Компьютерное зрение

Задача компьютерного зрения

- Понять, что запечатлено на изображении

Мы видим

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Компьютер видит

Задача компьютерного зрения

- «To see means to know what is where by looking»
  - *David Marr,* ***Vision****, 1982*

- Понять, что запечатлено на изображении

- Что это в действительности обозначает?
  - Зрение - источник семантической информации о мире
  - Зрение - источник метрической информации о трехмерном мире

Семантическая информация

Slide credit: Fei-Fei, Fergus & Torralba

Классификация сцены

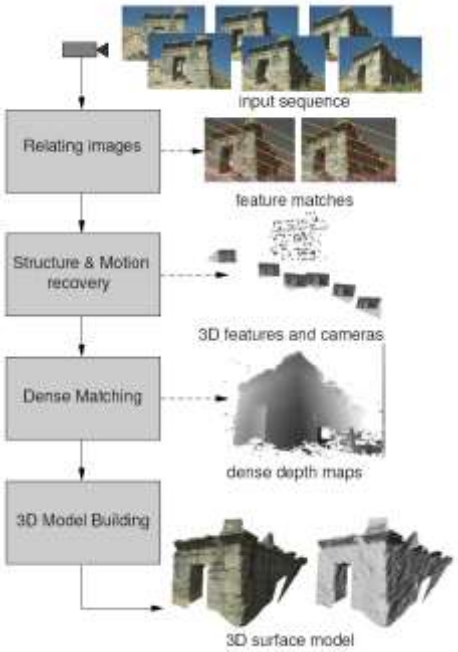
Поиск и локализация объектов

Семантическая сегментация

Качественная информация

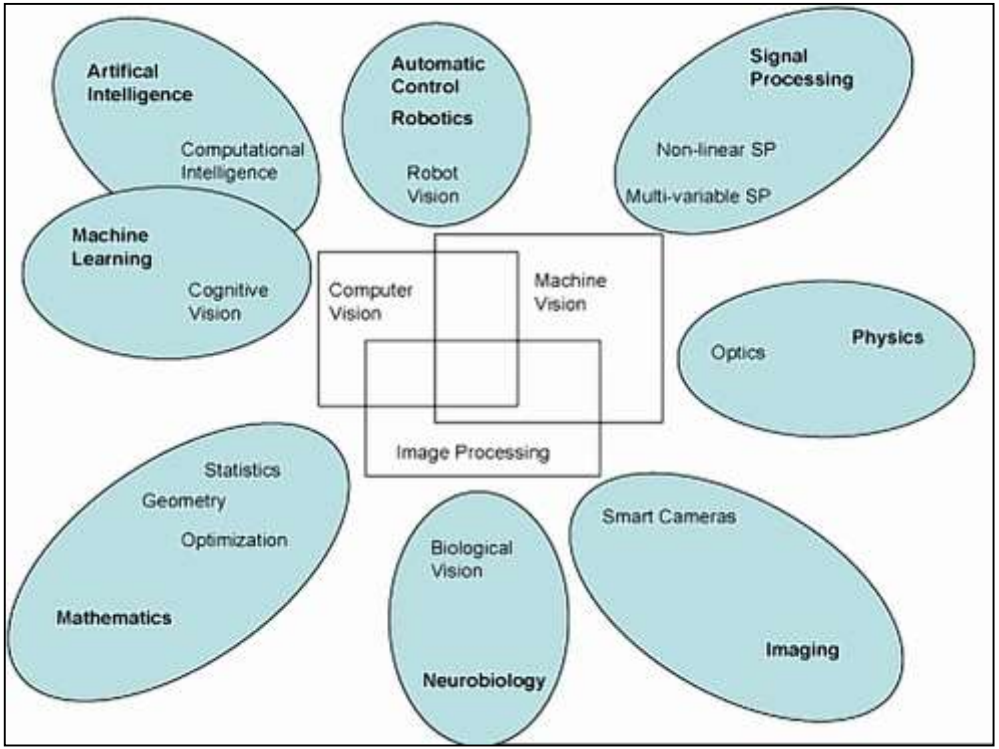
Метрическая информация

Структура из движения



Моделирование по пользовательским снимкам

Смежные дисциплины



Зрение… принятые названия

- Обработка изображений (Image processing)
  - На входе и выходе изображение
- Анализ изображений (Image analysis)
  - Фокусируется на работе с 2D изображениями
- Распознавание образов (Pattern recognition)
  - Распознавание, обучение на абстрактных числовых величинах, полученных в том числе и из изображений
- Компьютерное зрение (Computer vision)
  - Изначально воостановление 3д структуры по 2д изображениям, сейчас шире, как принятие решений о физических объектах, основываясь на их изображениях
- Фотограмметрия (Photogrammetry)
  - Исторически измерение расстояний между объектами по 2D изображениям
- Машинное зрение (Machine vision)
  - Обычно понимается как решение промышленных, производственных задач (сложилось исторически)

Зачем?

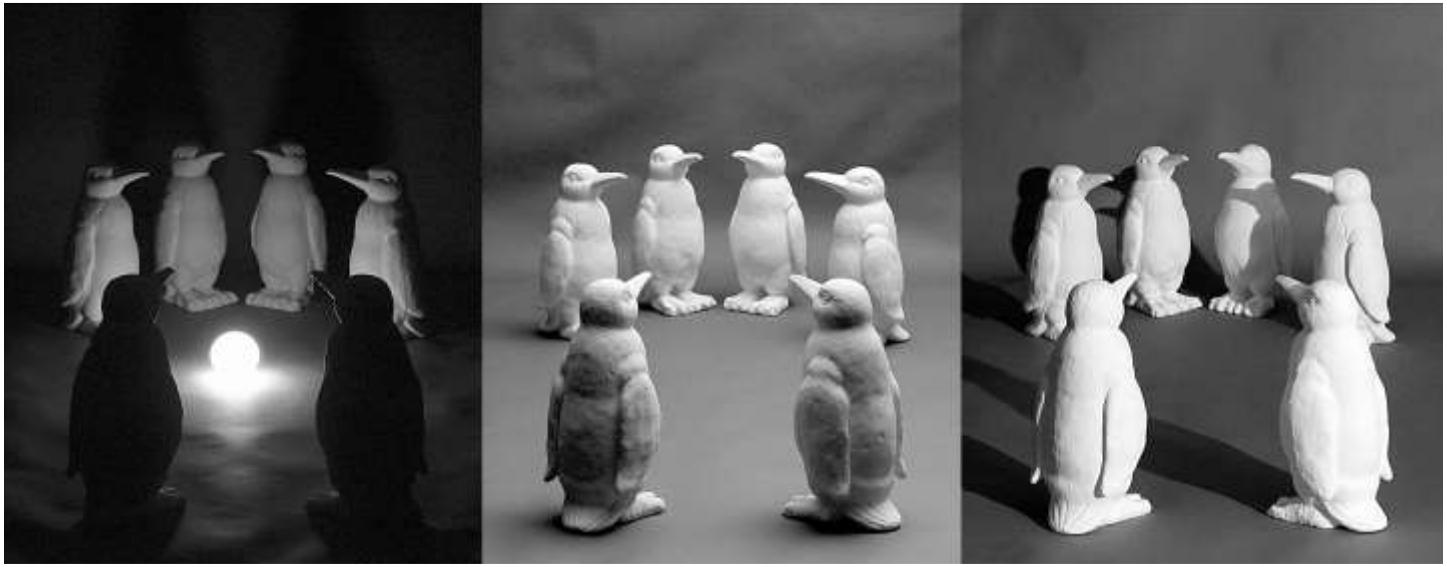
- Полезно – много практических применений
- Интересно – наглядное применение массы математических методов
- Сложно
  - 25+% мозга человека отвечает за зрение
  - «ИИ-полная» задача – решение задачи зрения на уровне человека равносильно решению задачи искусственного интеллекта

Почему зрение – это сложно?

Точка наблюдения

Michelangelo 1475-1564

Освещение



Масштаб

Перекрытие

Magritte, 1957

Движение

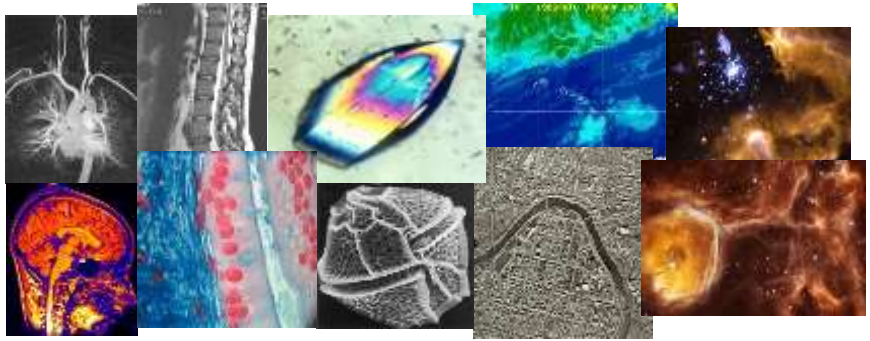
Внутриклассовая изменчивость

Решаемые задачи

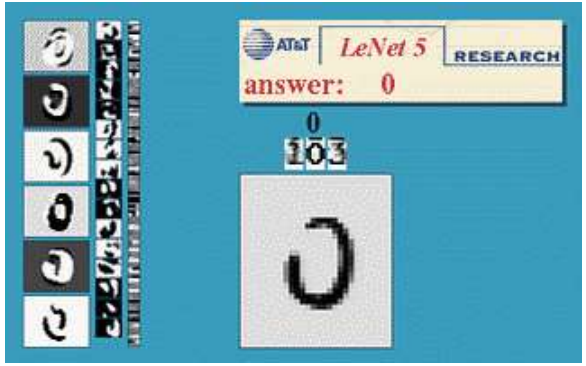
- Изображения и видео повсюду
- Бурно растущая область
  - Обработка – улучшение качества, ретушь, изменение размера и формы, композиция
  - Интернет – поиск, аннотация, поиск дубликатов, распознавание объектов
  - Видеонаблюдение – отслеживание, распознавание объектов, распознавание жестов и событий
  - Промышленные системы – диагностика, контроль качества
  - Спецэффекты в кино – композиция, монтаж фонов, захват движения

Распространение изображений

Google Image Search Picasa flickr webshots picsearch YouTube Broadcast Yourself



Распознавание текста



Digit recognition, AT&T labs
http://www.research.att.com/~yann/

License plate readers
http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

Детектор лиц (2001)

Алгоритм Viola-Jones – первый быстрый и надежный алгоритм поиска лиц. Демонстрация силы машинного обучения.

Распознавание объектов

- Microsoft Research

Lincoln

Умные машины

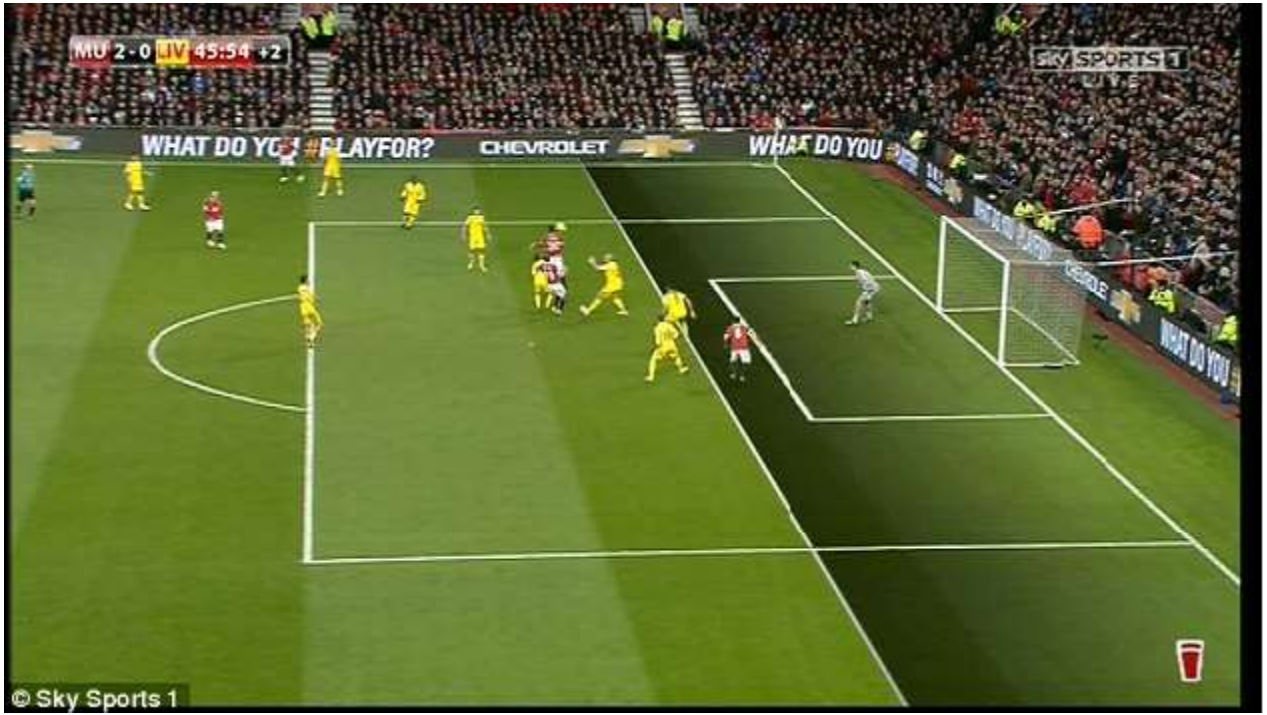
- Mobileye
  - Топ-модели от BMW, GM, Volvo
  - К 2010: 70% производителей машин

Умные машины

Захват движения

Pirates of the Carribean, Industrial Light and Magic

Спортивные соревнования



Sportvision. Offside.

Зрение в космосе

NASA'S Mars Exploration Rover Spirit.

Системы зрения использовались для:

- Склейка панорам
- 3D моделирование местности
- Поиск препятствий, определение местоположения

Трехмерные карты

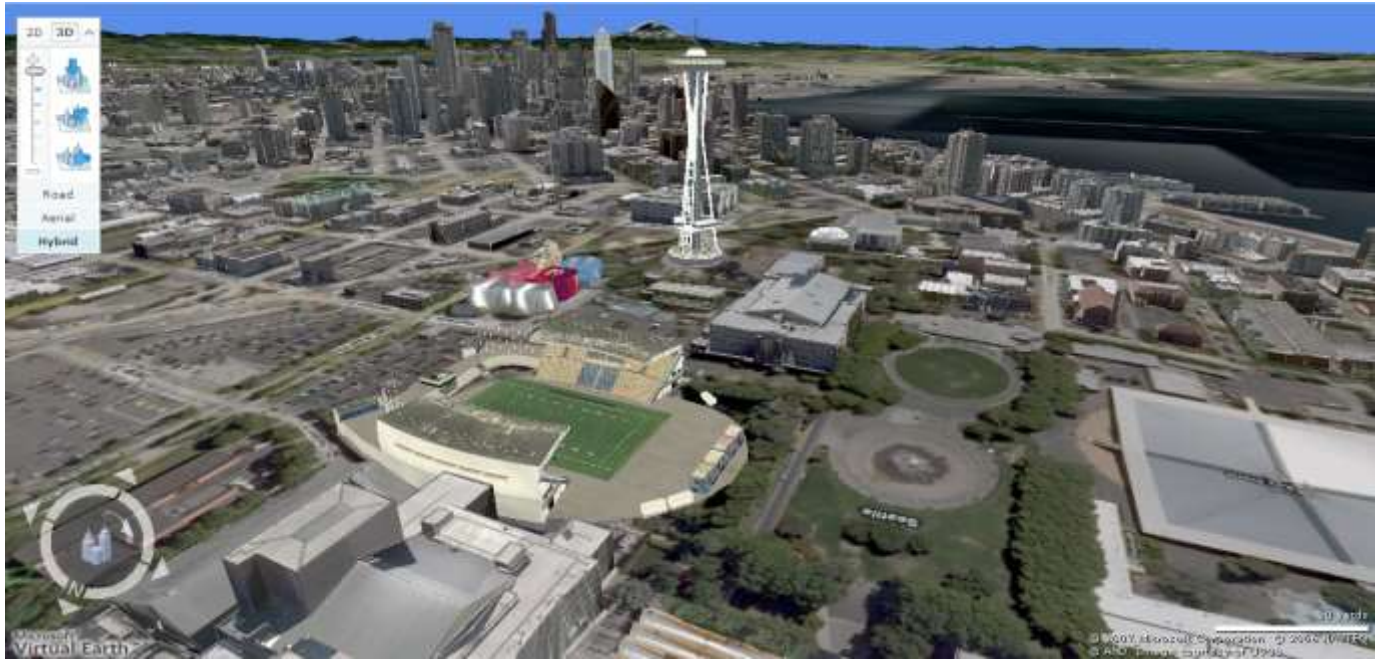
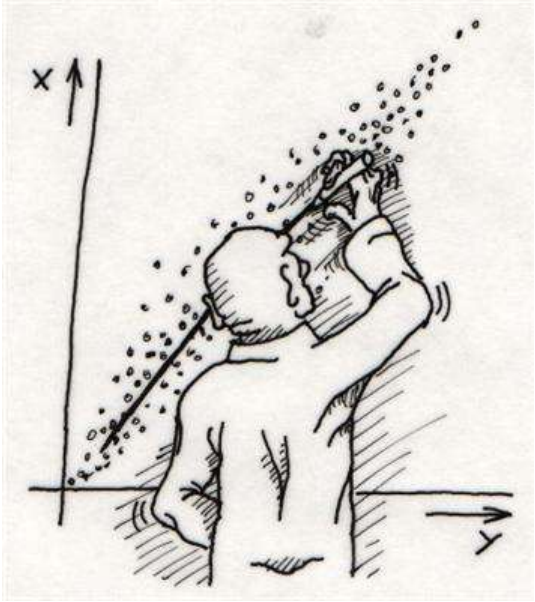
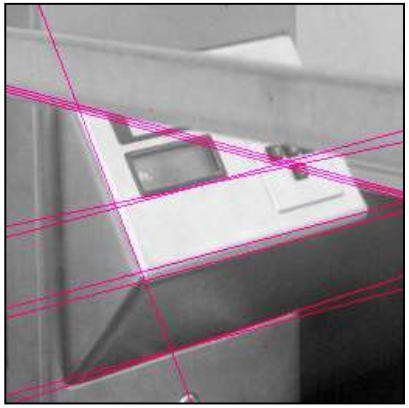


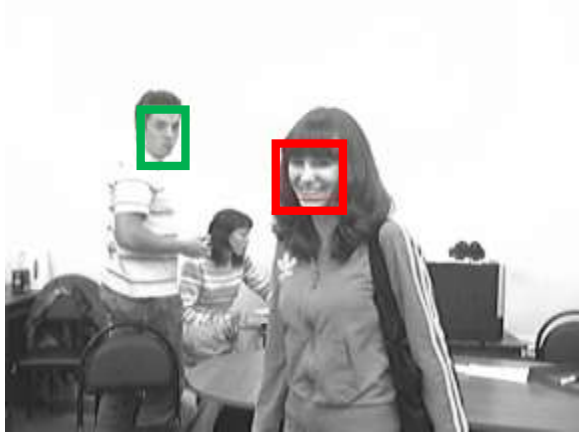
Image from Microsoft's Virtual Earth
(see also: Google Earth)

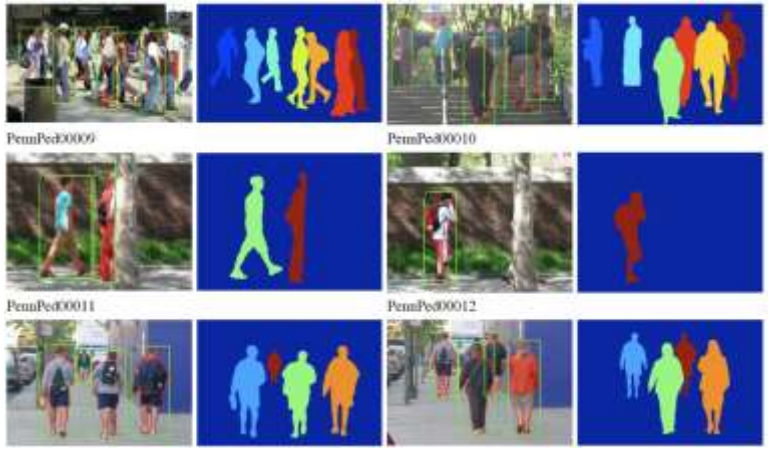
Сопоставление изображений

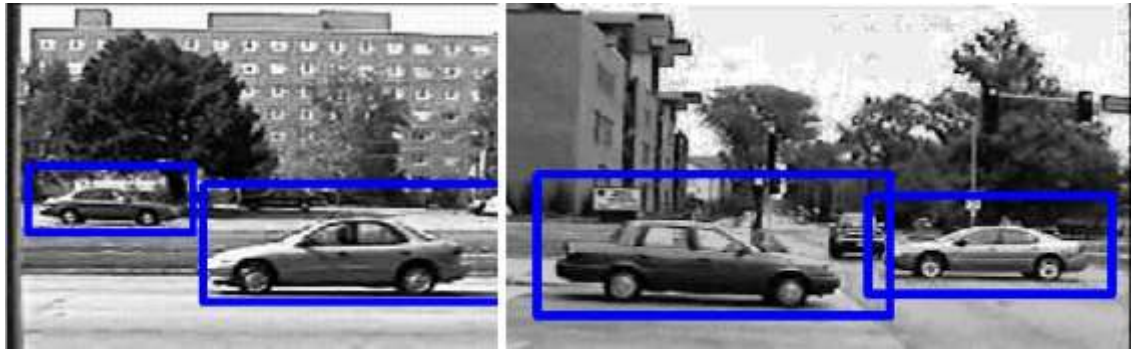




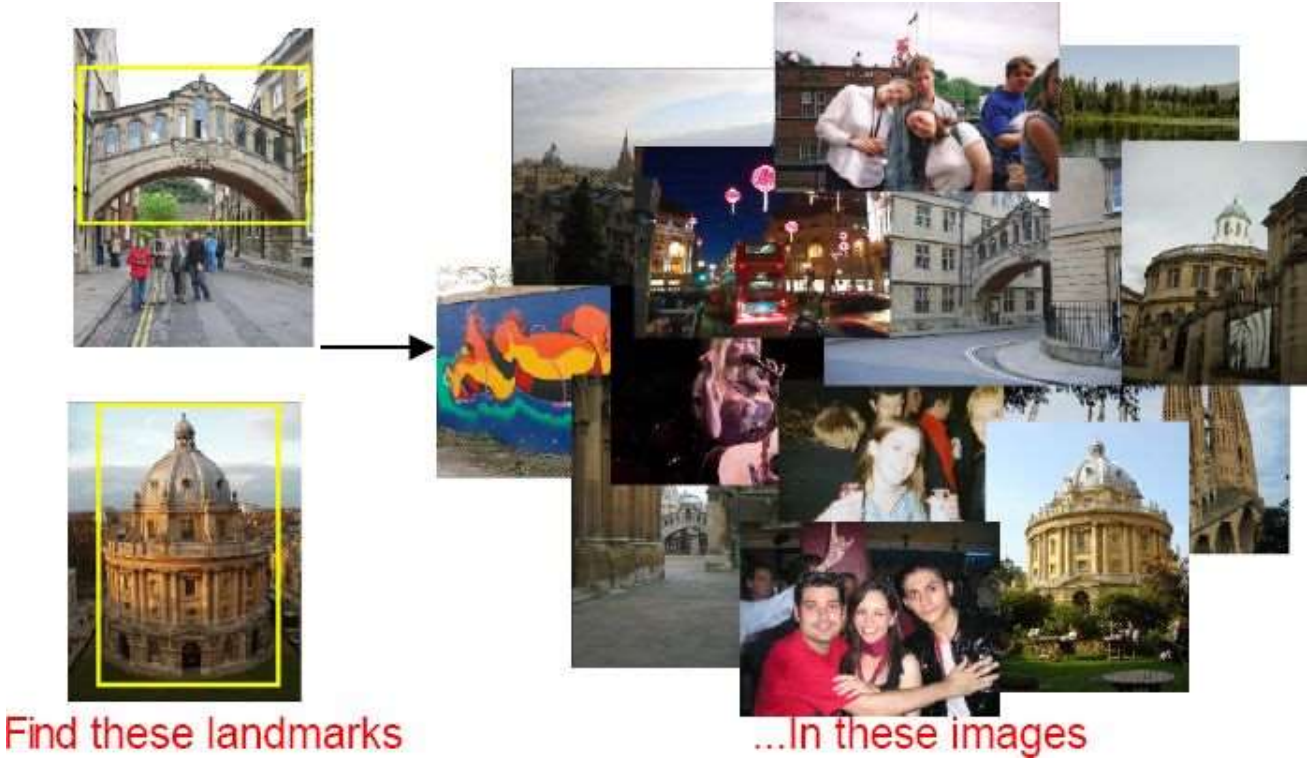
Поиск и локализация объектов







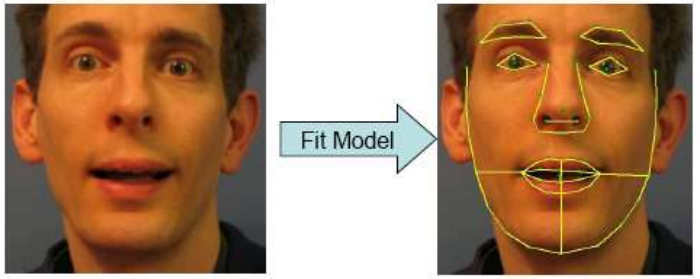
Поиск изображений в базе

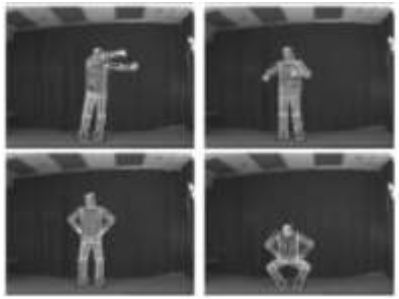


Find these landmarks

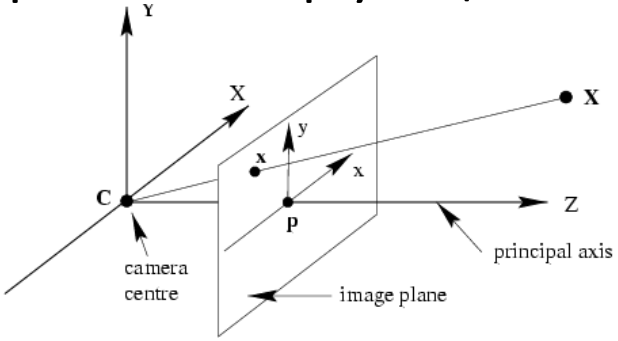
...In these images

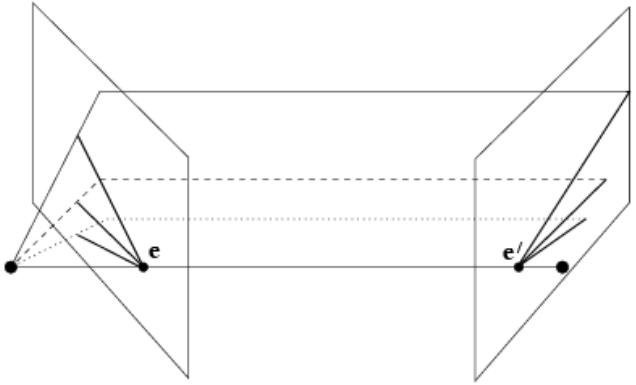
Часть 2: Изображения человека





Часть 2: Трехмерная реконструкция





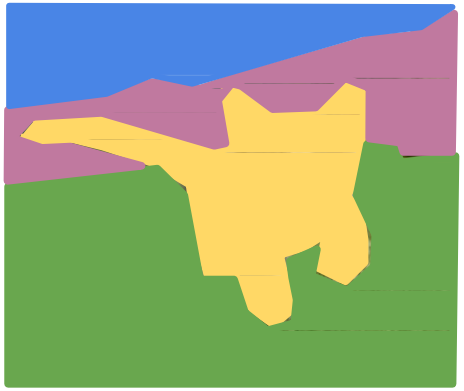
Задачи компьютерного зрения

Классификация

CAT

Только сам объект

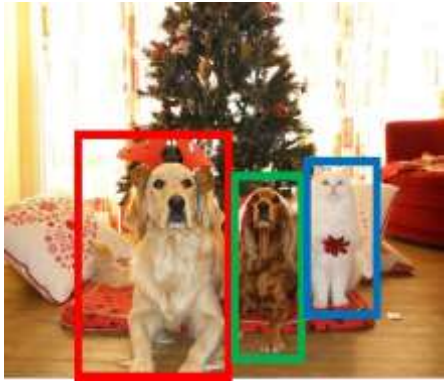
Семантическая сегментация



GRASS, CAT, TREE, SKY

Нет объектов, только пиксели

Детекция



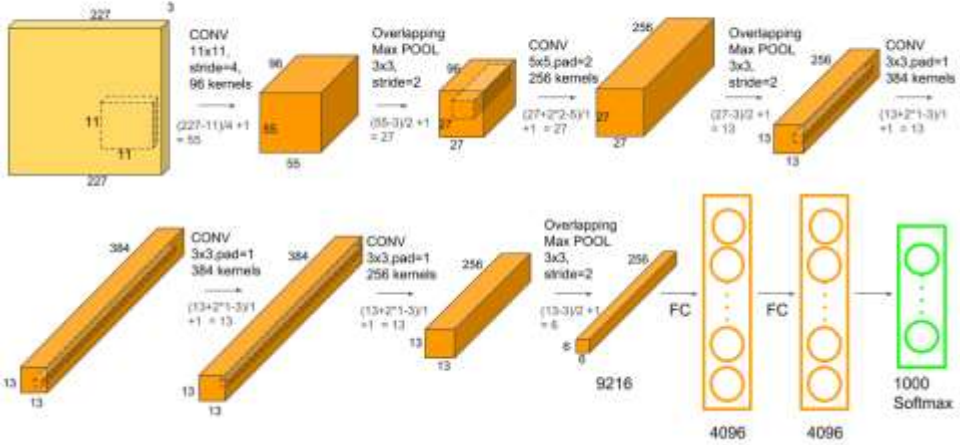
DOG, DOG, CAT

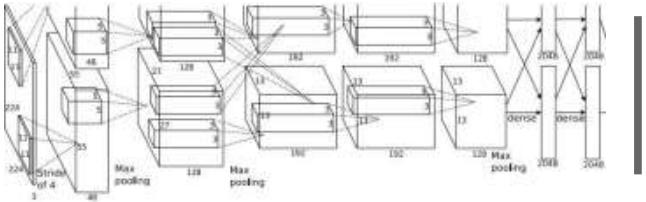
Instance сегментация

DOG, DOG, CAT

Несколько объектов

Image Classification. AlexNet





Vector:
4096

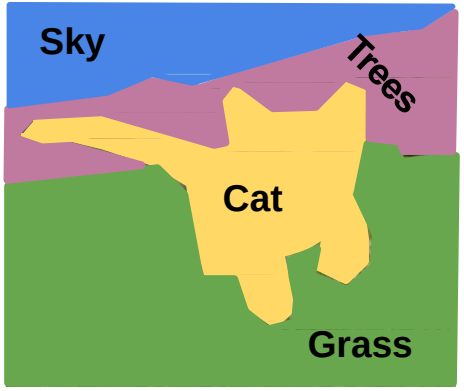
Fully-Connected:
4096 to 1000

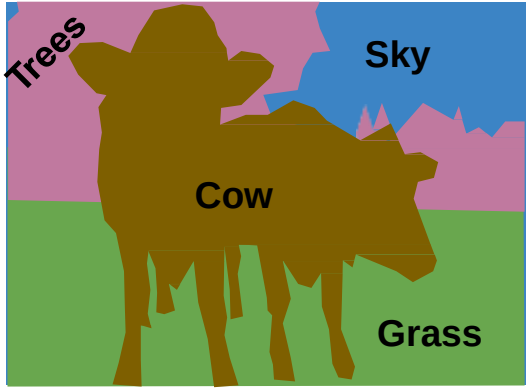
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Семантическая сегментация

Каждый пиксель относим к какой-то категории объектов

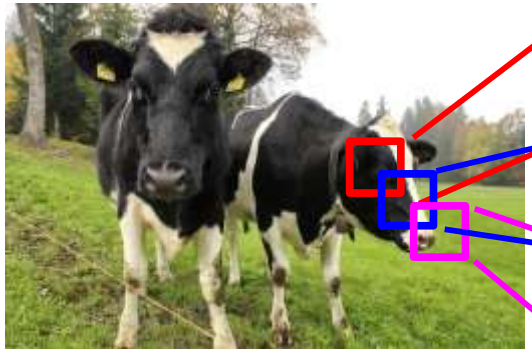
Не различаем сами объекты от фона, думаем только о пикселях



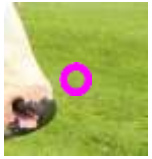


Идея семантической сегментации: скользящее окно

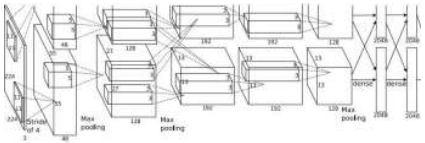
Full image



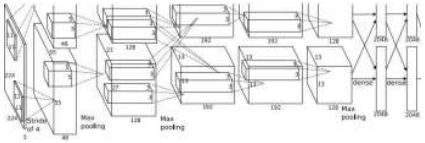
Извлечь кусок картинки



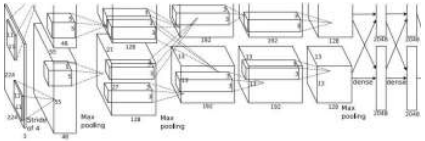
Классифицировать центральный пиксель с помощью CNN



Cow



Cow



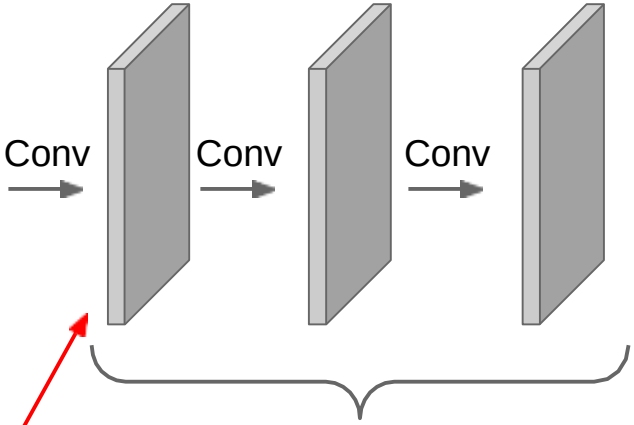
Grass

Проблема:
Крайне неэффективно перебирать все пиксели!

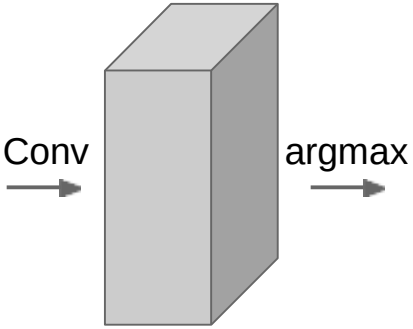
Идея семантической сегментации: полностью свёрточная сеть

Создать сеть как последовательность сверточных слоев, чтобы предсказать для всех пикселей за раз.

Input:
3 x H x W



Convolutions:
D x H x W



Scores:
C x H x W

Predictions:
H x W

Проблема: свёртки с сохранением оригинального разрешения очень дорогие

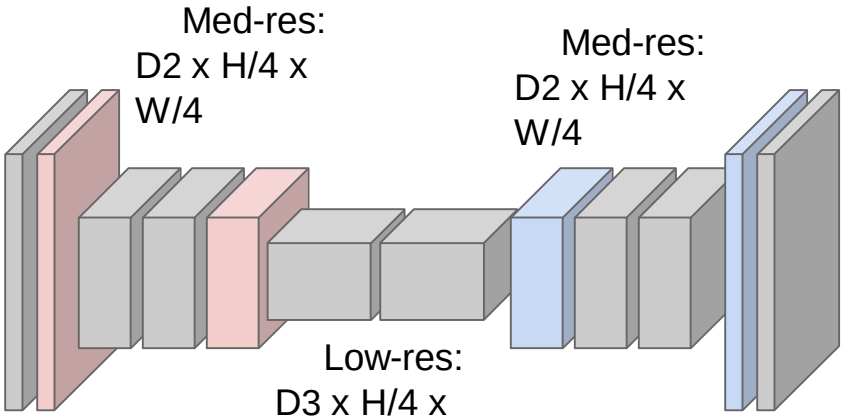
Идея семантической сегментации: полностью свёрточная сеть

Понижение: Субдискретизация (pooling), пошаговое свёртывание (strided convolution)

Создать сеть как последовательность сверточных слоев с понижением и повышением разрешения в самой сети.

Повышение: ???

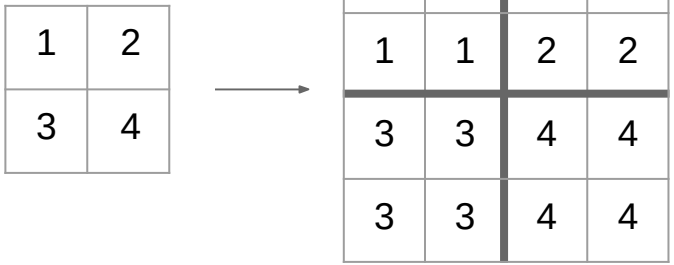
Input:
3 x H x W



Predictions:
H x W

Внутрисетевое повышение разрешения: “Unpooling”

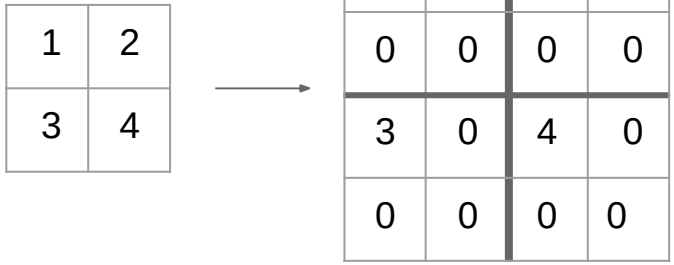
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

Внутрисетевое повышение разрешения: “Max Unpooling”

Max Pooling
Запоминаем, какой элемент был максимальным

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
Используем позиции из pooling слоя

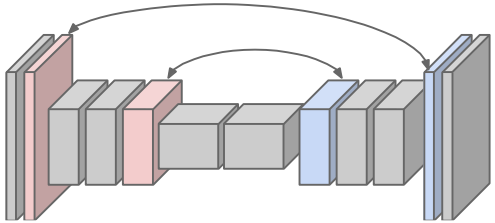
1	2
3	4

Input: 2 x 2

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

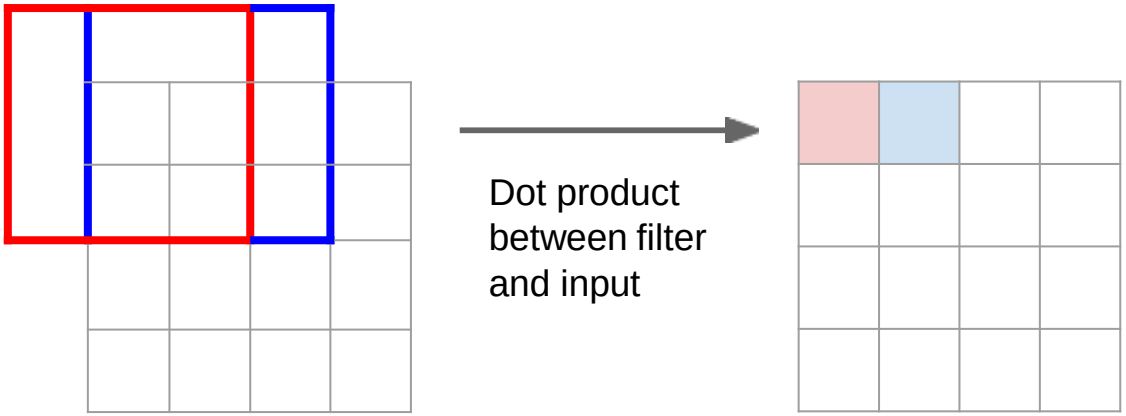
Output: 4 x 4

Соответствующие пары слоев понижения и повышение разрешения



Learnable Upsampling: Transpose Convolution

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

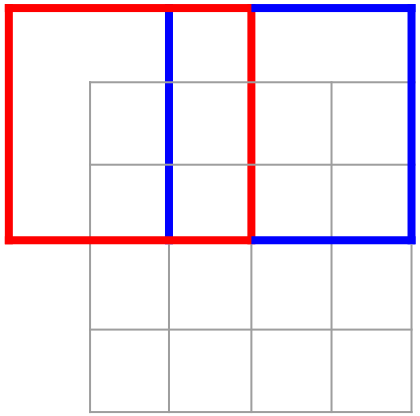


Input: 4 x 4

Output: 4 x 4

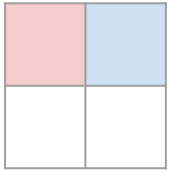
Learnable Upsampling: Transpose Convolution

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



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

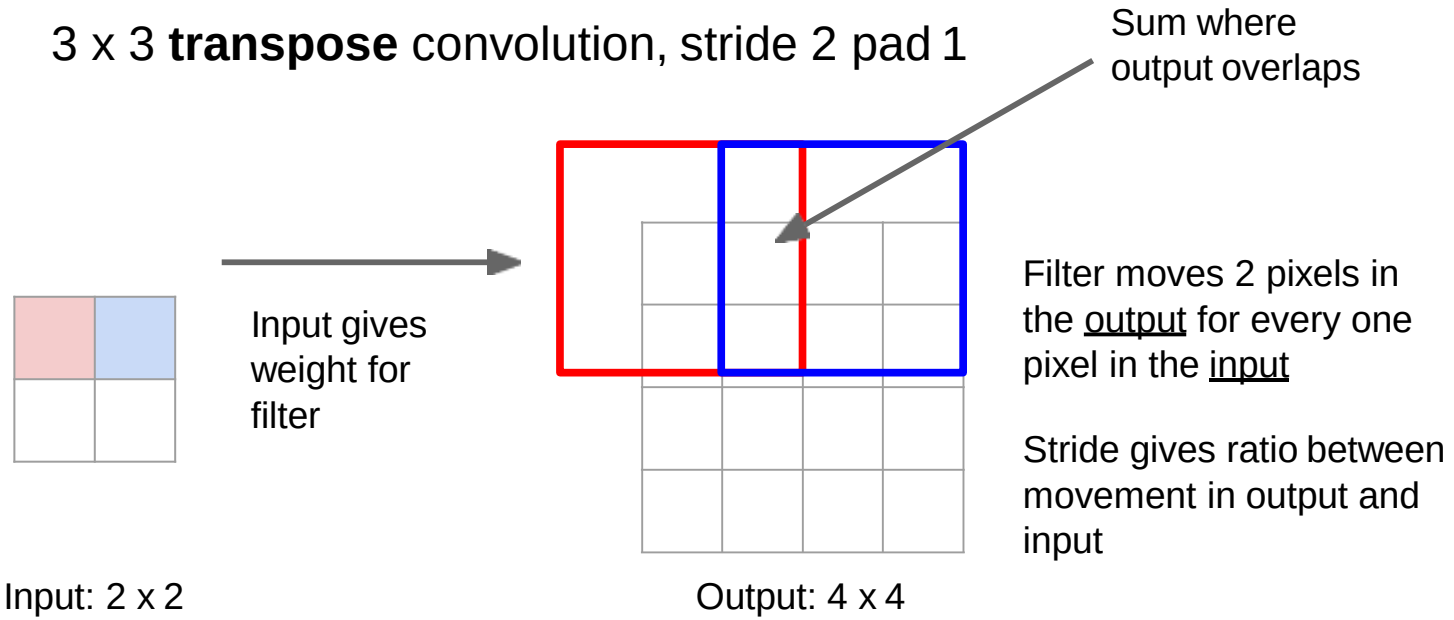
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

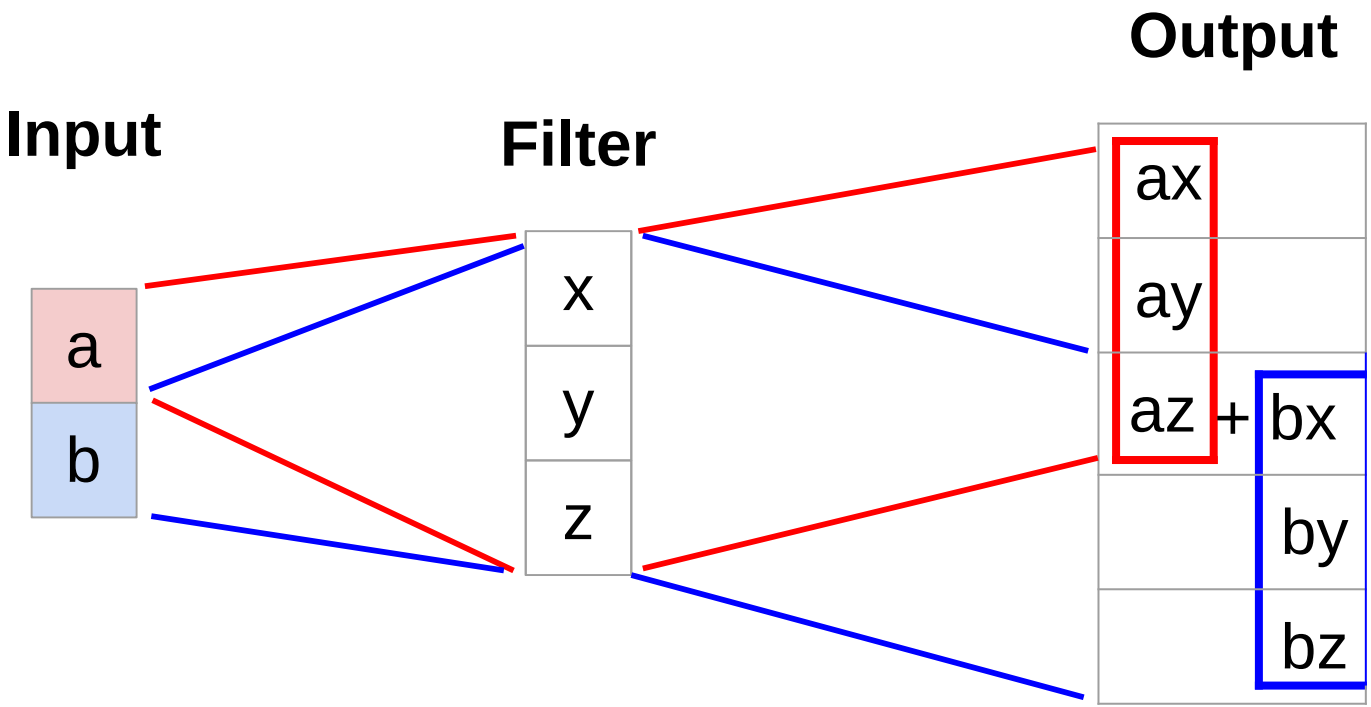
Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

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

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, convolution transpose is no longer a normal convolution!

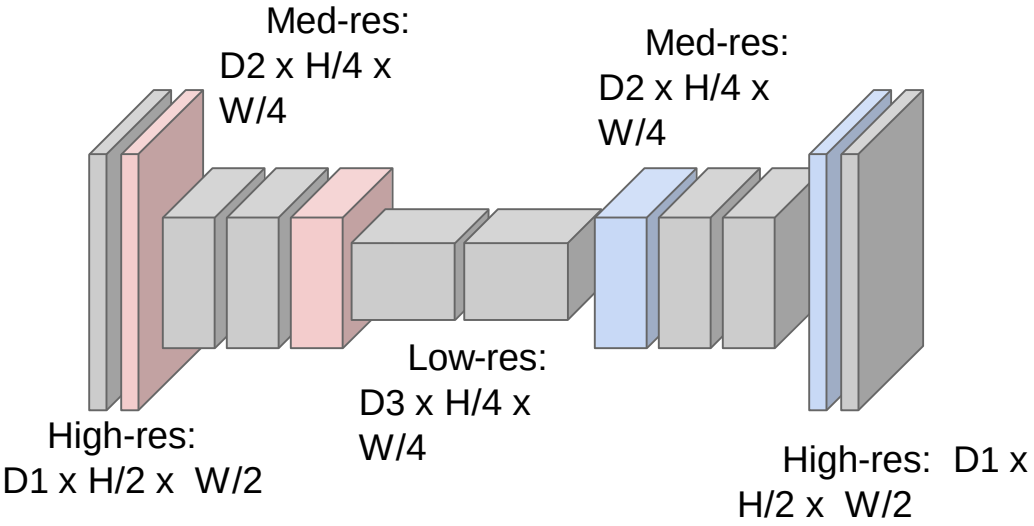
Идея семантической сегментации: полностью свёрточная сеть

Понижение: Субдискретизация (pooling), пошаговая свёртка (strided convolution)

Создать сеть как последовательность сверточных слоев с понижением и повышением разрешения в самой сети.

Повышение: Unpooling или пошаговая транспонированная свёртка

Input:
3 x H x W



Predictions:
H x W

Детекция

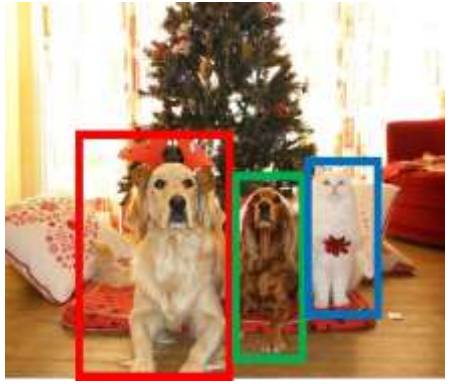
Классификация

CAT

Семантическая сегментация

GRASS, CAT, TREE, SKY

Детекция



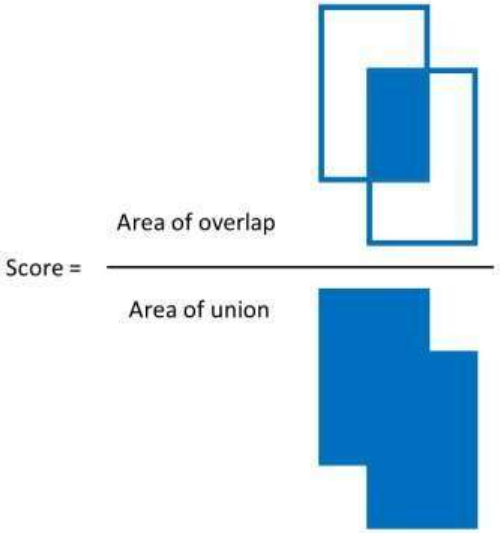
DOG, DOG, CAT

Instance сегментация

DOG, DOG, CAT

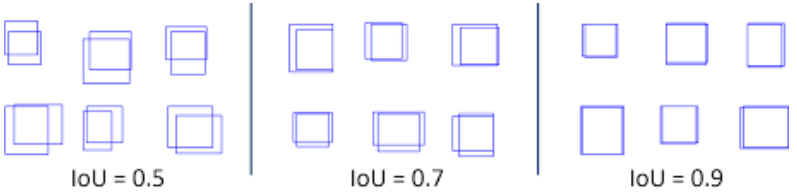
Несколько объектов

Критерий обнаружения (IoU)



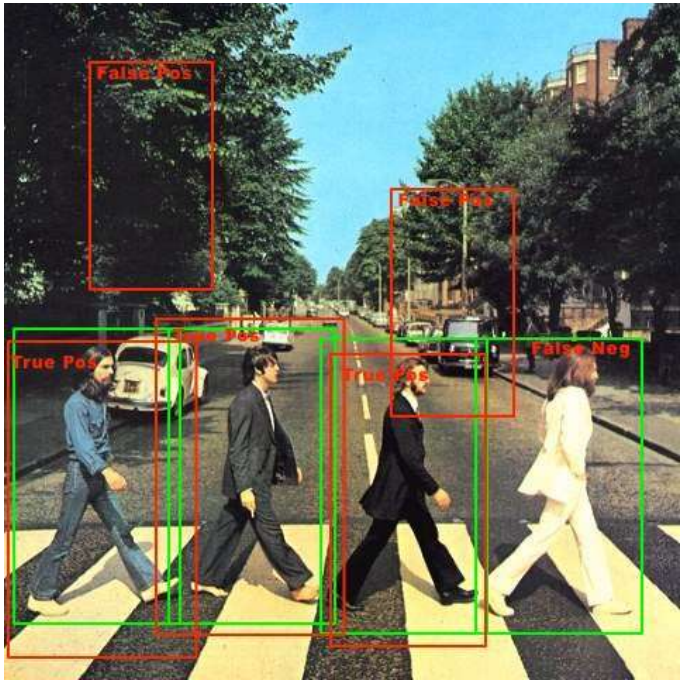
IoU = Intersection over Union

Обнаружение, если IoU > *p* (пр.: 0.5)



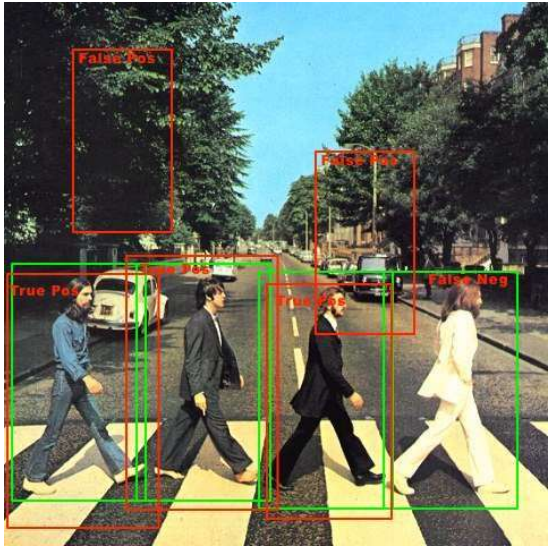
Оценка качества детектора

- Выход алгоритма: отсортированные по качеству обнаружения
- Все обнаружения оцениваются:
  - IoU > p => true positive
  - IoU < p => false positive
- Пропущенный пример => false negatives



Precision & recall

- Точность (Precision)
  - Доля истинных объектов основного класса среди всех классифицированных, как основной класс
- Полнота (Recall)
  - Доля правильно распознанных объектов основного класса среди всех объектов основного класса из тестовой выборки



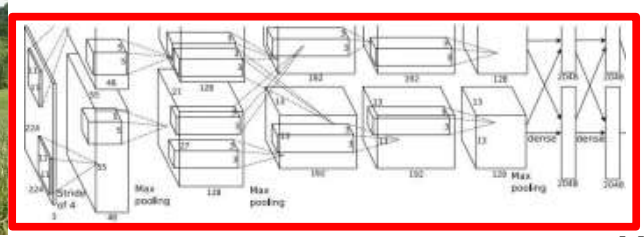
$$Precision = \frac{Number\ of\ true\ detections}{Number\ of\ detections}$$

$$Recall = \frac{Number\ of\ true\ detections}{Number\ of\ gt\ objects}$$

Детекция: один объект

(Классификация + Локализация)

This image is CC0 public domain



Часто предобучена на ImageNet (Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

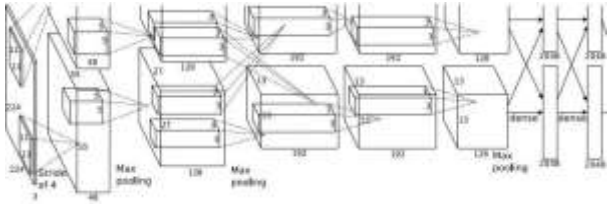
Correct box: (x', y', w', h')

Multitask Loss

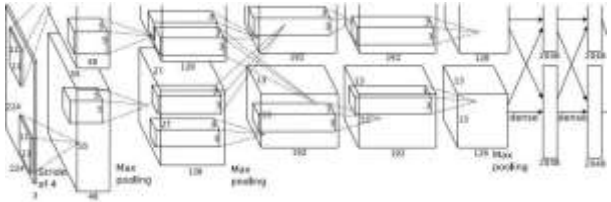
+ **Loss**

Считаем локализацию задачей регрессии

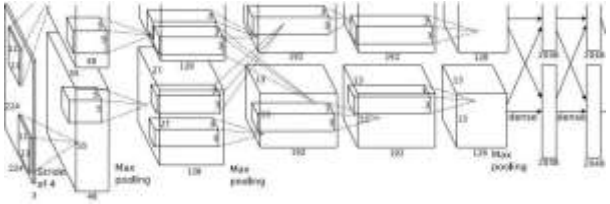
Детекция: Несколько объектов



CAT: (x, y, w, h)



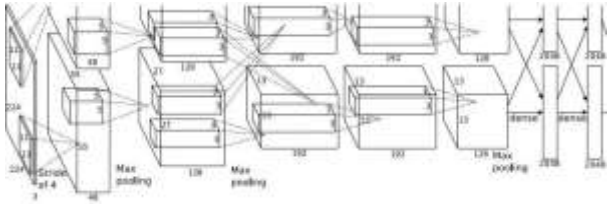
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



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

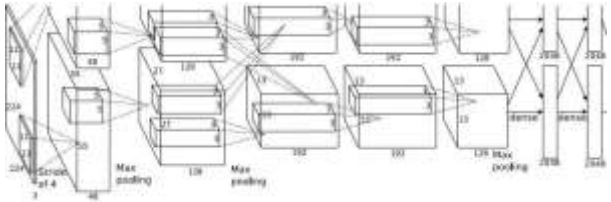
Детекция: несколько объектов

Каждая картинка имеет разное количество выводов



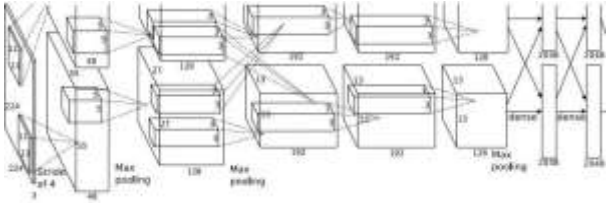
CAT: (x, y, w, h)

4 числа



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

12 чисел

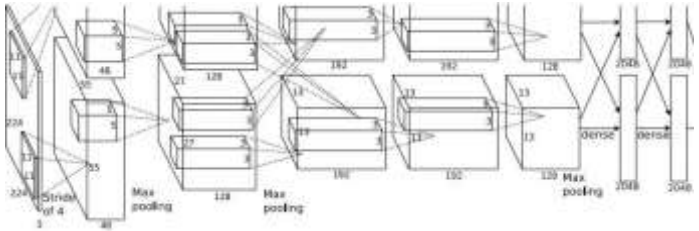


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Много чисел

Детекция: несколько объектов

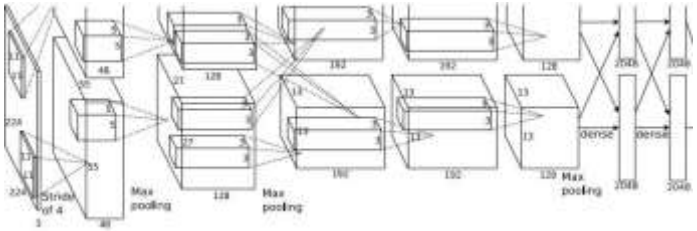
Применяем CNN ко многим различным вырезкам картинки. CNN классифицирует каждую вырезку как объект или фон.



Dog? NO
Cat? NO
Background? YES

Детекция: несколько объектов

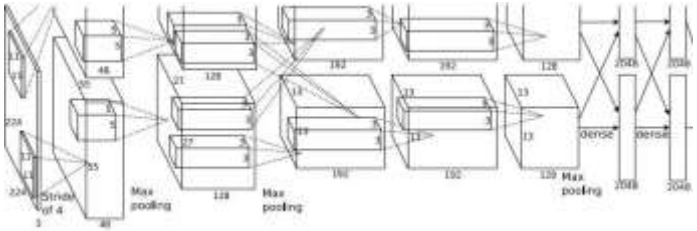
Применяем CNN ко многим различным вырезкам картинки. CNN классифицирует каждую вырезку как объект или фон.



Dog? YES
Cat? NO
Background? NO

Детекция: несколько объектов

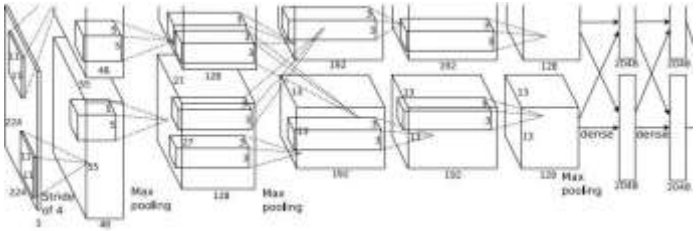
Применяем CNN ко многим различным вырезкам картинки. CNN классифицирует каждую вырезку как объект или фон.



Dog? YES
Cat? NO
Background? NO

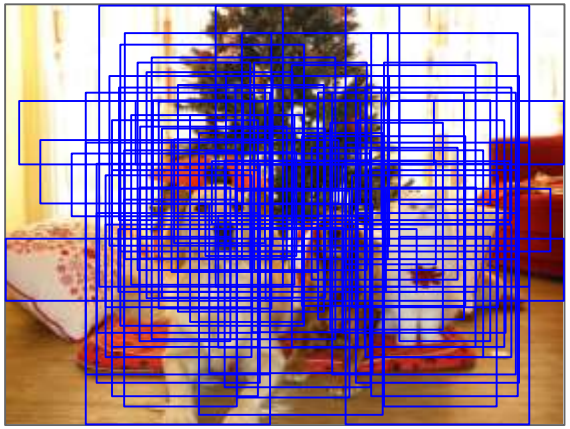
Детекция: несколько объектов

Применяем CNN ко многим различным вырезкам картинки. CNN классифицирует каждую вырезку как объект или фон.

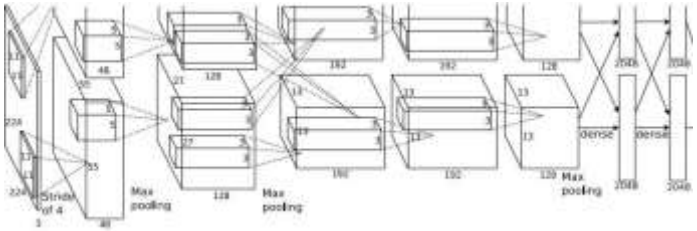


Dog? NO
Cat? YES
Background? NO

Детекция: несколько объектов



Применяем CNN ко многим различным вырезкам картинки. CNN классифицирует каждую вырезку как объект или фон.

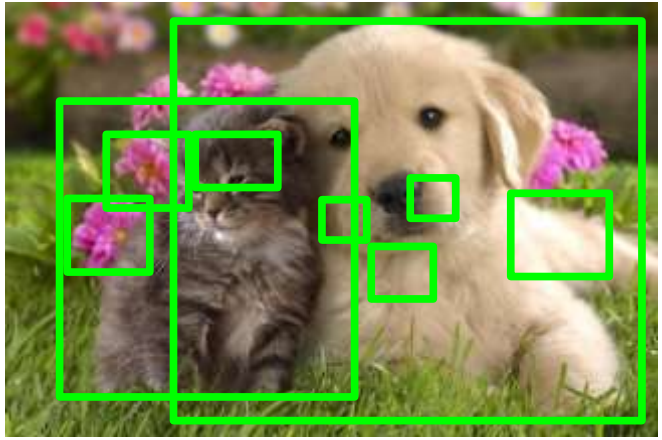


Dog? NO
Cat? YES
Background? NO

Проблема: Необходимо применить CNN к огромному количеству мест, масштабов и интервалов, что очень затратно.

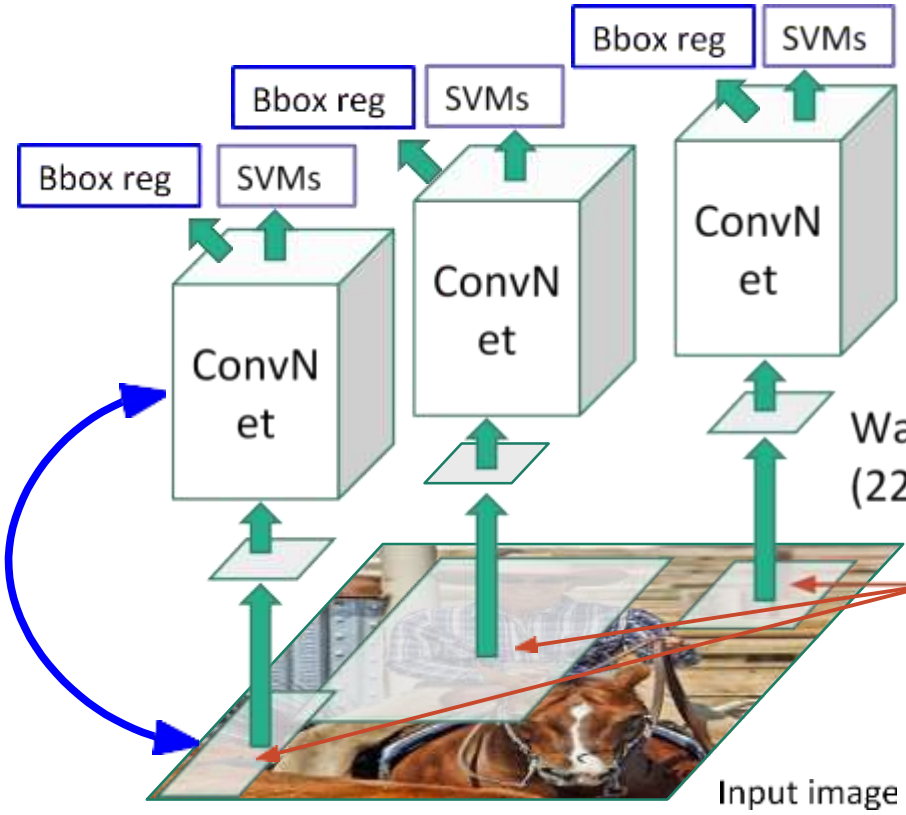
Предполагаемые области: Selective Search

- Найти области изображения, которые похожи на объекты
- Относительно быстро; Выборочный поиск дает 2000 предполагаемых областей за несколько секунд на CPU



"Slow" R-CNN

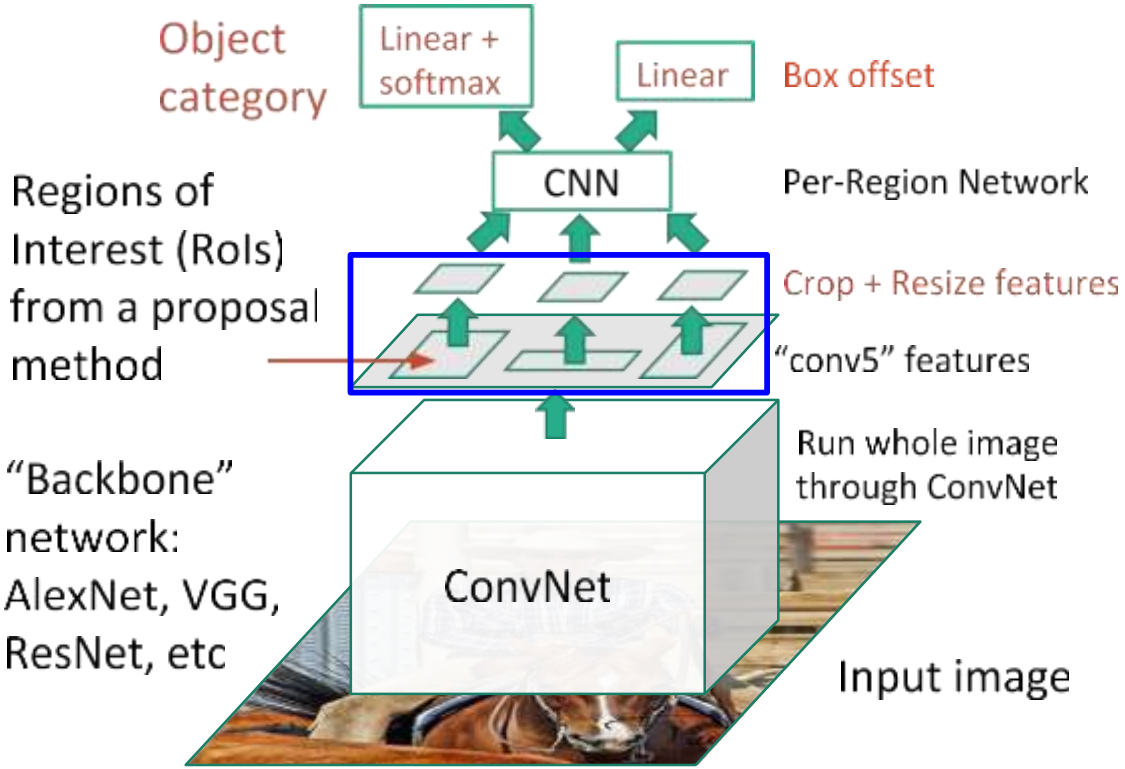
Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



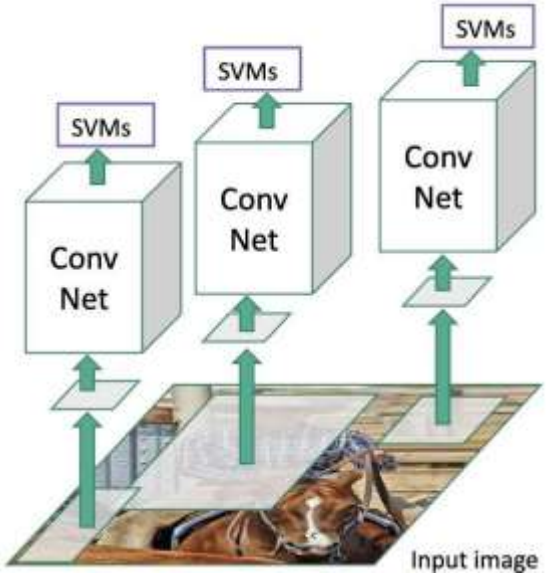
Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Process image before cropping! Swap convolution and cropping!

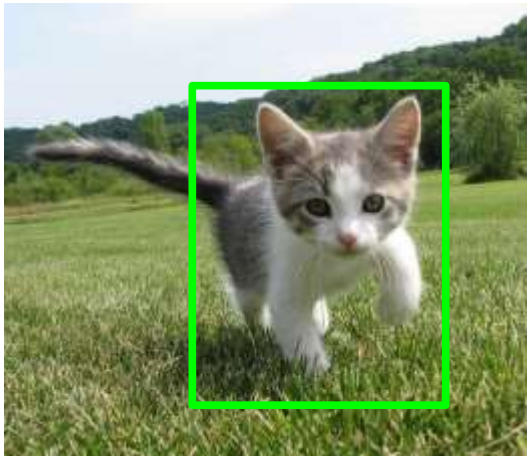
Fast R-CNN



"Slow" R-CNN



Cropping Features: RoI Pool



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

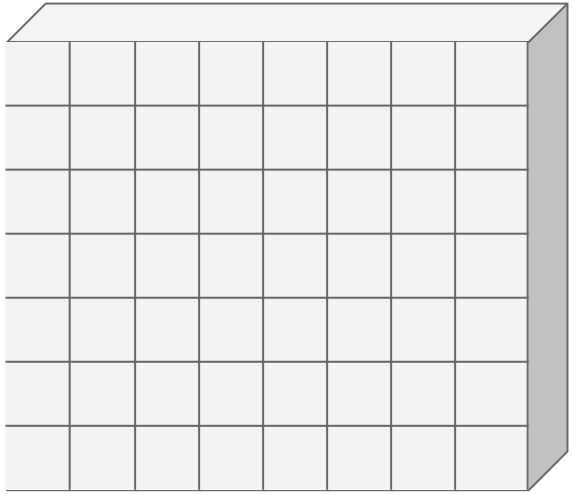
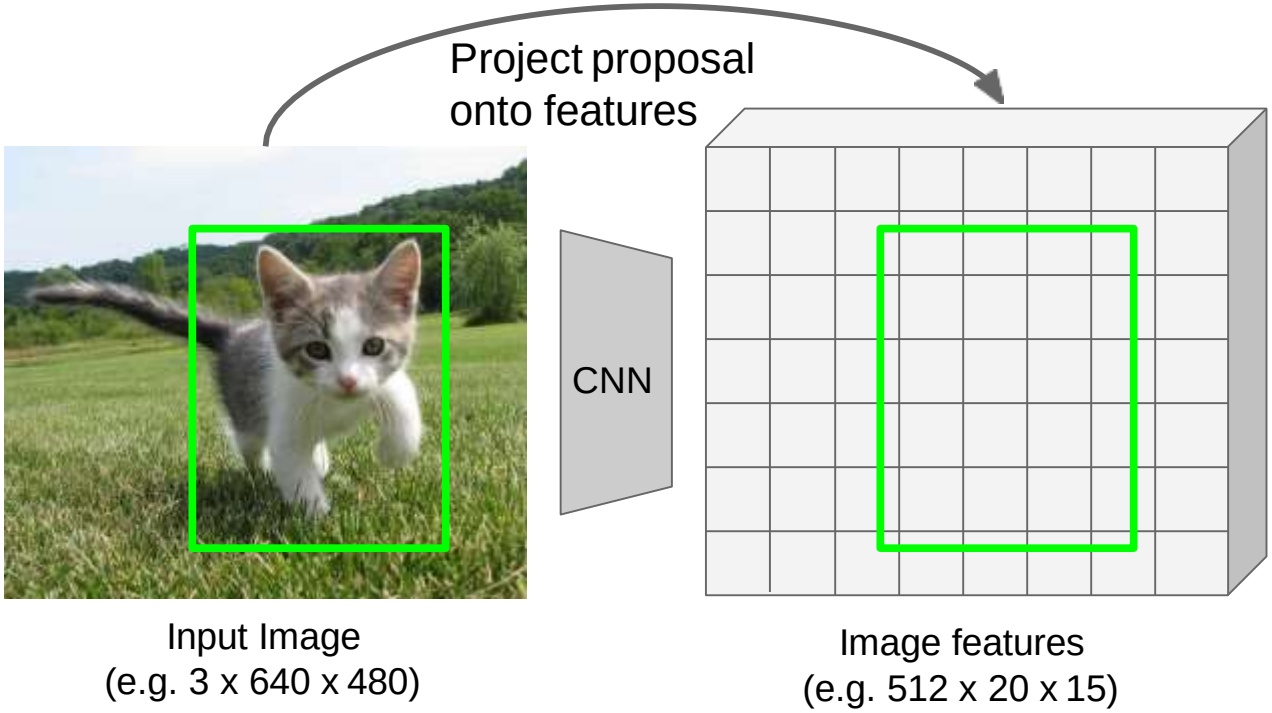
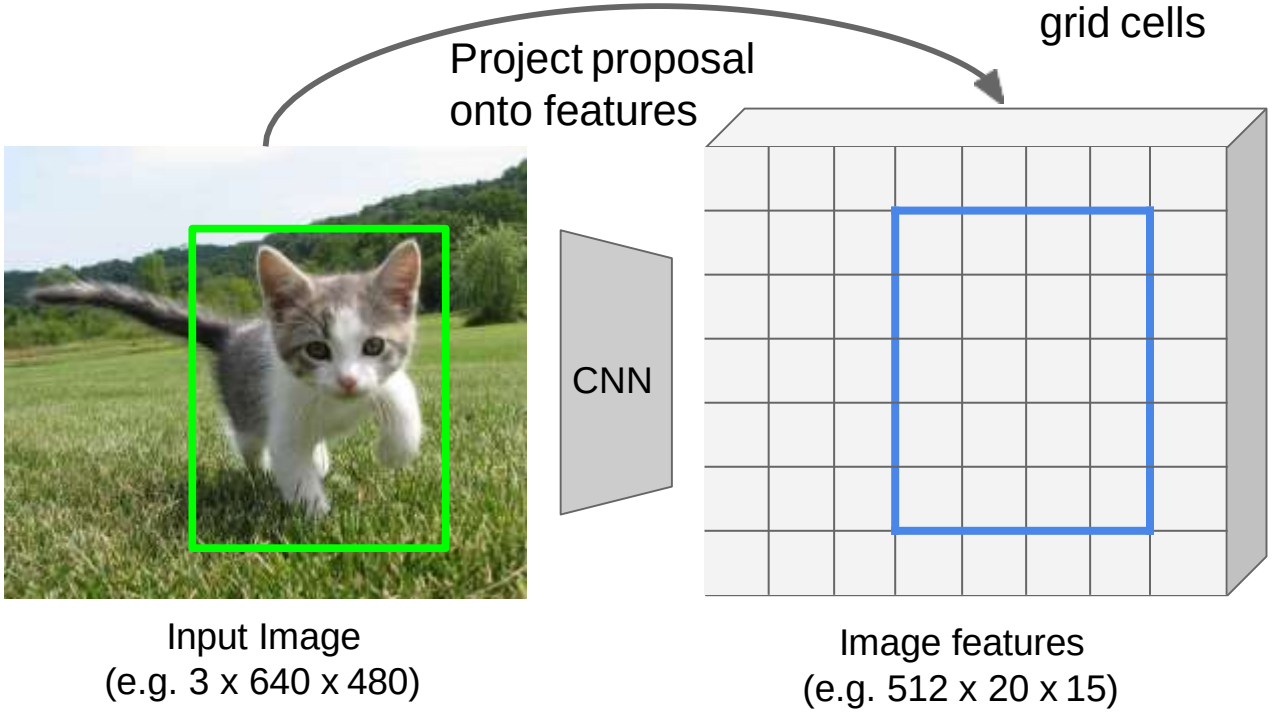


Image features
(e.g. 512 x 20 x 15)

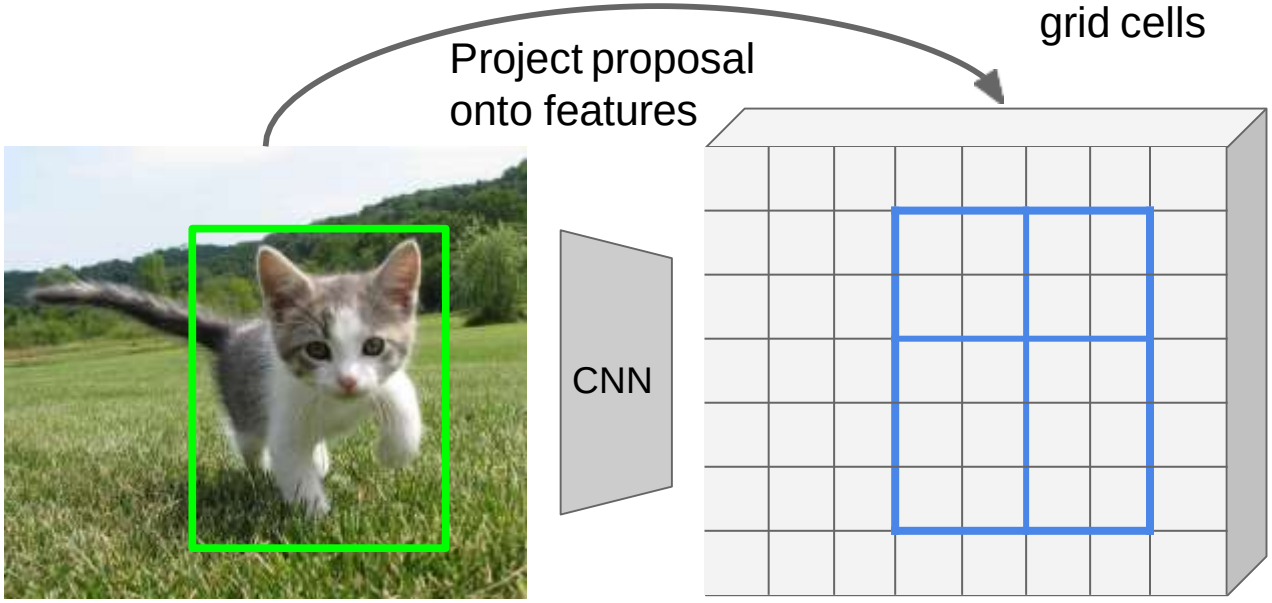
Cropping Features: RoI Pool



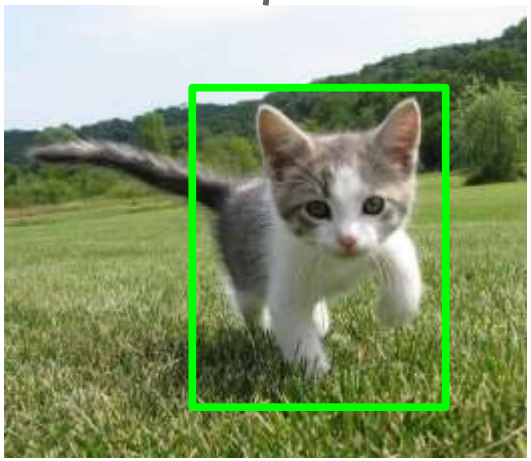
Cropping Features: RoI Pool



Cropping Features: RoI Pool



Cropping Features: RoI Pool



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

Project proposal onto features

CNN

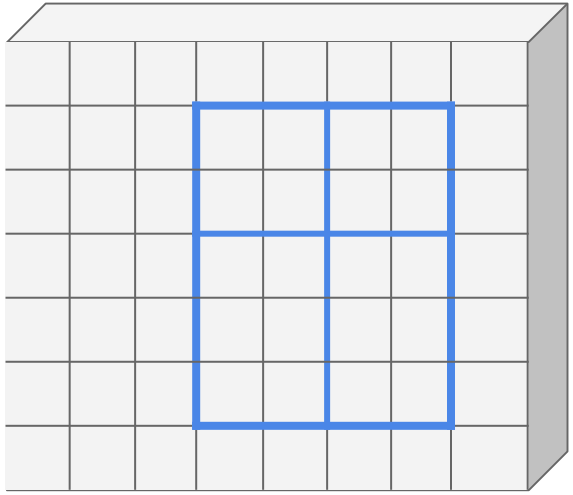
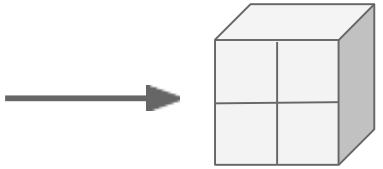


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

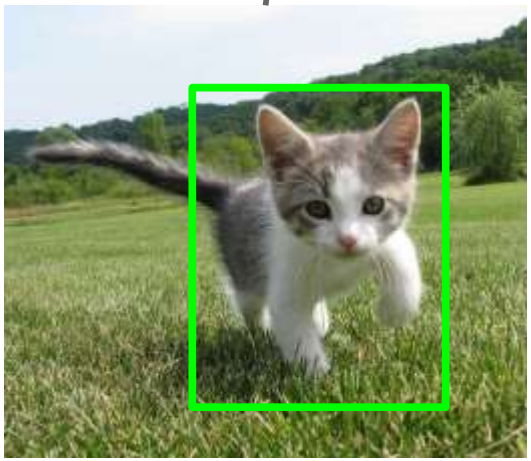
Max-pool within each subregion



Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Cropping Features: RoI Pool



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

Project proposal onto features

CNN

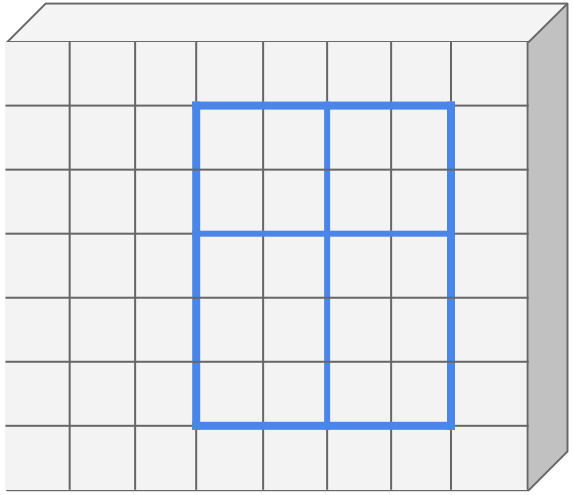
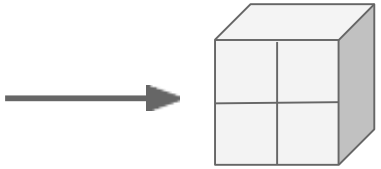


Image features
(e.g. 512 x 20 x 15)

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

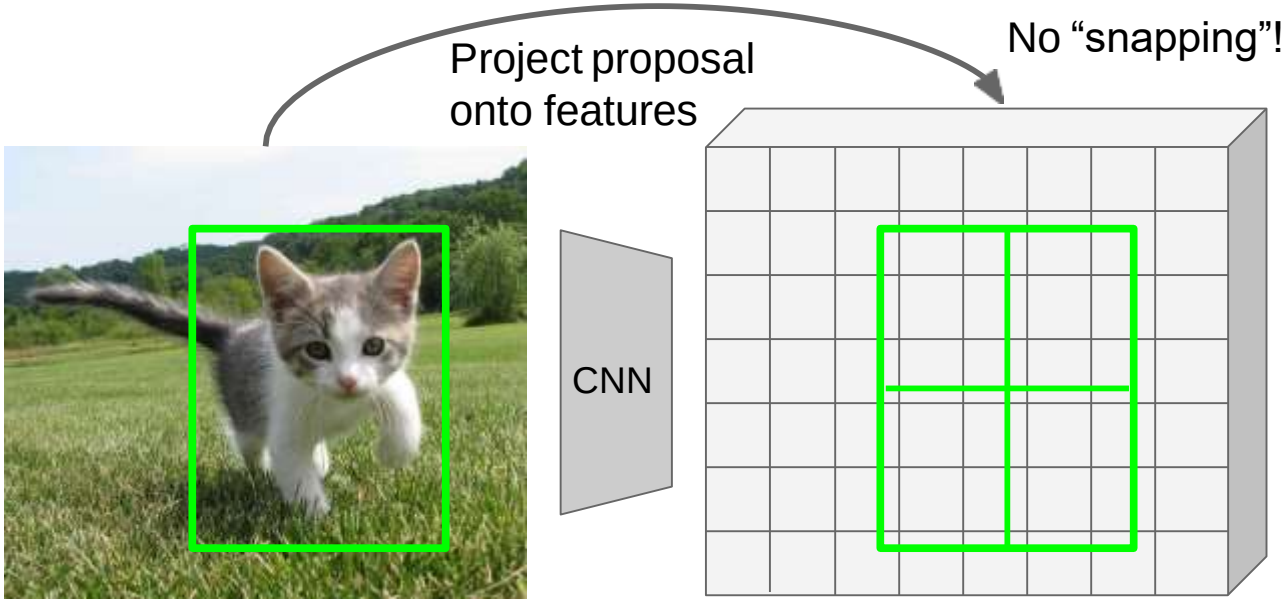


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

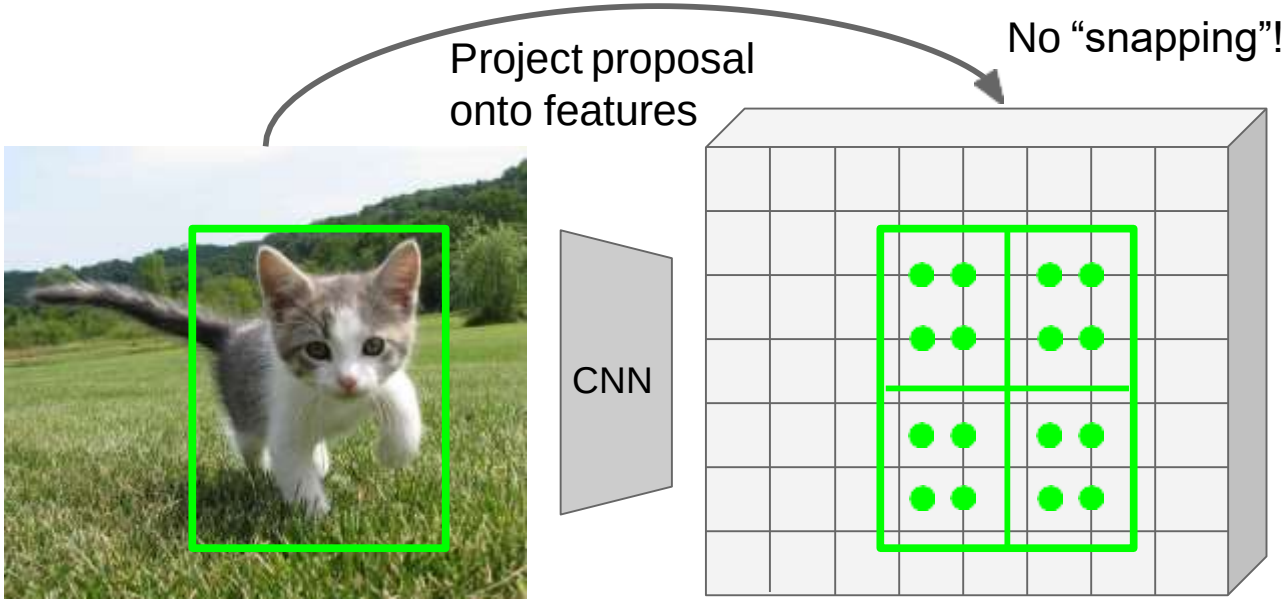
Problem: Region features slightly misaligned

Cropping Features: RoI Align

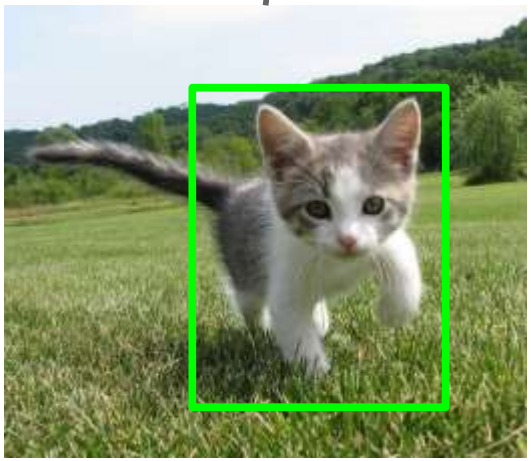


Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



Cropping Features: RoI Align



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

Project proposal onto features

No “snapping”!

CNN

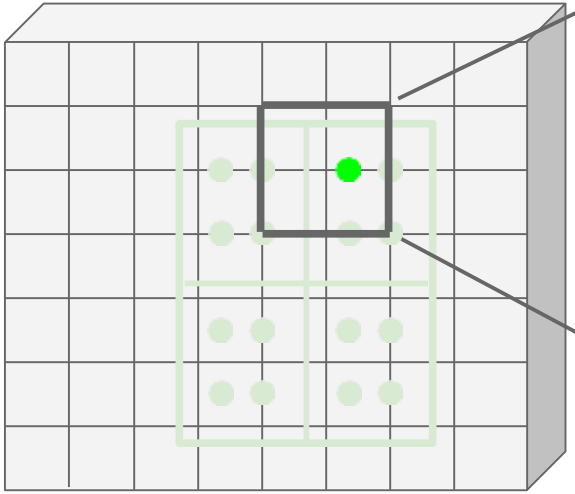
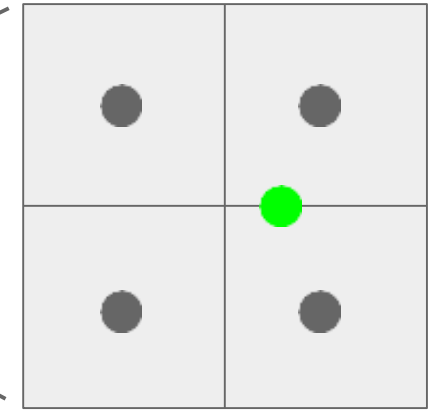


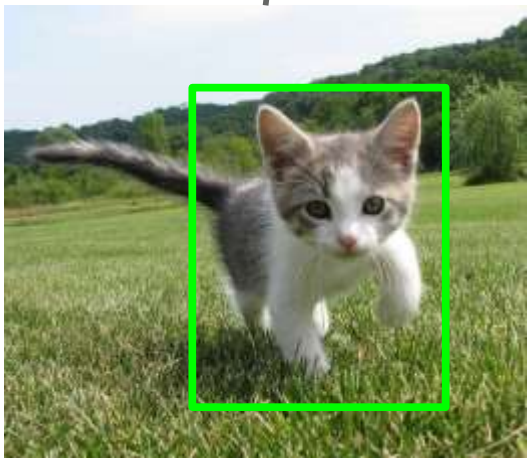
Image features
(e.g. 512 x 20 x 15)

Sample at regular points in each subregion using bilinear interpolation



Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

Cropping Features: RoI Align



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

Project proposal onto features

No “snapping”!

CNN

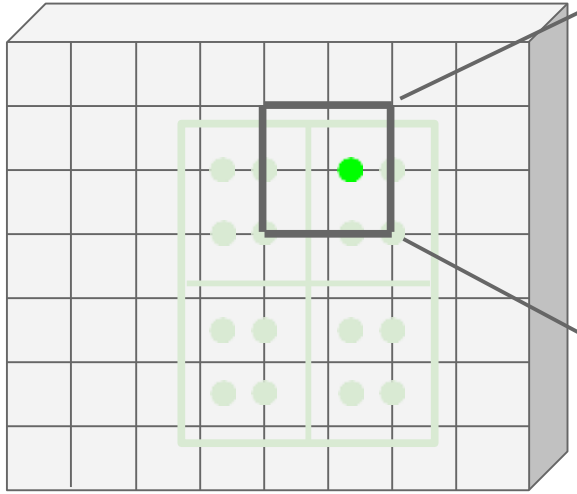
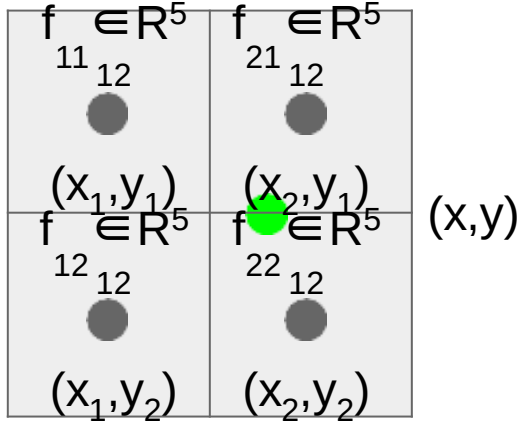


Image features
(e.g. 512 x 20 x 15)

Sample at regular points in each subregion using bilinear interpolation

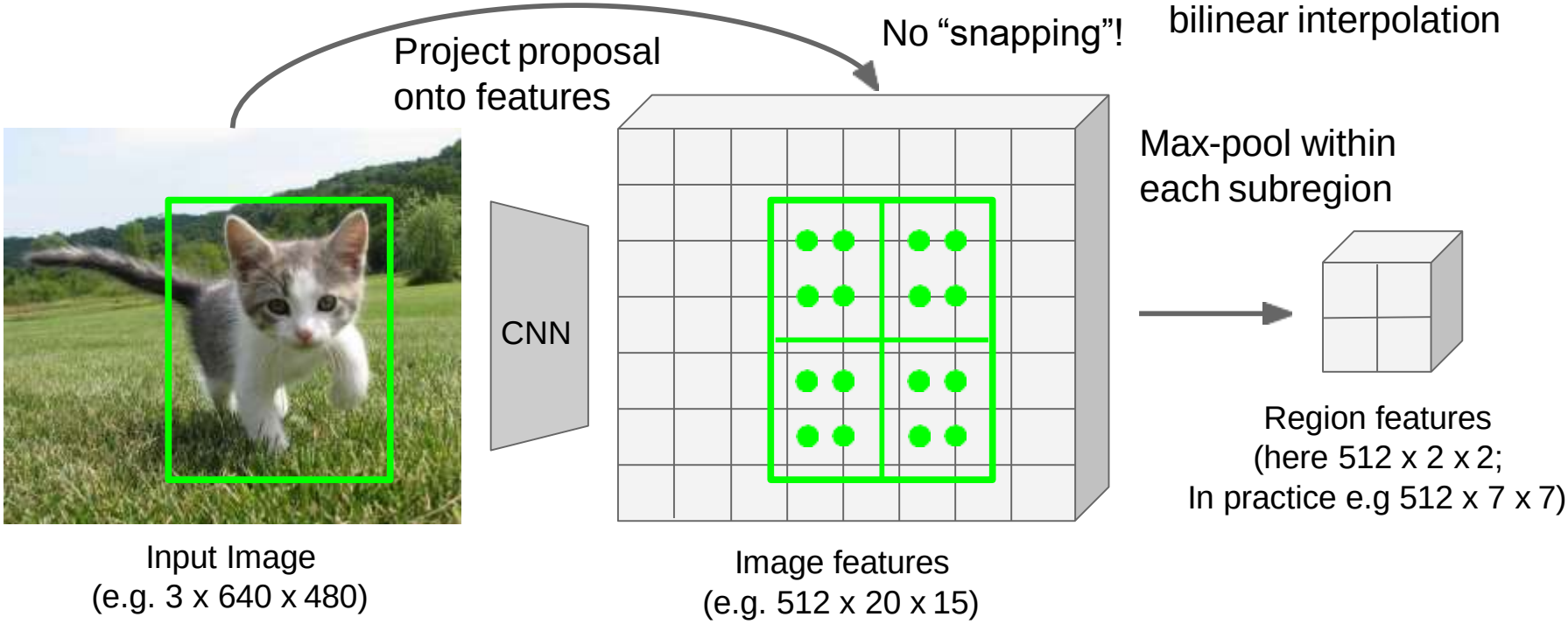


Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

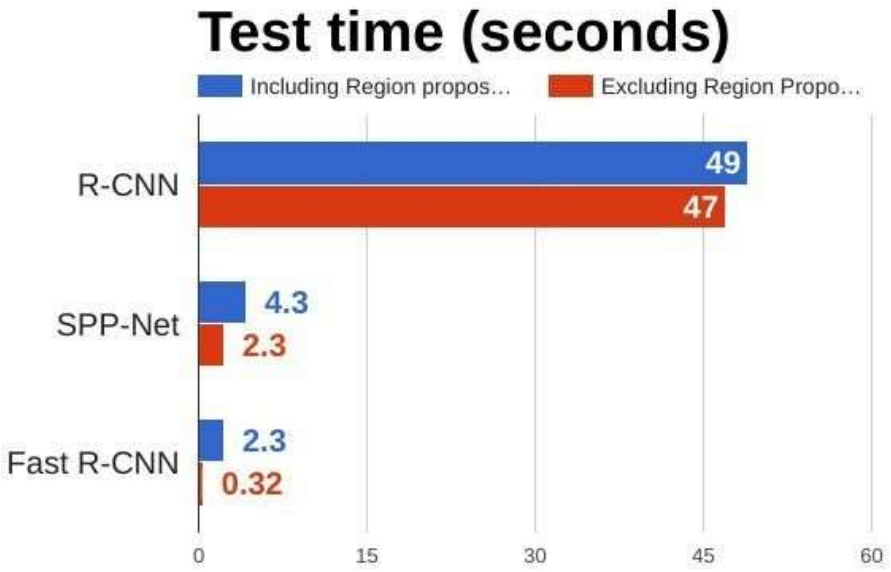
$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



R-CNN vs Fast R-CNN

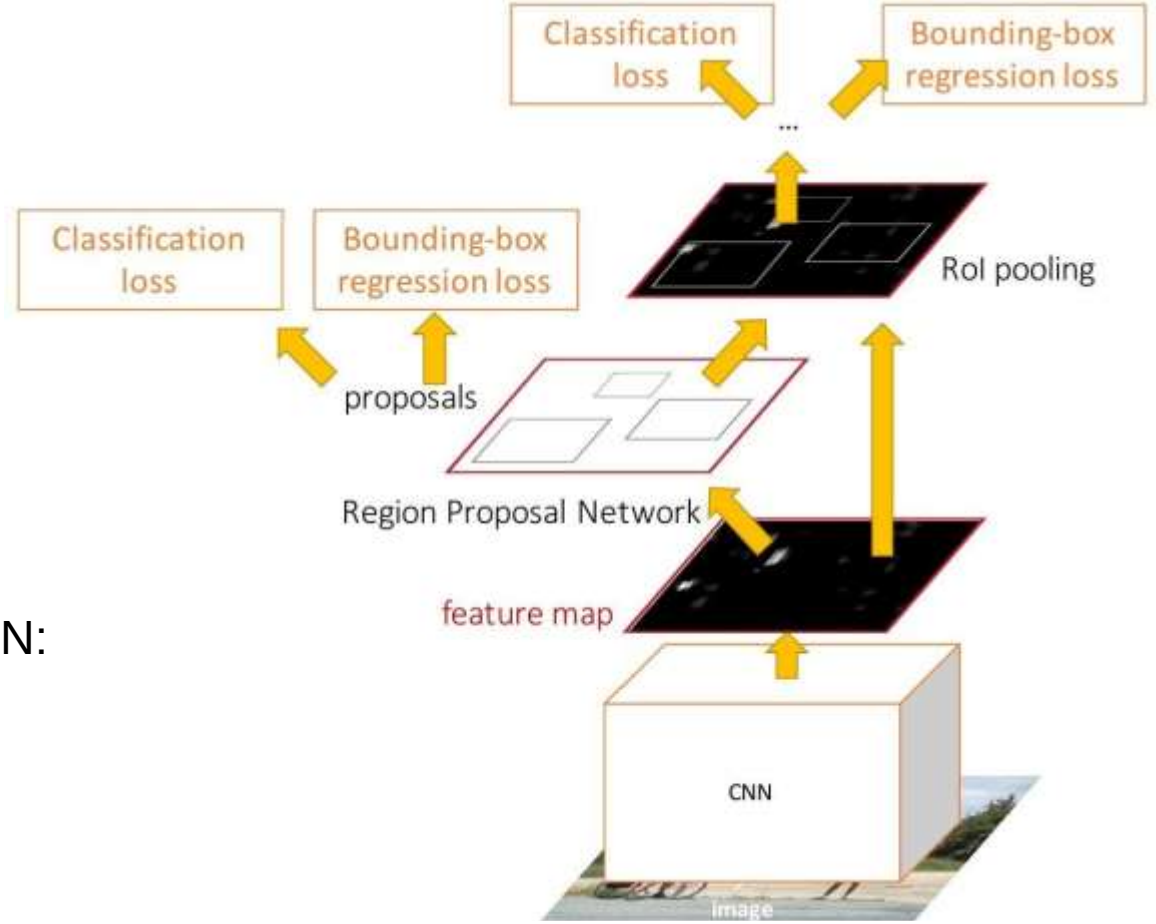


Faster R-CNN:

Make CNN do proposals!

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

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



Region Proposal Network

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

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

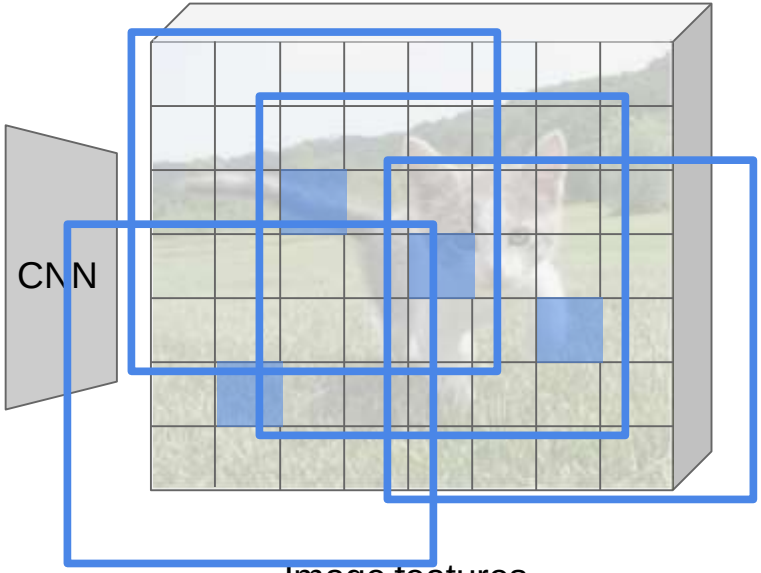


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

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

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

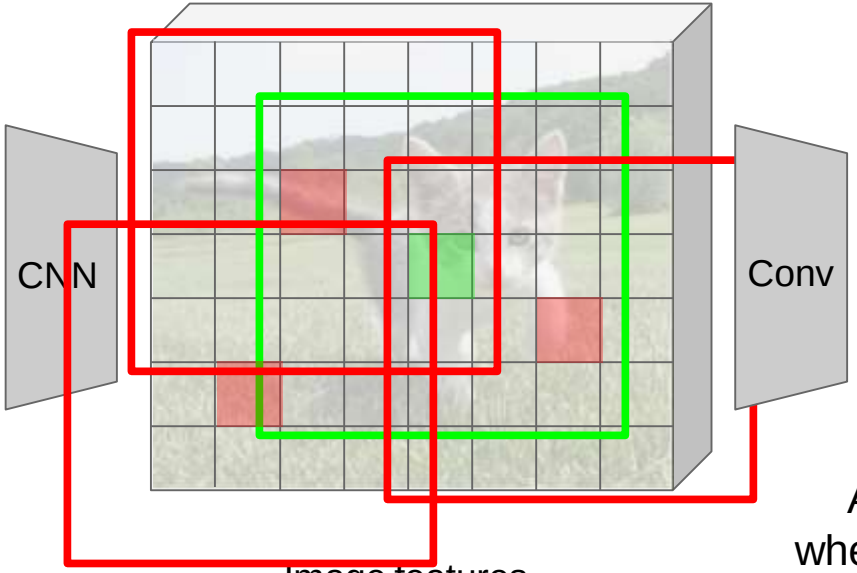


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression)

Region Proposal Network

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

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

CNN

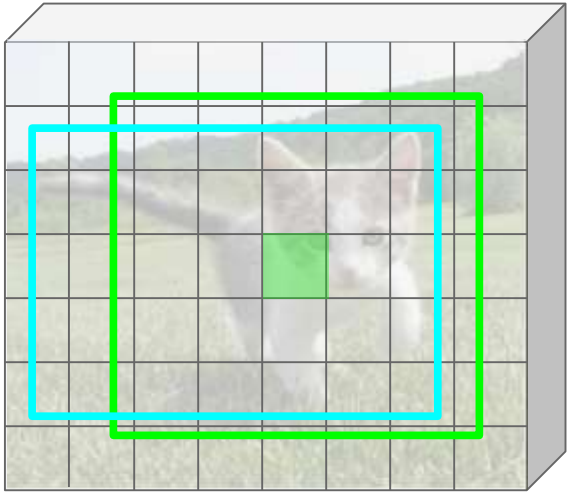


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

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

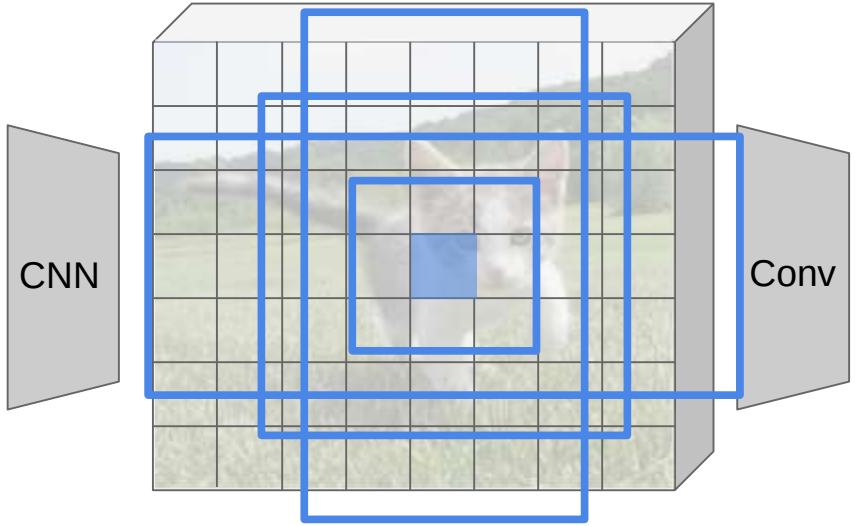


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

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

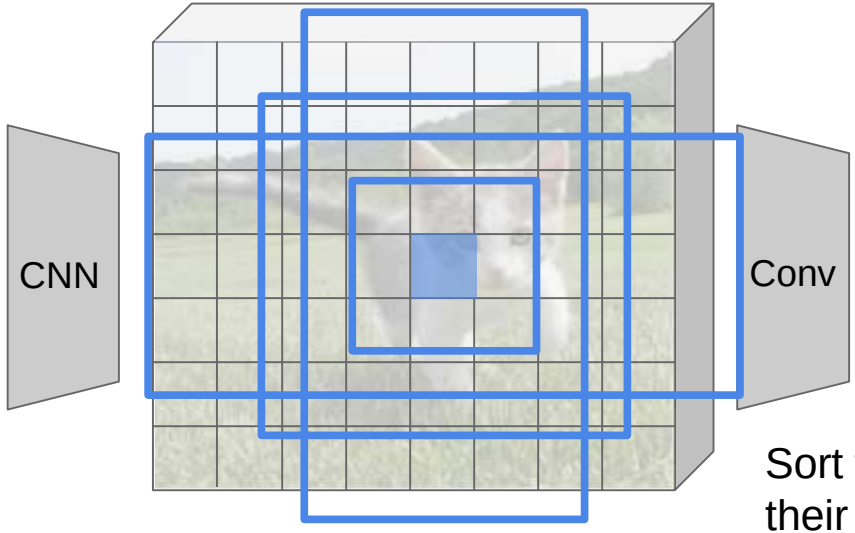


Image features
(e.g. 512 x 20 x 15)

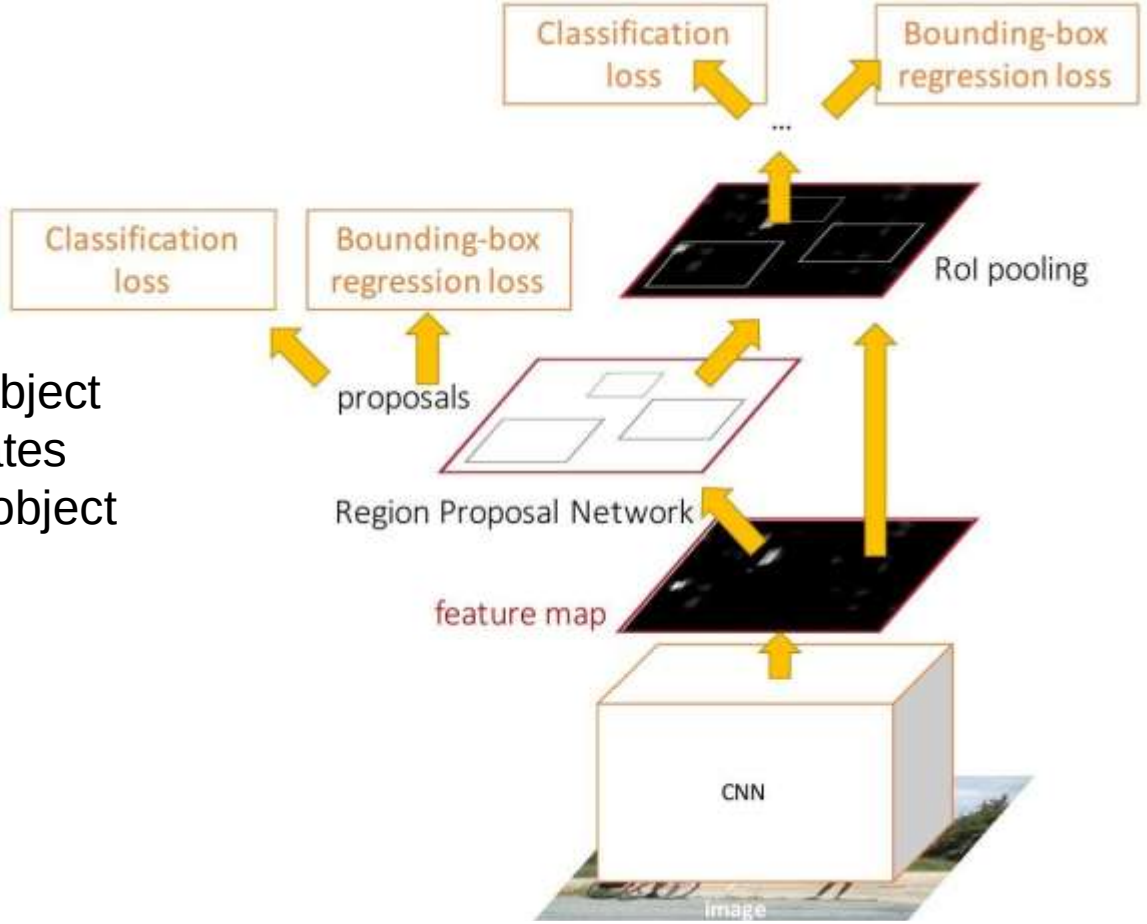
Sort the K*20*15 boxes by their “object” score, take top ~300 as our proposals

Faster R-CNN:

Make CNN do proposals!

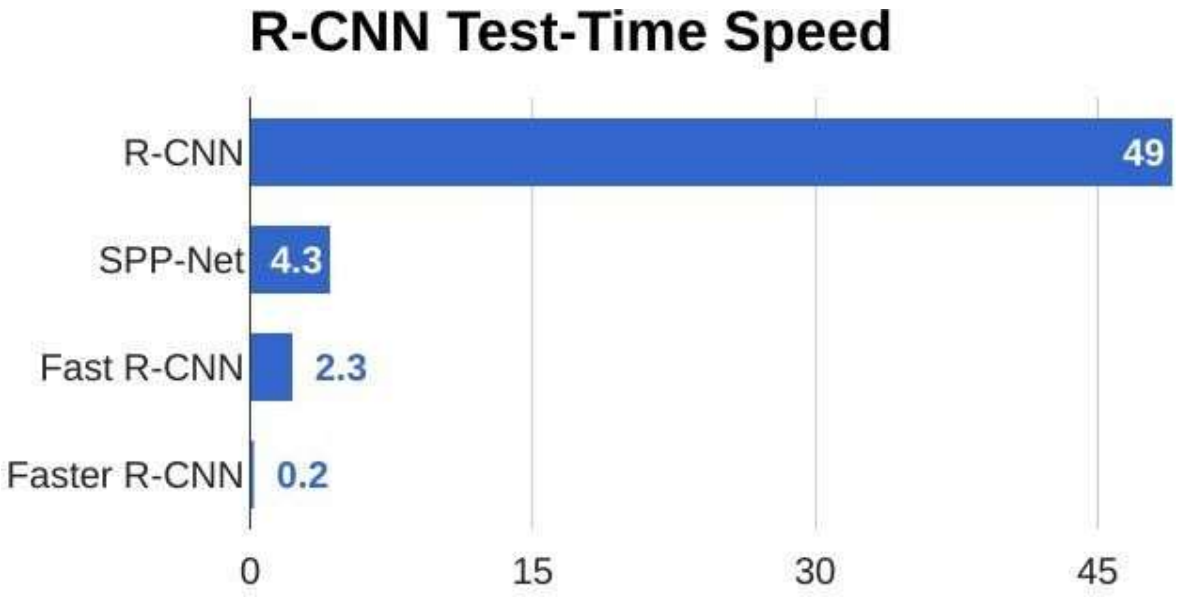
Jointly train with 4 losses:

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



Faster R-CNN:

Make CNN do proposals!



Instance Segmentation

Classification

Semantic Segmentation

Object Detection

Instance Segmentation

CAT

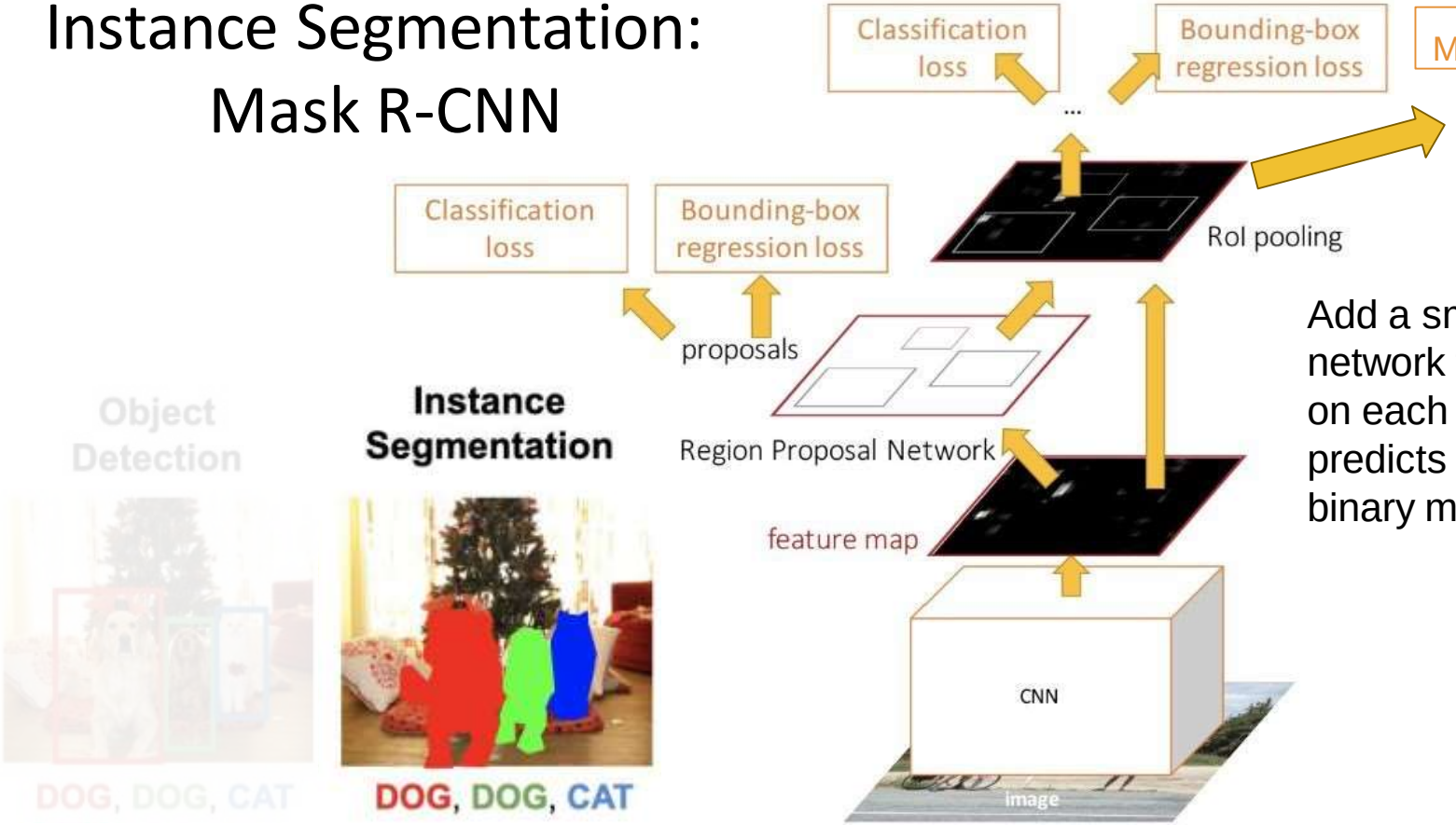
DOG, DOG, CAT

DOG, DOG, CAT

Multiple Object

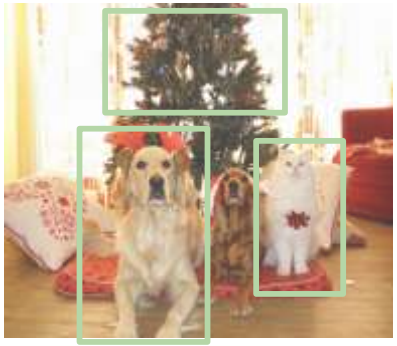
Object Detection: Faster R-CNN

Instance Segmentation: Mask R-CNN

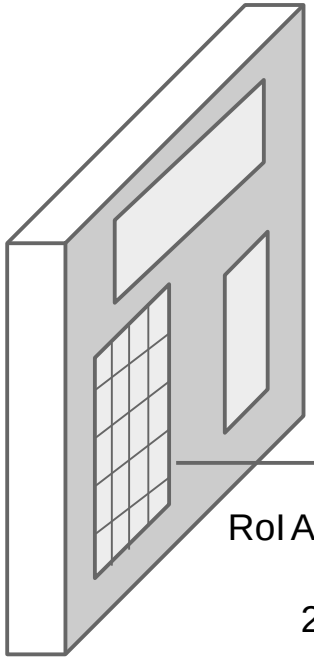


Add a small mask network that operates on each RoI and predicts a 28x28 binary mask

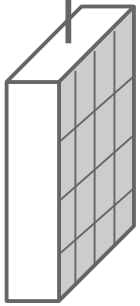
Mask R-CNN



CNN
+RPN

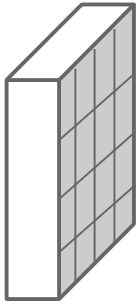


RoI Align



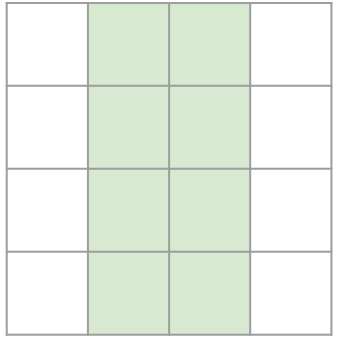
256 x 14 x 14

Conv



256 x 14 x 14

Conv

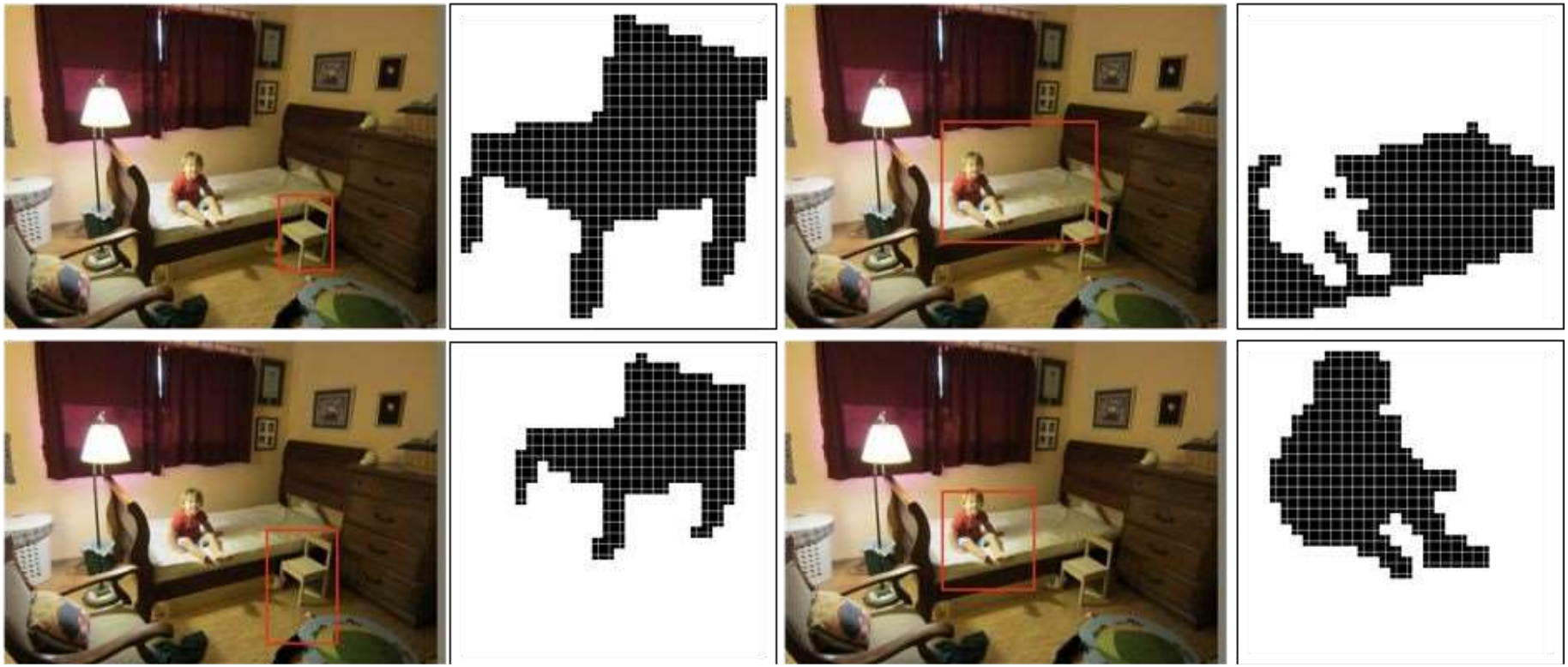


Predict a mask
for each of C
classes

C x 28 x 28

Classification Scores: C
Box coordinates (per class): 4 *
C

Mask R-CNN: Example Mask Training Targets



Вопросы

Вопрос 1: Какие проблемы встречаются в процессе анализа изображений?

Вопрос 2: Оценка качества детектора (Критерий обнаружения/score, полнота и точность - расписать)

Вопрос 3: Показать схематично, как решается задача instance segmentation (т.е. указать последовательность действий)