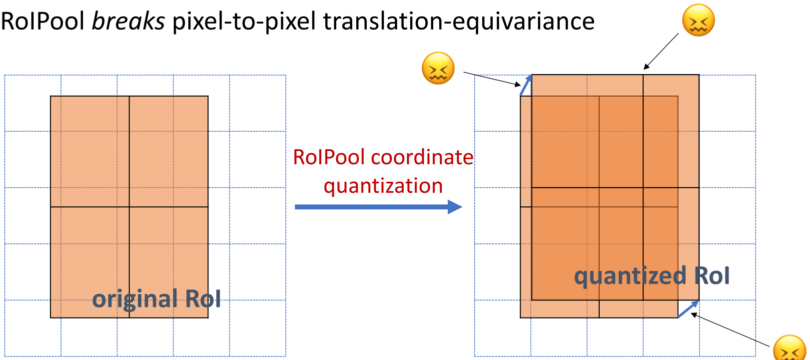
Source: Kaiming He, ICCV 2017

Instance Segmentation: Mask-RCNN

- Mask R-CNN adopts the same two-stage procedure with identical first stage [*i.e.*, RPN] as R-CNN
- In second stage in addition to class prediction and bounding box regression Mask-RCNN, **in parallel**, outputs a binary mask for each RoI
- The mask branch has Km^2 dimensional output for each RoI [binary mask of $m \times m$ resolution one for each K classes] boxes
- RoIPool breaks pixel-to-pixel translation-equivariance

Instance Segmentation: Mask-RCNN

- RoIPool *breaks* pixel-to-pixel translation-equivariance



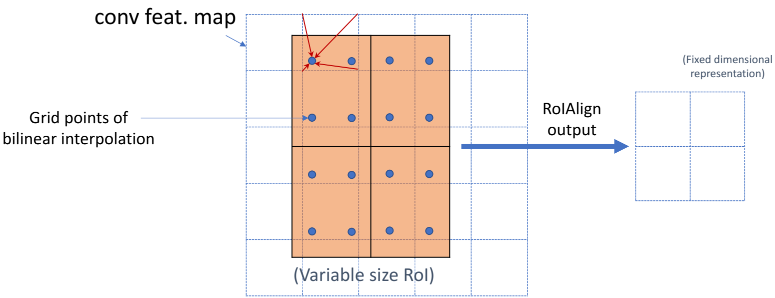
Source: Kaiming He, ICCV 2017

Instance Segmentation: Mask-RCNN

RoIAlign

FAQs: how to sample grid points within a cell?

- 4 regular points in 2x2 sub-cells
- other implementation could work



Source: Kaiming He, ICCV 2017

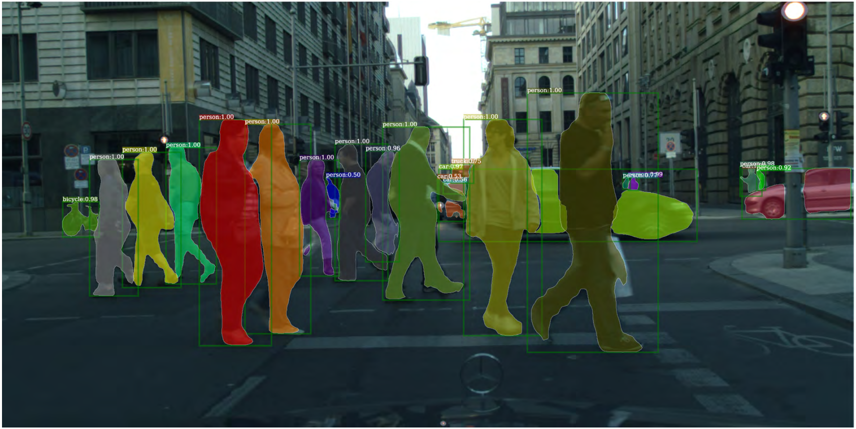
Instance Segmentation: Mask-RCNN

disconnected object

Mask R-CNN results on COCO

Source: Kaiming He, ICCV 2017

Instance Segmentation: Mask-RCNN



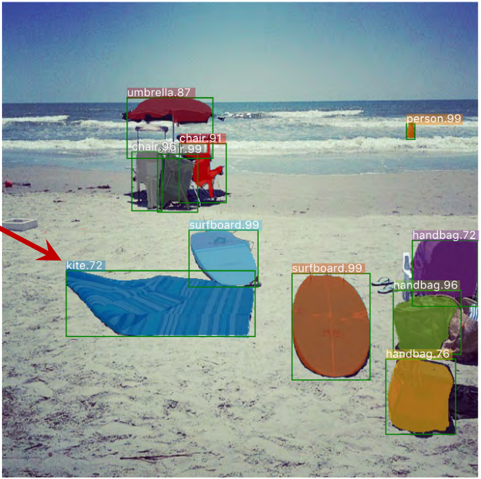
Mask R-CNN results on CityScapes

Source: Kaiming He, ICCV 2017

Instance Segmentation: Mask-RCNN

Failure case: recognition

not a kite



Mask R-CNN results on COCO

Source: Kaiming He, ICCV 2017