

Основы глубинного обучения

Лекция 6

Задачи компьютерного зрения

Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2020

Интерпретация моделей

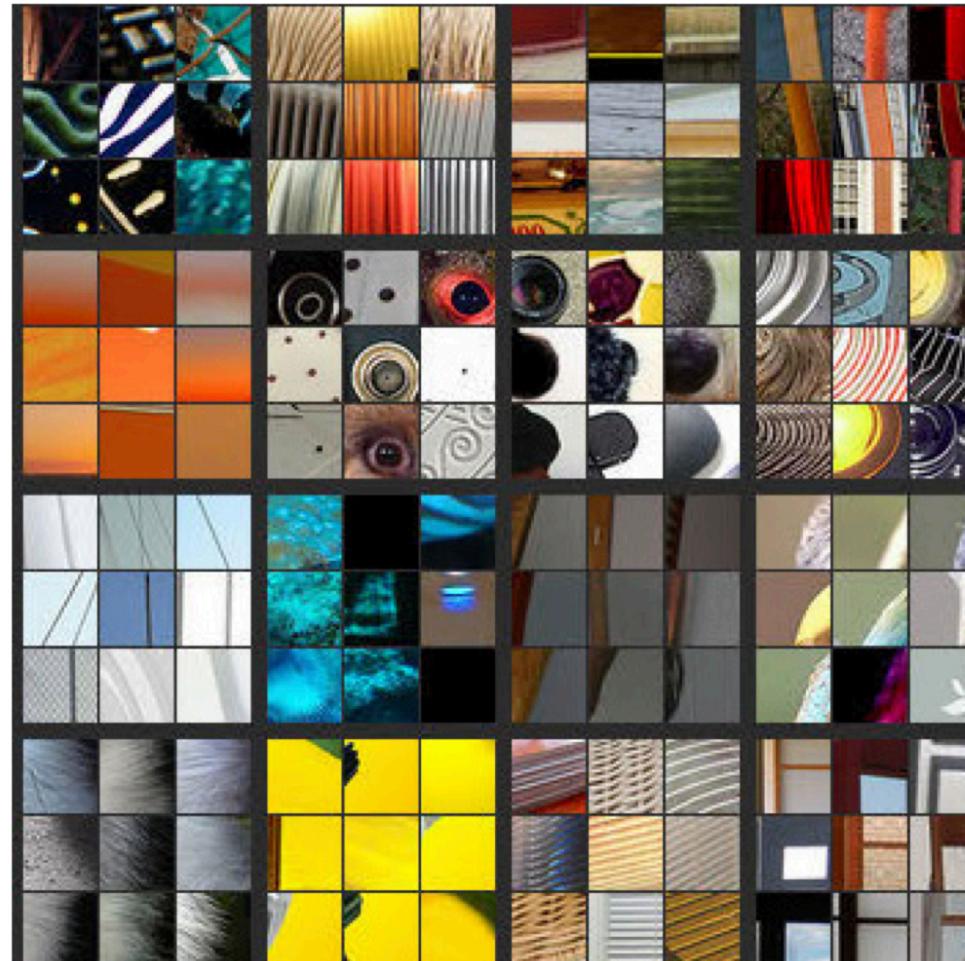
Что находят свёрточные сети?

- Можно для каждого фильтра найти кусочки картинок, дающие самый сильный отклик
- Сделаем это для AlexNet

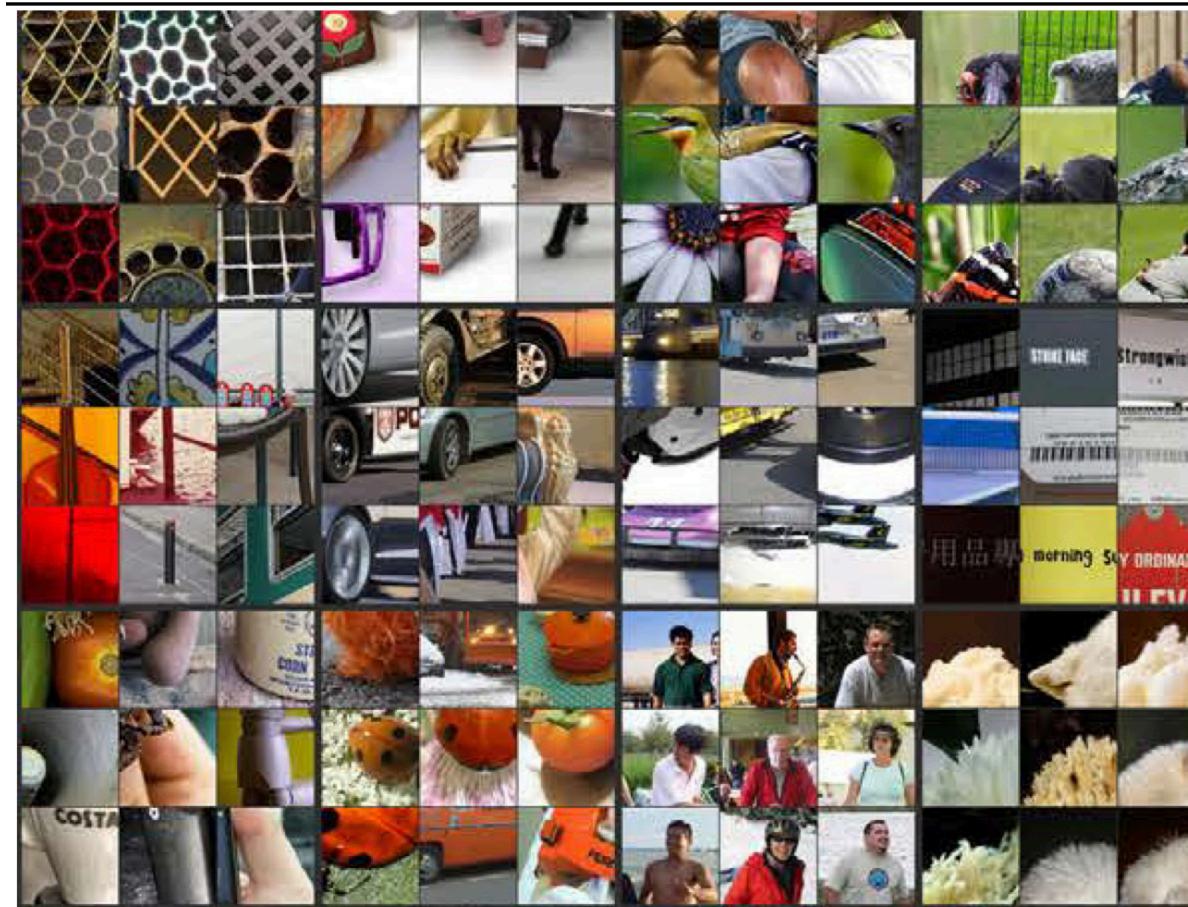
Слой 1



Слой 2



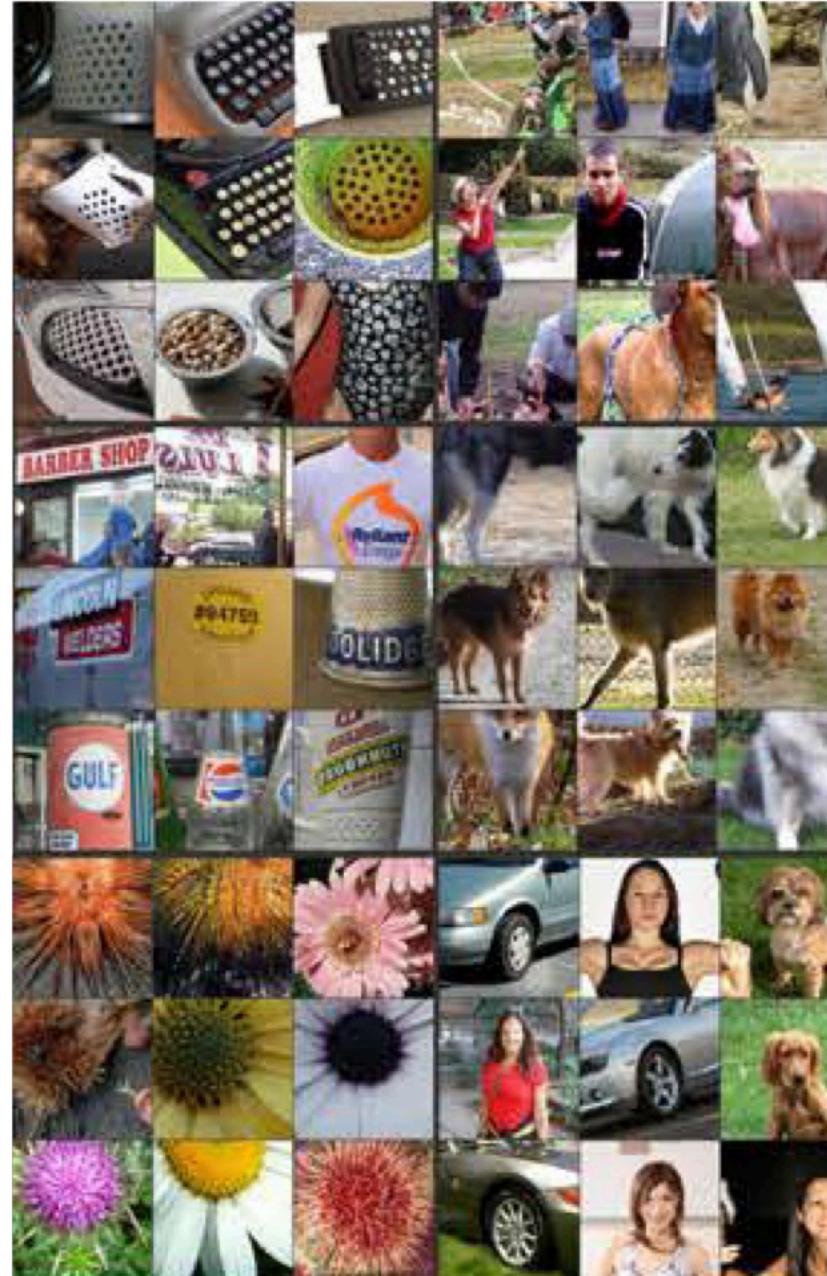
Слой 3



Слой 4



Слой 5

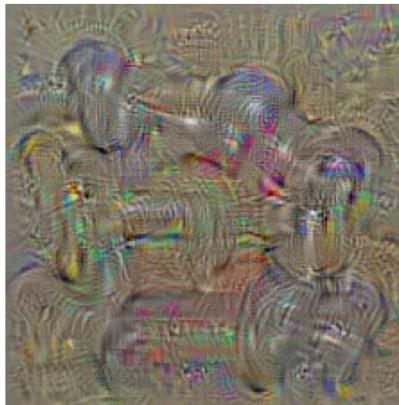


Максимизация вероятности класса

$$a_y(x) - \lambda \|x\|_2^2 \rightarrow \max_x$$

- Инициализируем картинку случайным шумом
- Ищем оптимальную для данного класса картинку градиентным спуском

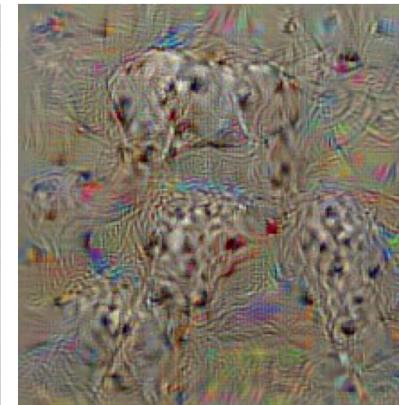
Максимизация вероятности класса



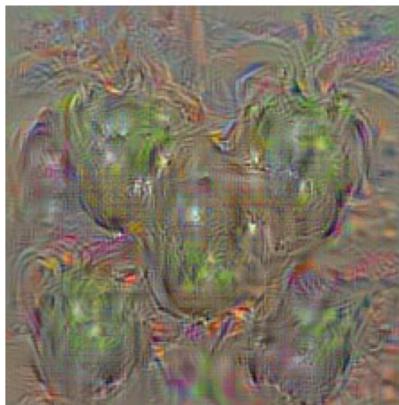
dumbbell



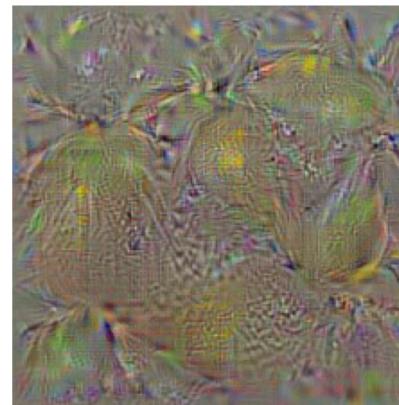
cup



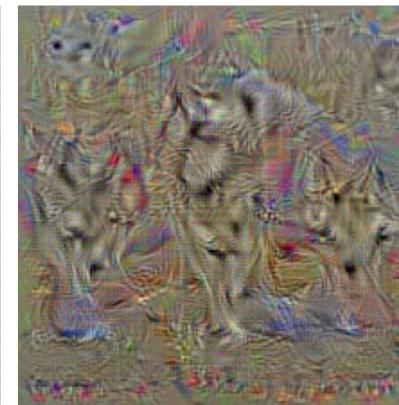
dalmatian



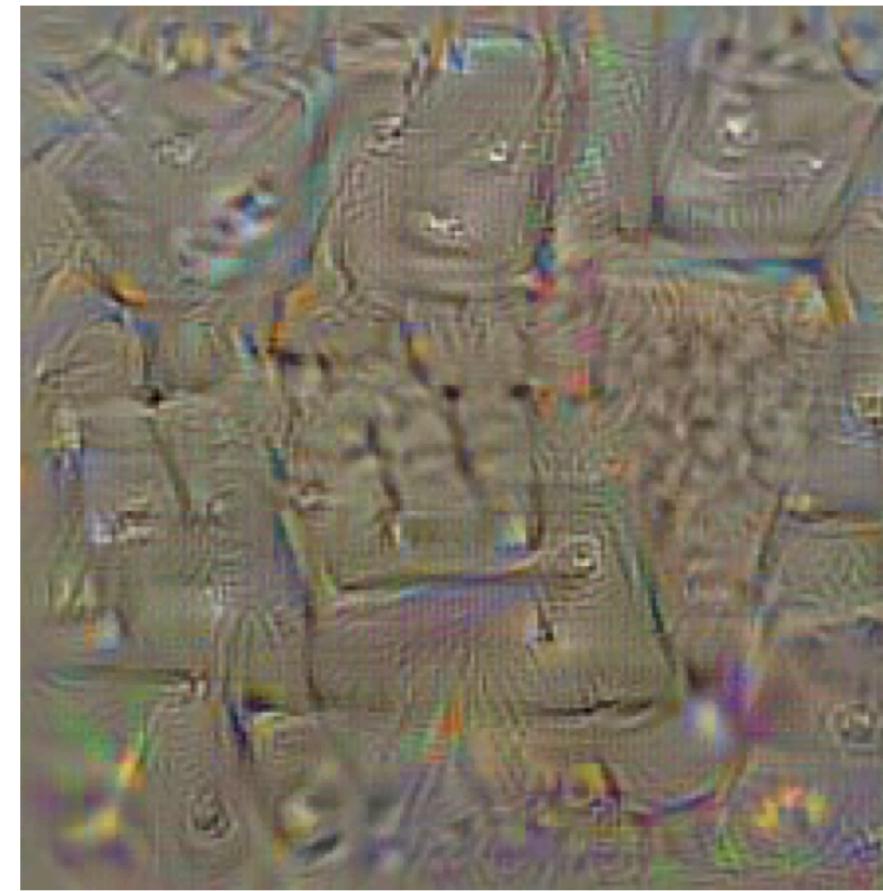
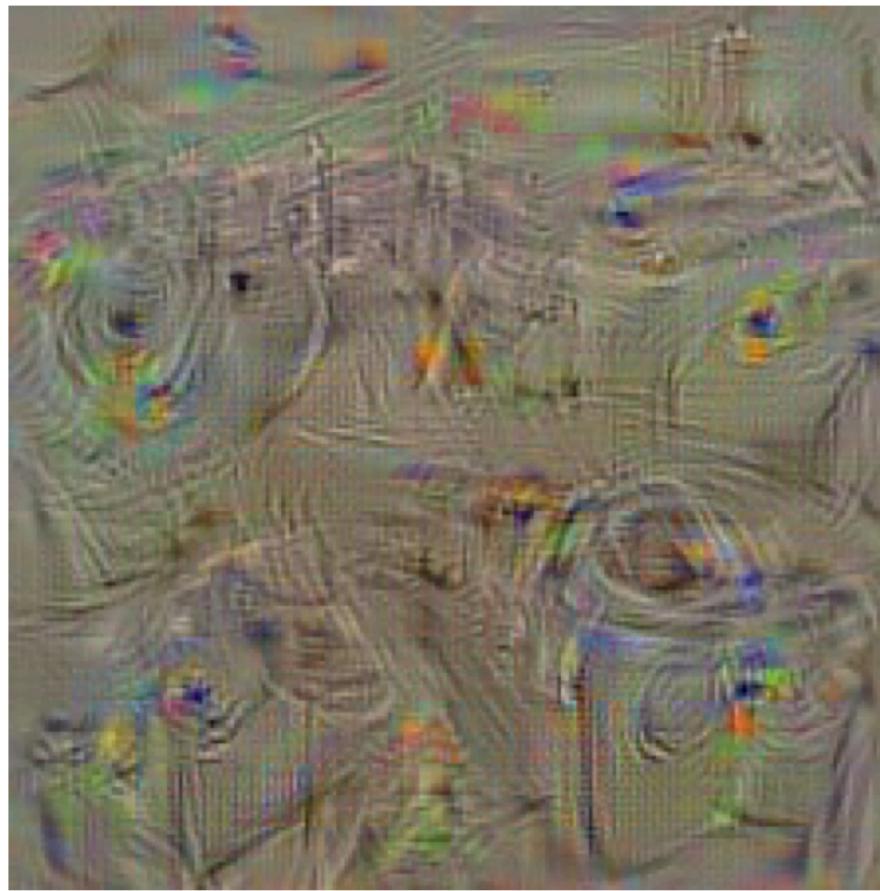
bell pepper



lemon



husky



Максимизация вероятности класса



Hartebeest



Measuring Cup



Ant



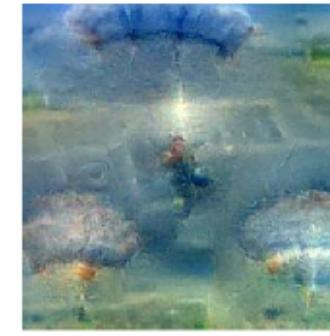
Starfish



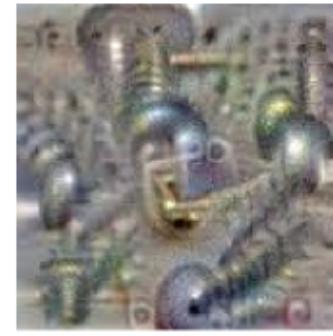
Anemone Fish



Banana

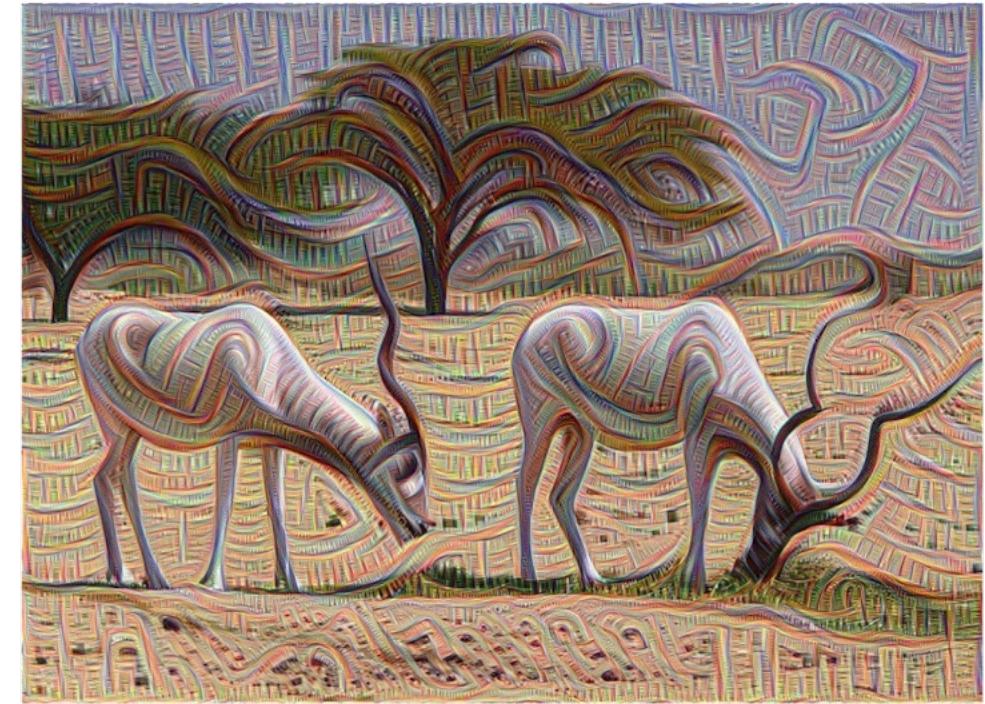


Parachute

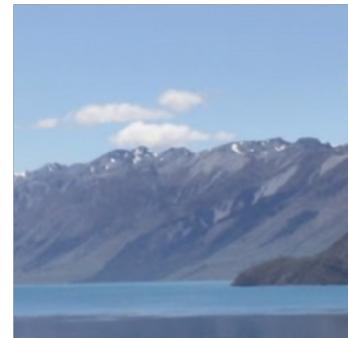


Screw

Максимизация вероятности класса



Максимизация вероятности класса



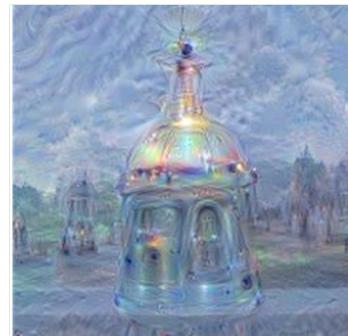
Horizon



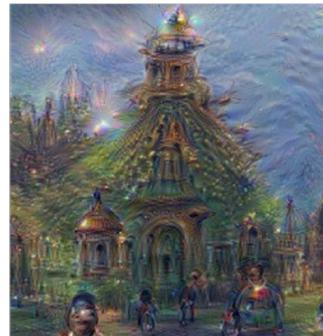
Trees



Leaves



Towers & Pagodas

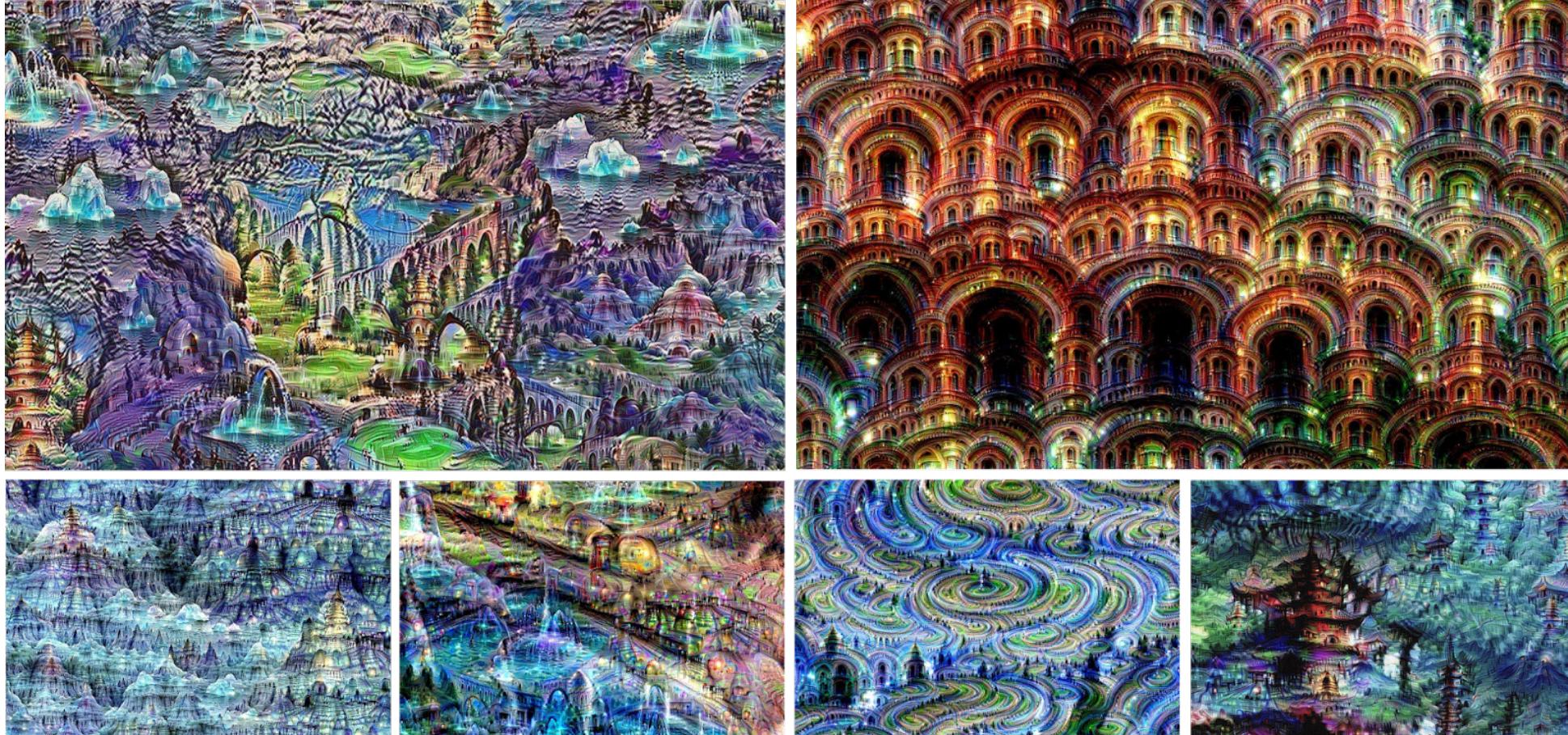


Buildings



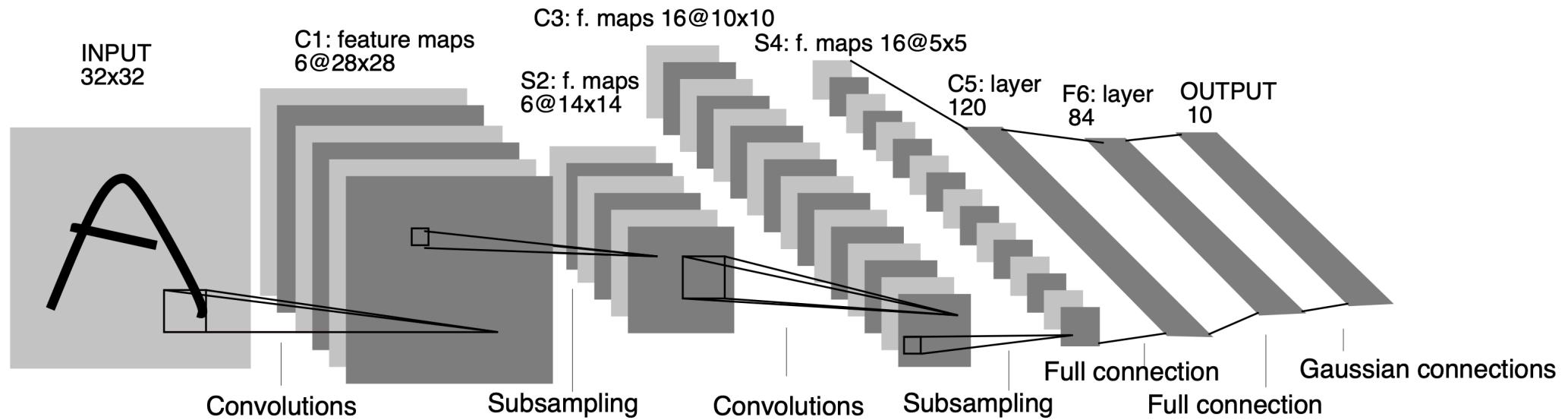
Birds & Insects

Максимизация вероятности класса



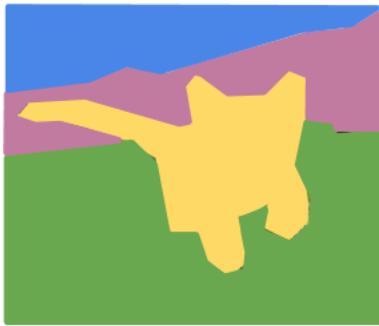
Компьютерное зрение

Классификация изображений



Обычно хочется другого

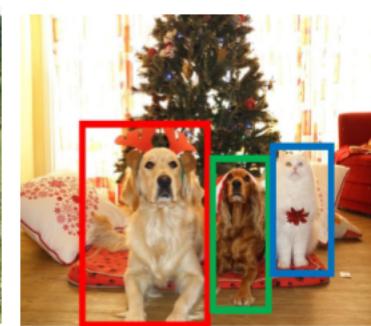
Semantic
Segmentation



Classification
+ Localization



Object
Detection



Instance
Segmentation

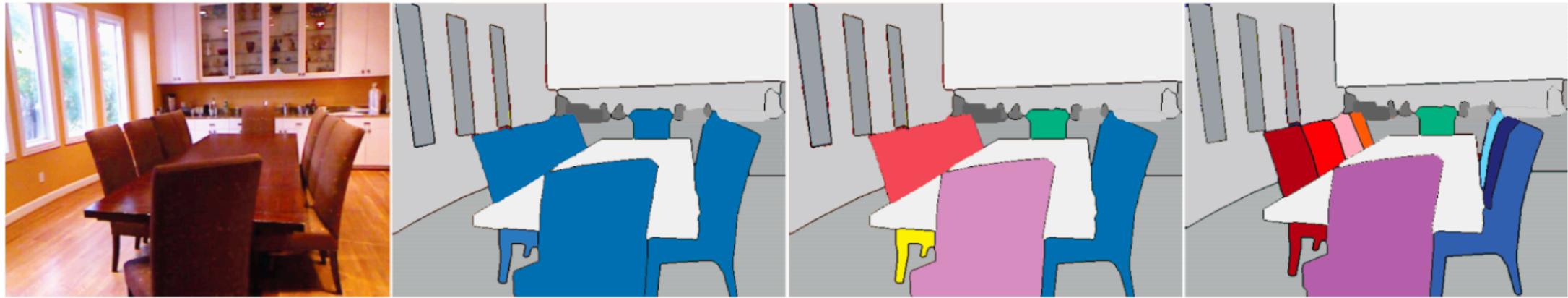


Обычно хочется другого

Но мы будем сводить все задачи к классификации!

Семантическая сегментация

Терминология



Semantic
segmentation

Instance
segmentation

Постановка задачи

- Данные: изображения и их корректные сегментации
- Пример: PASCAL VOC 2012



Постановка задачи

Table 2: Statistics of the segmentation image sets.

	train		val		trainval		test	
	img	obj	img	obj	img	obj	img	obj
Aeroplane	88	108	90	110	178	218	—	—
Bicycle	65	94	79	103	144	197	—	—
Bird	105	137	103	140	208	277	—	—
Boat	78	124	72	108	150	232	—	—
Bottle	87	195	96	162	183	357	—	—
Bus	78	121	74	116	152	237	—	—
Car	128	209	127	249	255	458	—	—
Cat	131	154	119	132	250	286	—	—
Chair	148	303	123	245	271	548	—	—
Cow	64	152	71	132	135	284	—	—
Diningtable	82	86	75	82	157	168	—	—
Dog	121	149	128	150	249	299	—	—
Horse	68	100	79	104	147	204	—	—
Motorbike	81	101	76	103	157	204	—	—
Person	442	868	445	865	887	1733	—	—
Pottedplant	82	151	85	171	167	322	—	—
Sheep	63	155	57	153	120	308	—	—
Sofa	93	103	90	106	183	209	—	—
Train	83	96	84	93	167	189	—	—
Tvmonitor	84	101	74	98	158	199	—	—
Total	1464	3507	1449	3422	2913	6929	—	—

Постановка задачи

- Каждый объект содержит сильно больше информации, чем при классификации!
- Можно хорошо обучаться на небольших выборках

Метрики качества

- Попиксельная доля верных ответов:

$$L(y, a) = \frac{1}{n} \sum_{i=1}^n [y_i = a_i]$$

- Мера Жаккара (считается отдельно для каждого класса):

$$J_k(y, a) = \frac{\sum_{i=1}^n [y_i = k][a_i = k]}{\sum_{i=1}^n \max([y_i = k], [a_i = k])}$$

(можно усреднить по всем классам)

ФУНКЦИЯ ПОТЕРЬ

- Для одного изображения:

$$L(y, a) = \sum_{i=1}^n \sum_{k=1}^K [y_i = k] \log a_{ik}$$

сумма по пикселям

сумма по классам

истинный класс в i -м пикселе

вероятность k -го класса в i -м пикселе
(из модели)

ФУНКЦИЯ ПОТЕРЬ

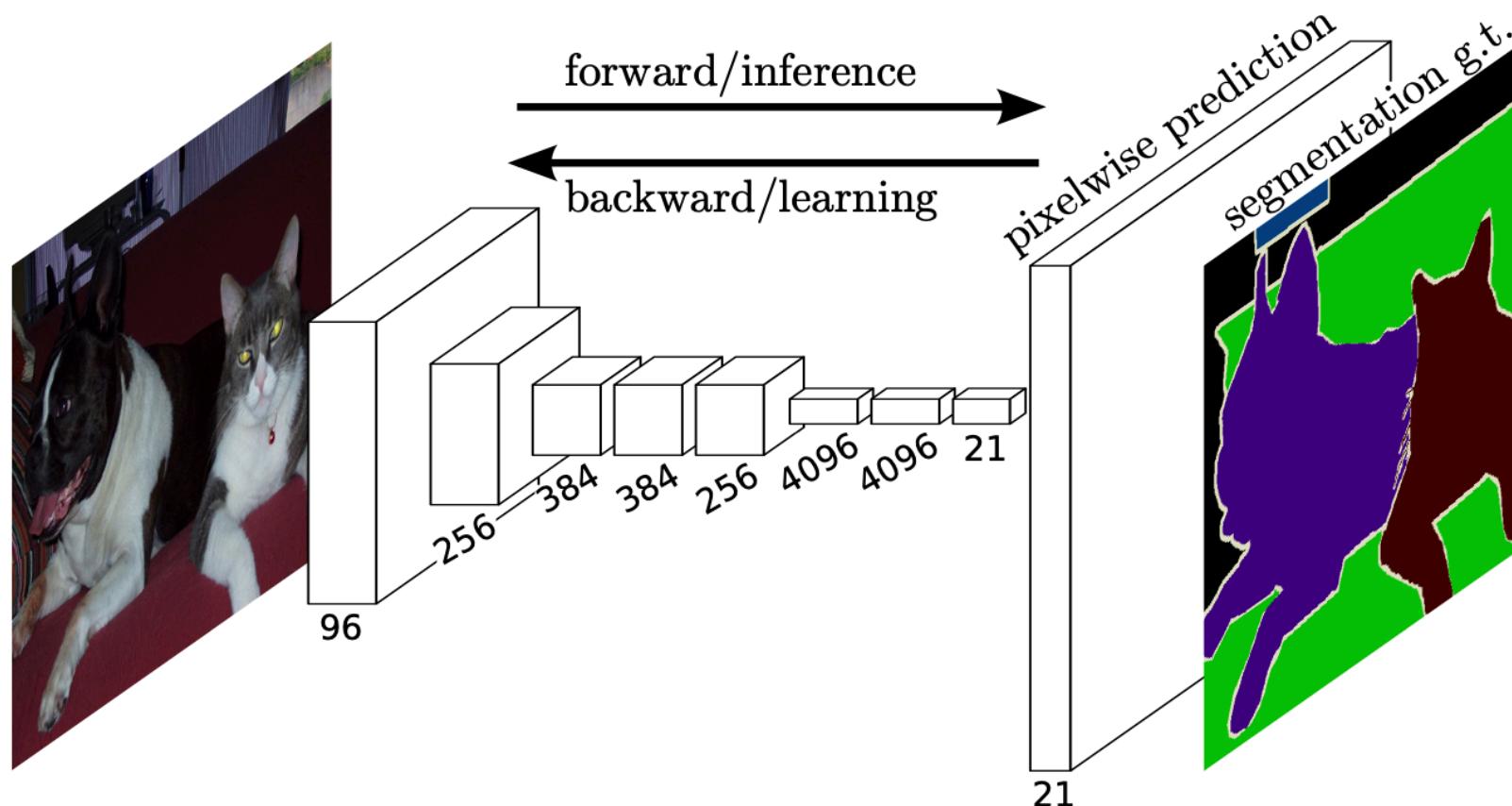
- Для одного изображения (categorical cross-entropy, CCE):

$$L(y, a) = \sum_{i=1}^n \sum_{k=1}^K [y_i = k] \log a_{ik}$$

- Если модель в i -м пикселе выдаёт какие-то числа b_{i1}, \dots, b_{iK} , то их можно превратить в вероятности через softmax:

$$a_{ik} = \frac{\exp(b_{ik})}{\sum_{m=1}^K \exp(b_{im})}$$

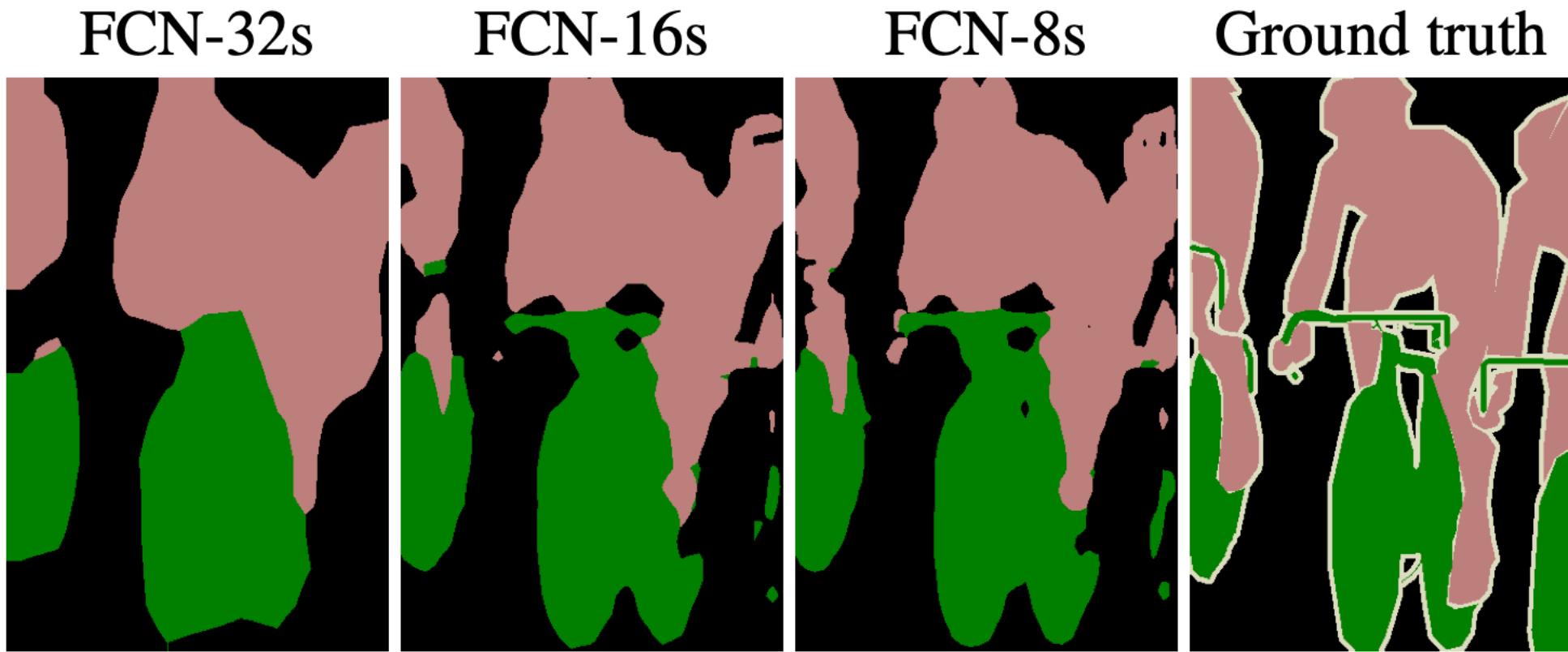
Fully Convolutional Network



Fully Convolutional Network

- Убираем полносвязные слои
- Остаются только свёртки
- Тензор с последнего слоя преобразуем свёртками 1×1 в тензор такого же размера, но с K каналами (по числу классов)
- Повышаем разрешение
 - Можно простым размножением пикселей
 - Можно хитрее

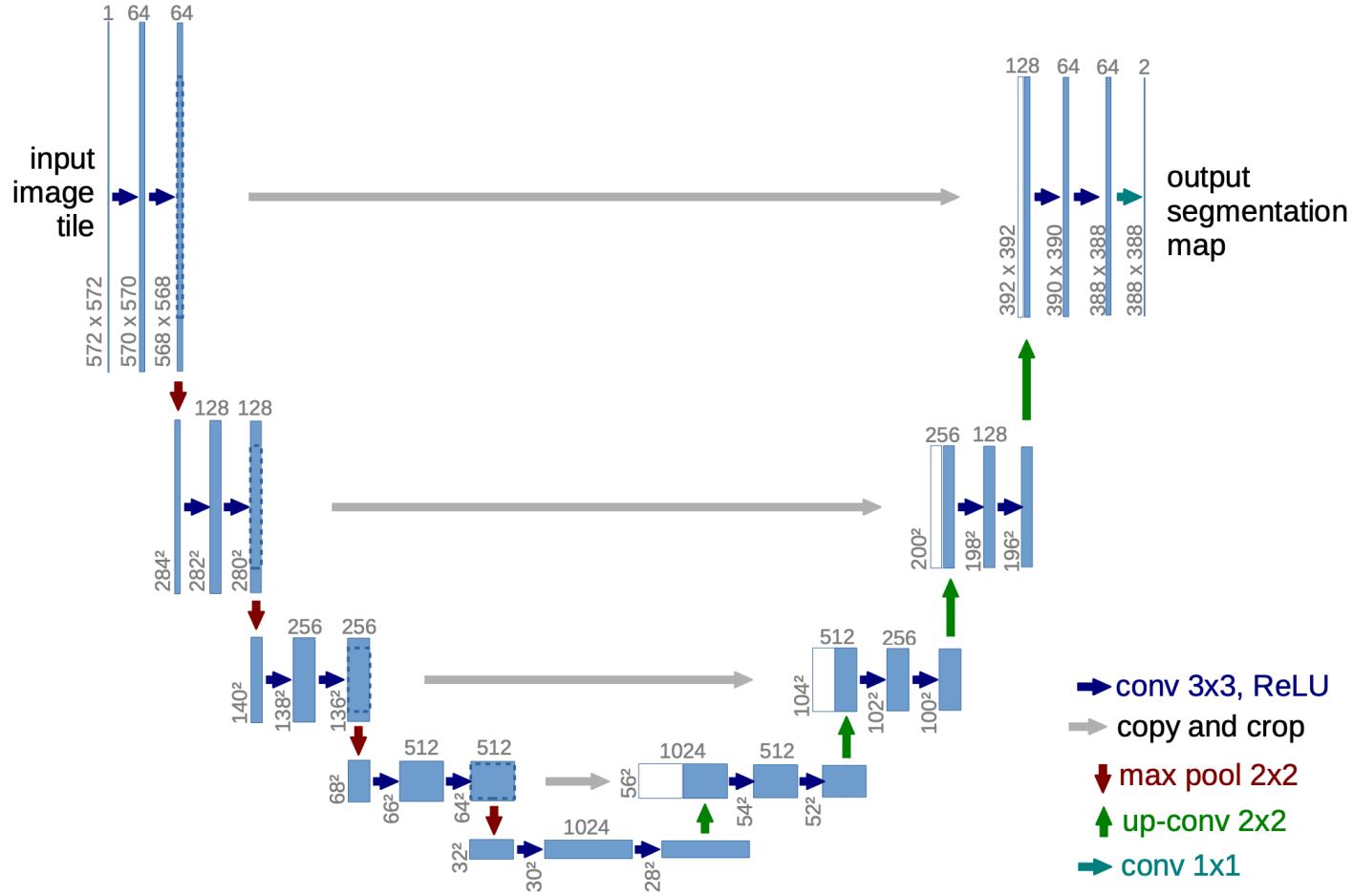
Fully Convolutional Network



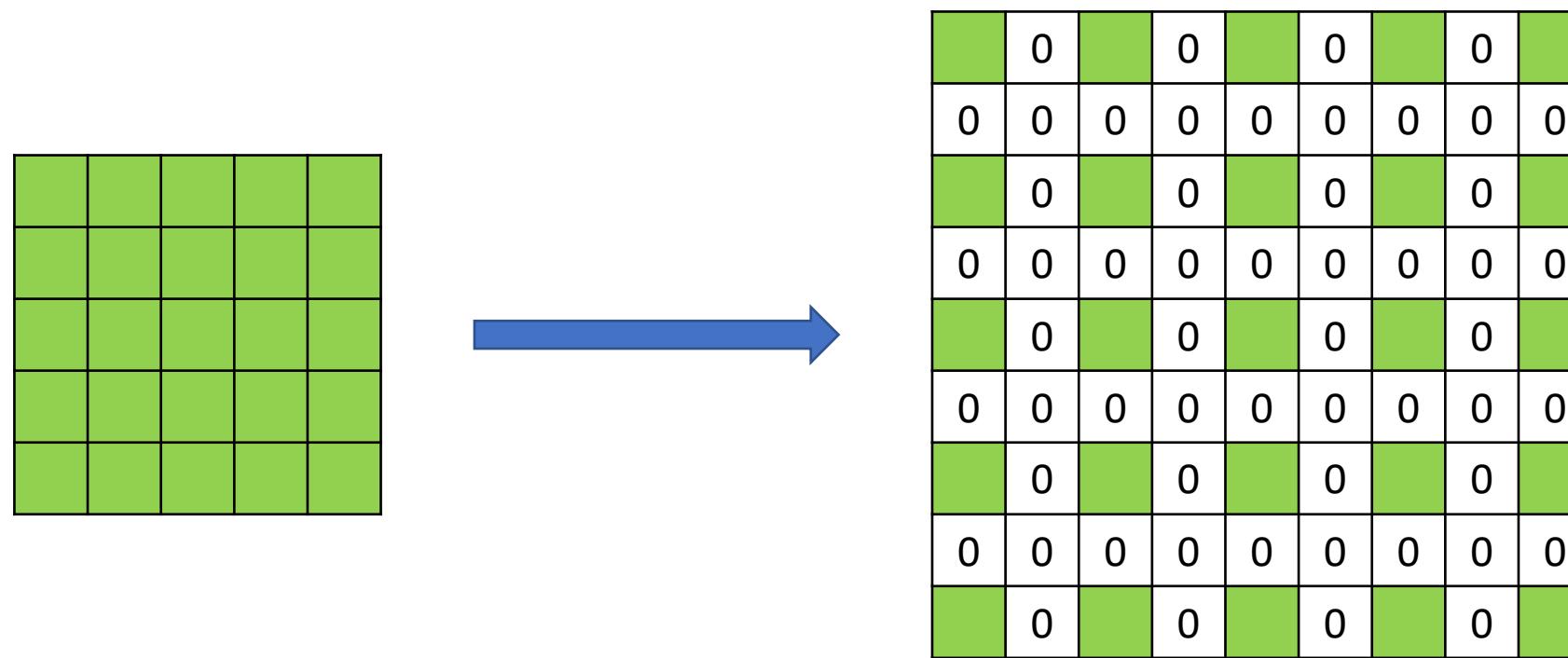
Чем это плохо?

- Тензор на последнем слое довольно маленький
- Классы предсказываются грубо, у объектов нечёткие границы
- Можно делать меньше пулингов
- Но тогда будут проблемы с размером поля восприятия

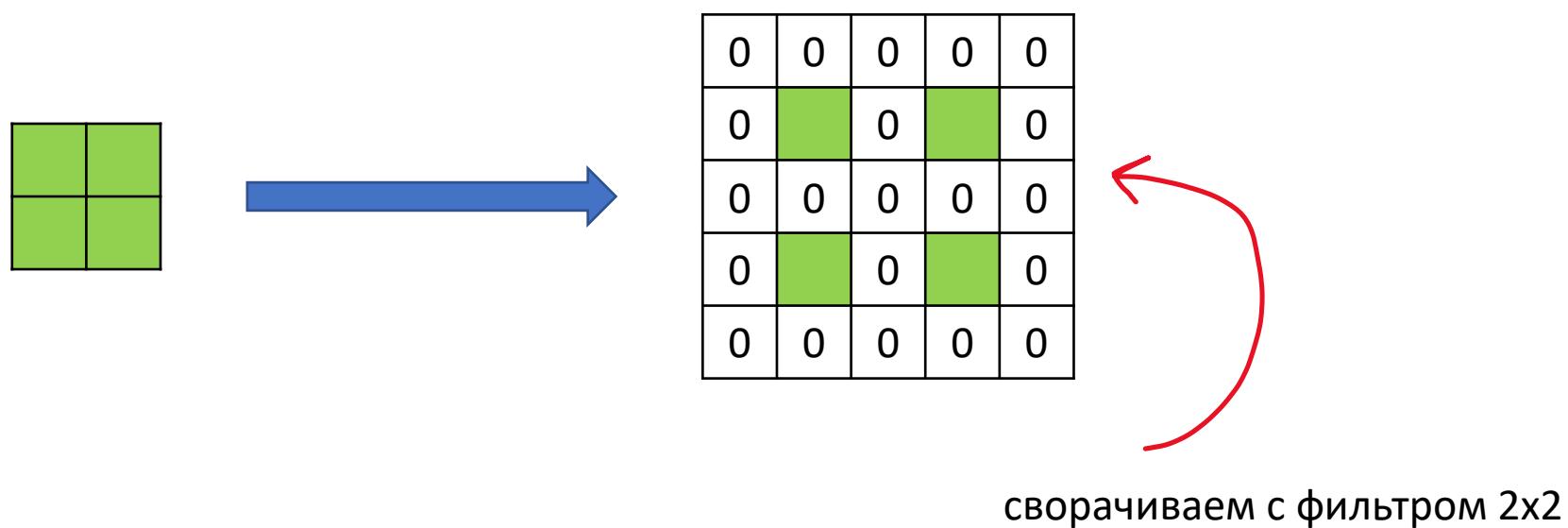
U-Net



Upsampling, вариант 1 (bed of nails)



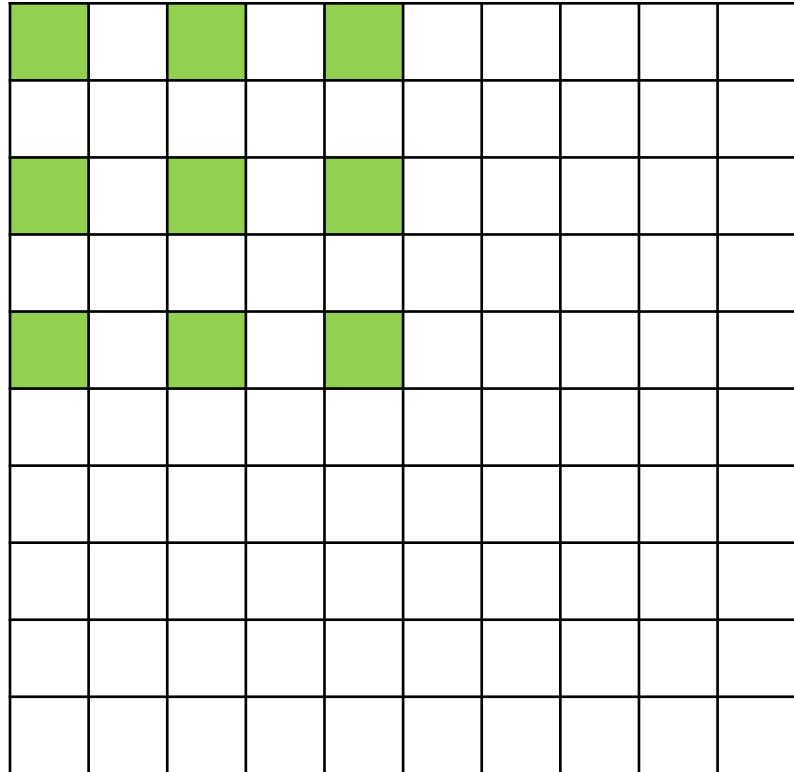
Upsampling, вариант 2 (up-convolution)



U-Net

- «Декодировщик» имеет очень большое поле восприятия
- Главное отличие от FCN — копирование слоёв между ветками сети

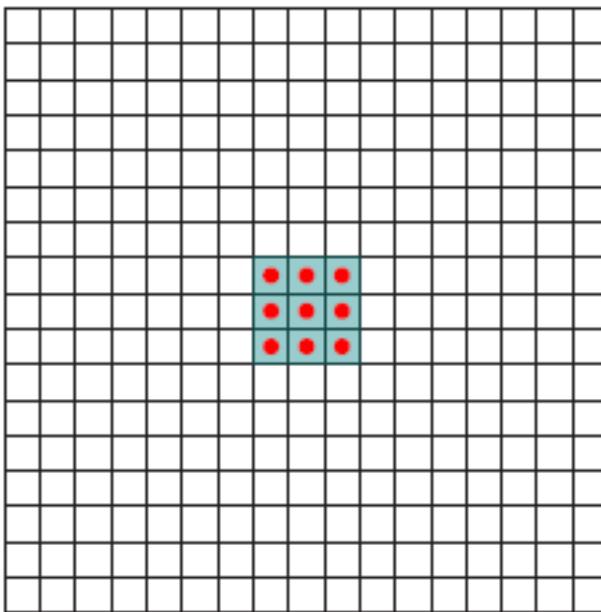
Dilated convolutions («раздутые» свёртки)



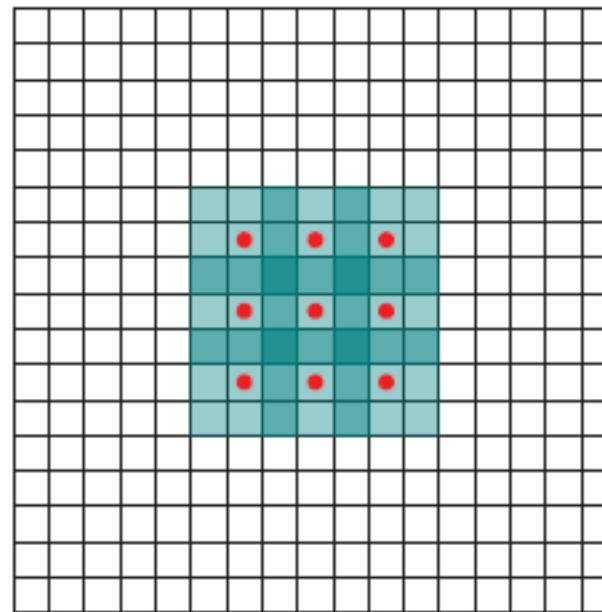
$$\begin{matrix} * & \begin{matrix} & & \\ & & \\ & & \end{matrix} & = & \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \end{matrix}$$

$$l = 2$$

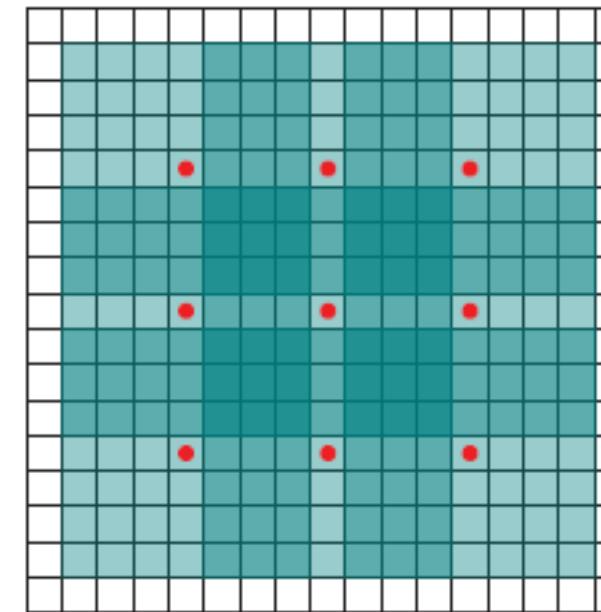
Dilated convolutions («раздутые» свёртки)



$$l = 1$$



$$l = 2$$



$$l = 4$$

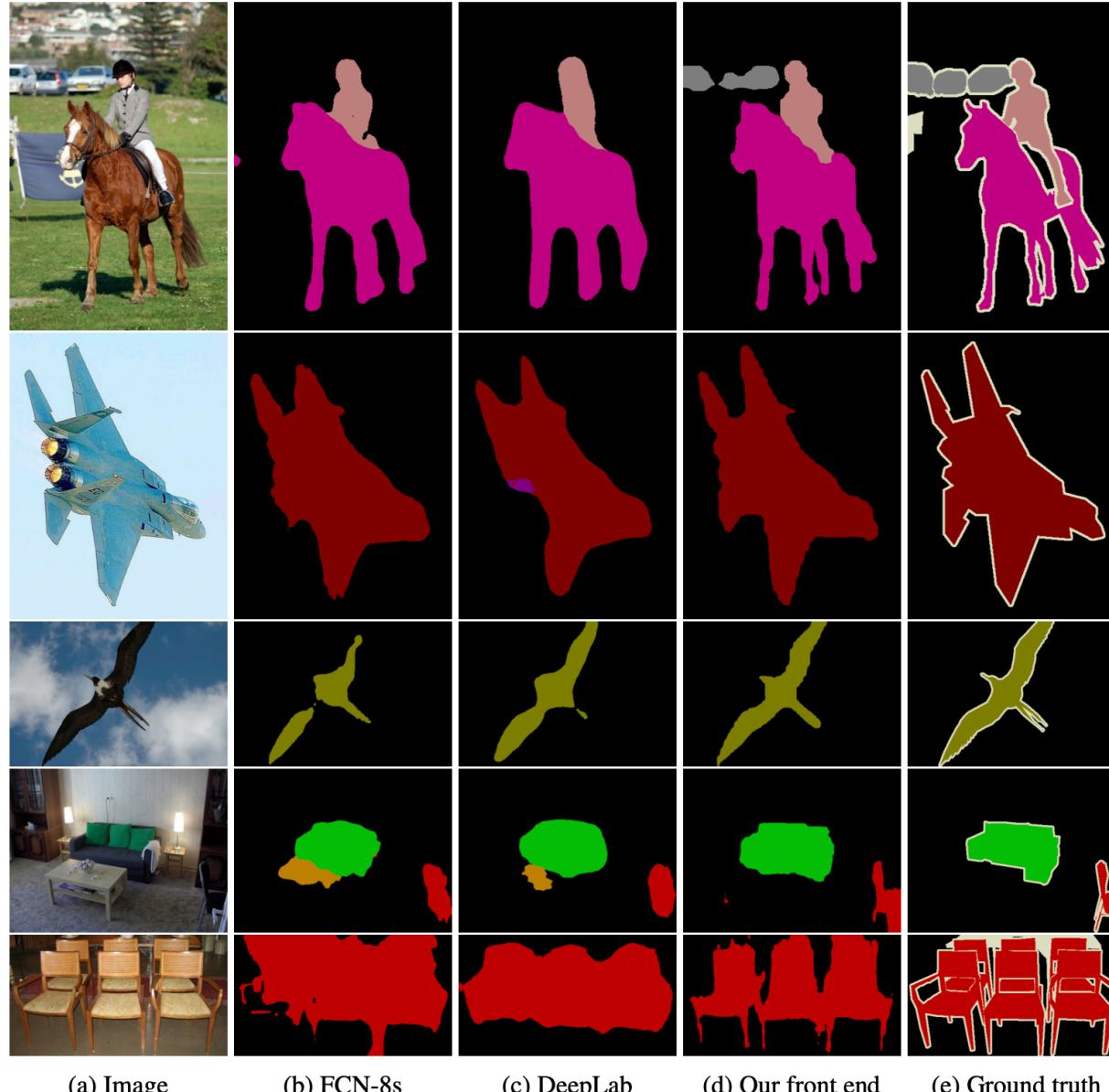
Dilated convolutions

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67
Output channels								
Basic	C	C	C	C	C	C	C	C
Large	$2C$	$2C$	$4C$	$8C$	$16C$	$32C$	$32C$	C

Table 1: Context network architecture. The network processes C feature maps by aggregating contextual information at progressively increasing scales without losing resolution.

Dilated convolutions

- Можем сохранять размер тензора и при этом увеличивать поле восприятия



(a) Image

(b) FCN-8s

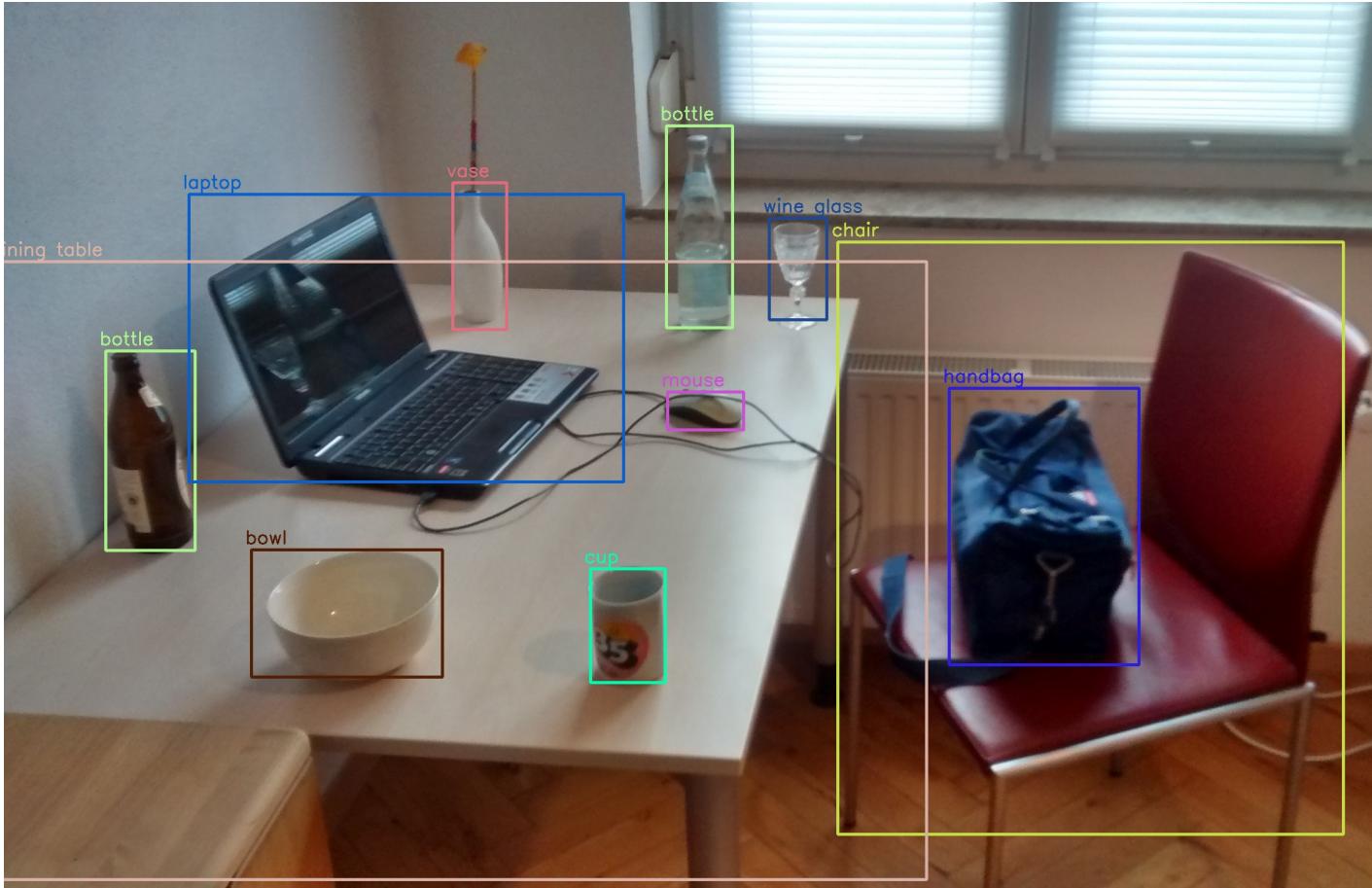
(c) DeepLab

(d) Our front end

(e) Ground truth

Детекция объектов

Задача детекции



Задача детекции

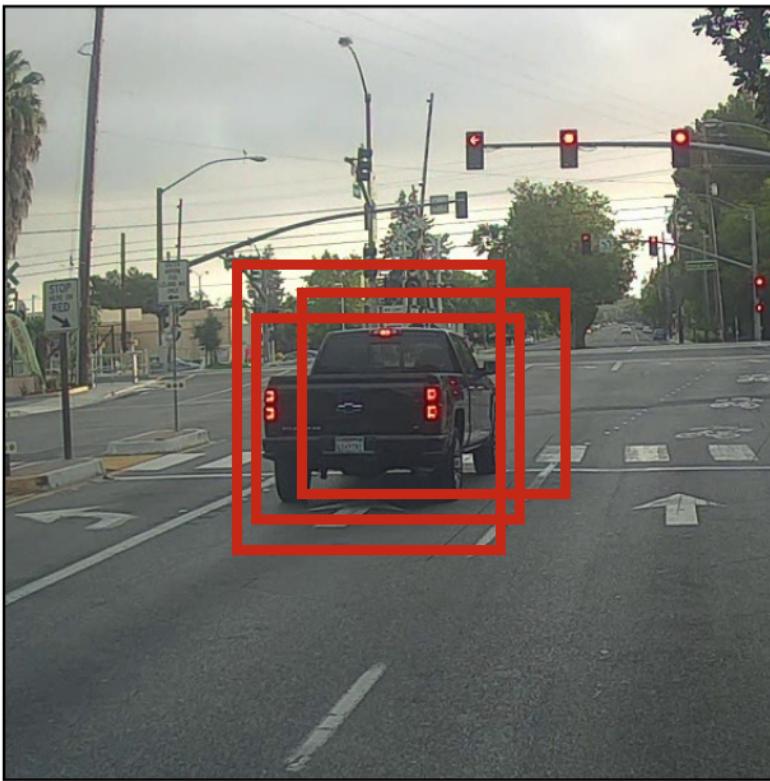
- Перебирать все прямоугольники слишком долго
- Не очень понятно, что такое хороший прямоугольник
- Прямоугольники разного размера, усложняется классификация

Non-maximum suppression

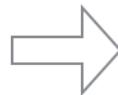
- Модель выдаёт для класса k список прямоугольников с уверенностями
- Проходим в порядке уменьшения уверенности
- Для каждого прямоугольника удаляем все последующие, с которыми Intersection over Union (IoU) > 0.5

Non-maximum suppression

Before non-max suppression



**Non-Max
Suppression**



After non-max suppression

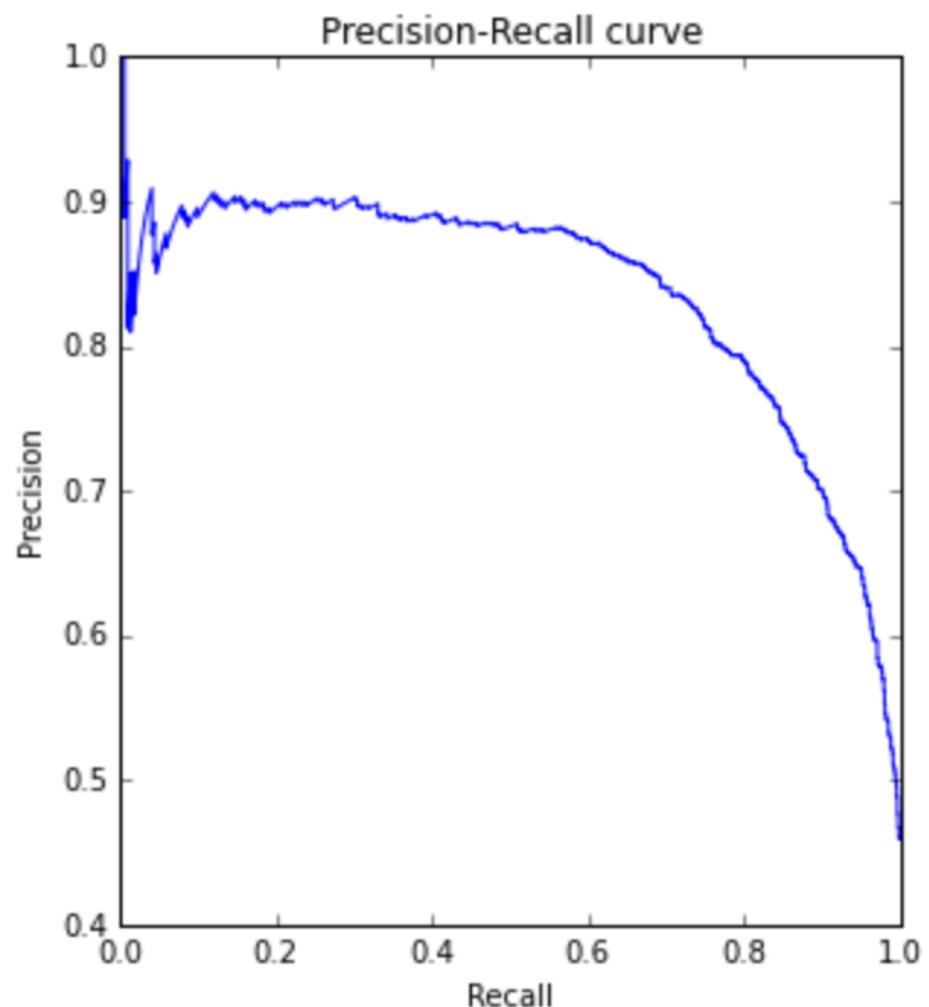


Метрики качества

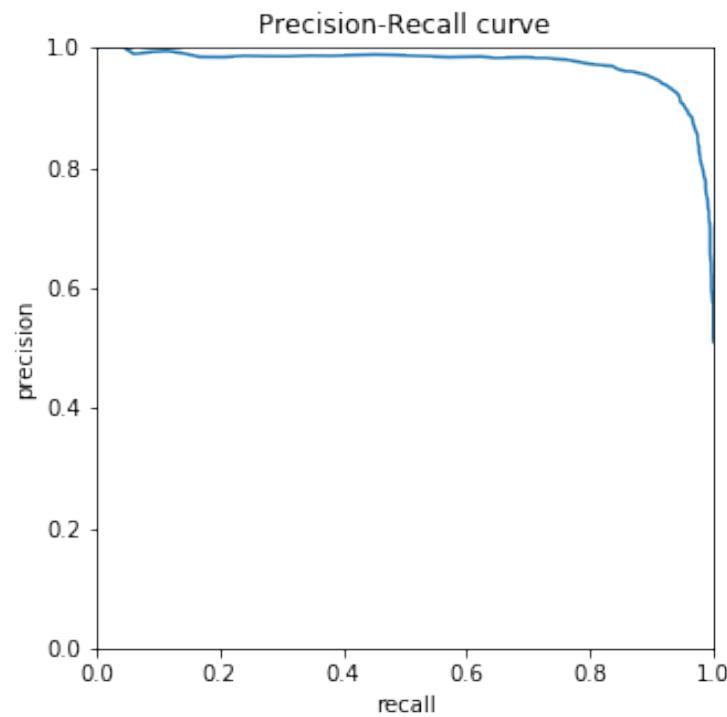
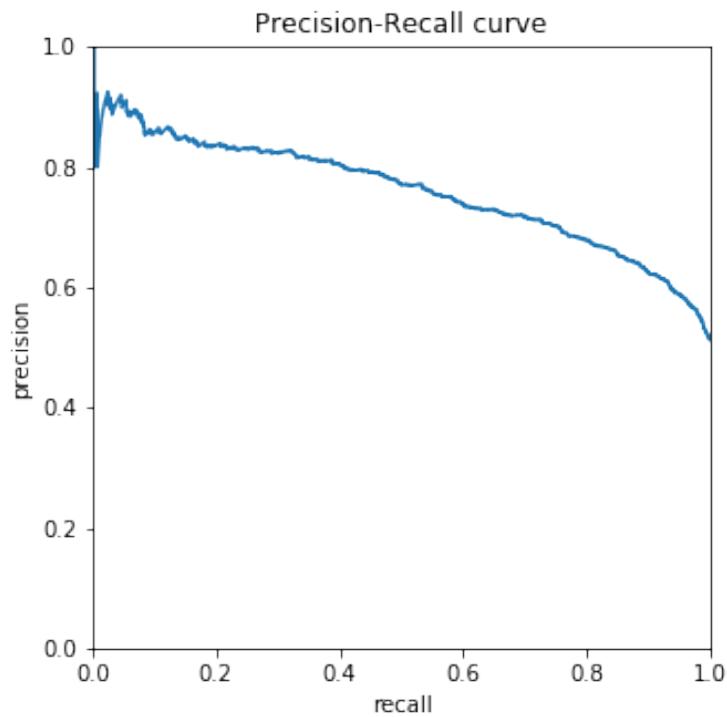
- Модель выдаёт для класса k список прямоугольников с уверенностями
- Считаем прямоугольник корректным, если
$$\text{IoU}(y, z) = \text{Jaccard}(y, z) > t$$
- Строим PR-кривую, считаем под ней площадь, получаем Average Precision
- Усредняем по всем классам, получаем mAP (mean AP)
- Раньше $t = 0.5$, сейчас скорее $t = 0.75$

PR-кривая

- Левая точка: $(0, 1)$
- Правая точка: $(1, r)$, r — доля положительных объектов
- Для идеального классификатора проходит через $(1, 1)$
- AUC-PRC — площадь под PR-кривой



PR-кривая



Two-shot detection

Будем решать задачу в два этапа:

1. Находим «кандидатов» — прямоугольники, где скорее всего что-то есть
2. Классифицируем эти прямоугольники

R-CNN

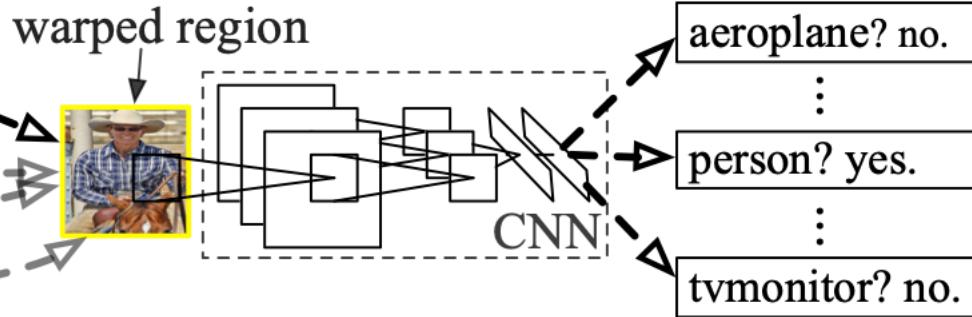
R-CNN: *Regions with CNN features*



1. Input
image



2. Extract region
proposals (~2k)



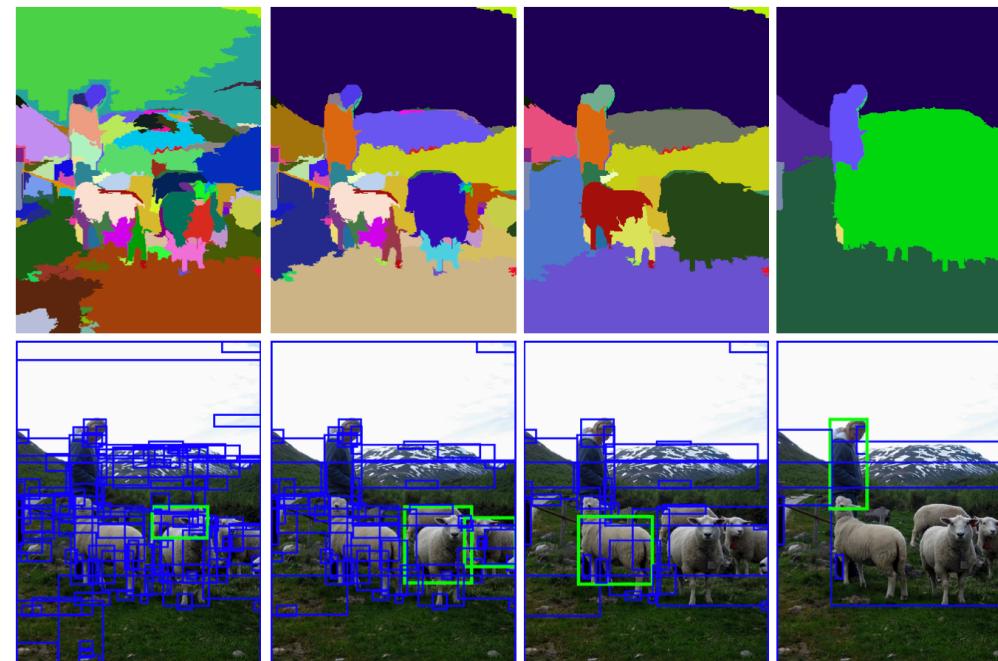
3. Compute
CNN features

4. Classify
regions

R-CNN

Генерация кандидатов:

- «Внешние» методы из классического компьютерного зрения
- Выбирается около 2000 прямоугольников



R-CNN

Извлечение признаков:

- Выходы предпоследнего слоя AlexNet (4096 чисел)

R-CNN

Классификация прямоугольников:

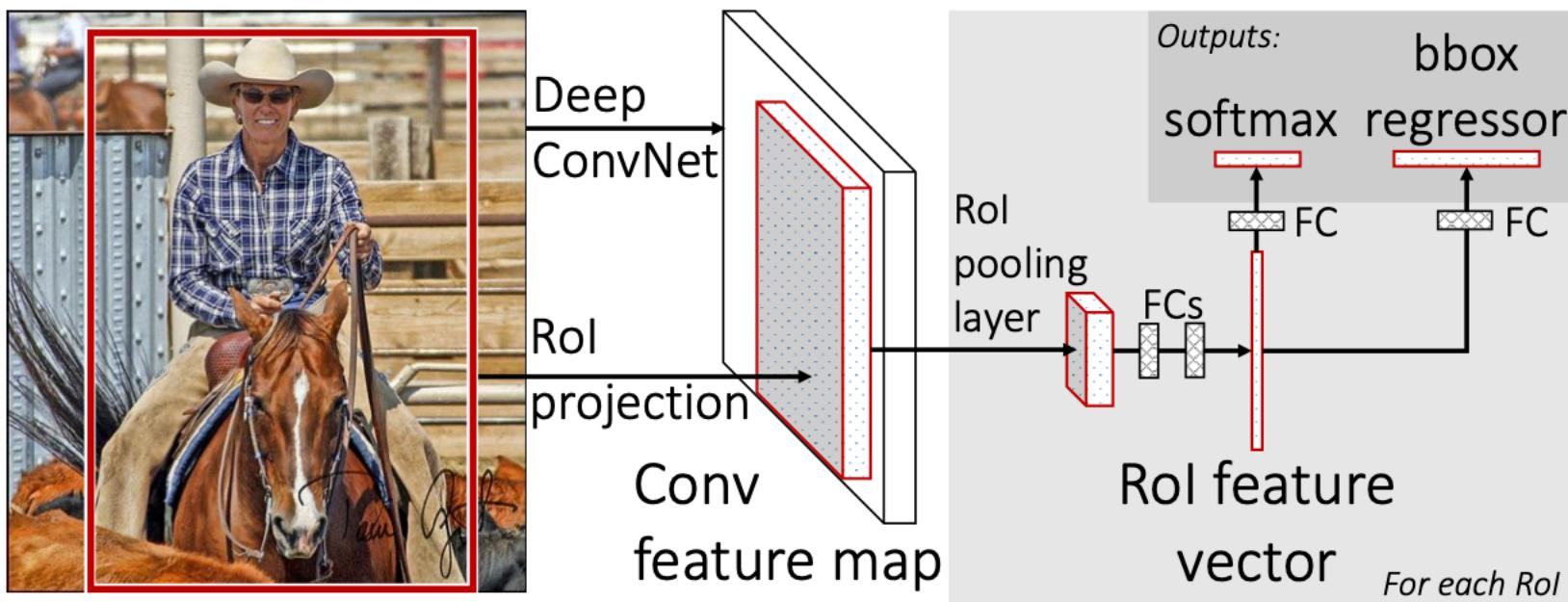
- SVM
- One-vs-all

R-CNN

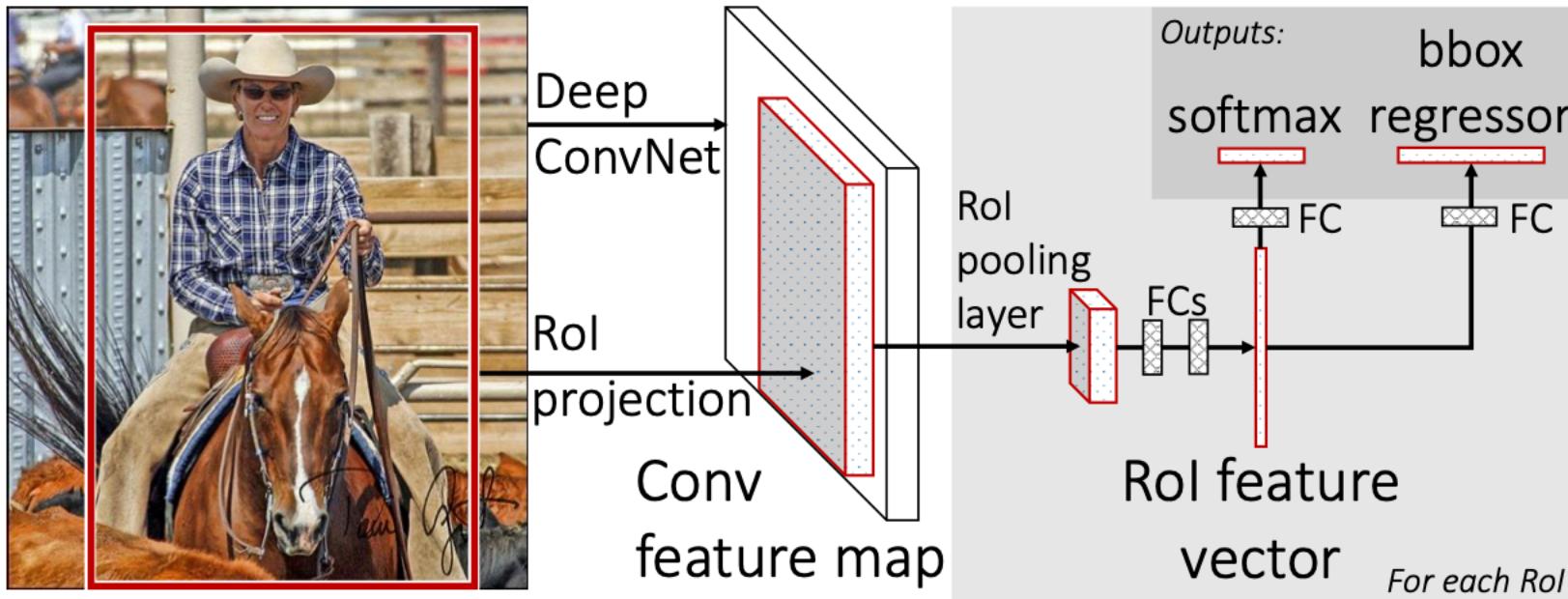
Проблемы:

- Не end-to-end
- Генерация кандидатов может быть очень сложной
- Свёрточная сеть не настраивается под данные
- Долго (много признаков, много классификаторов)

Fast R-CNN

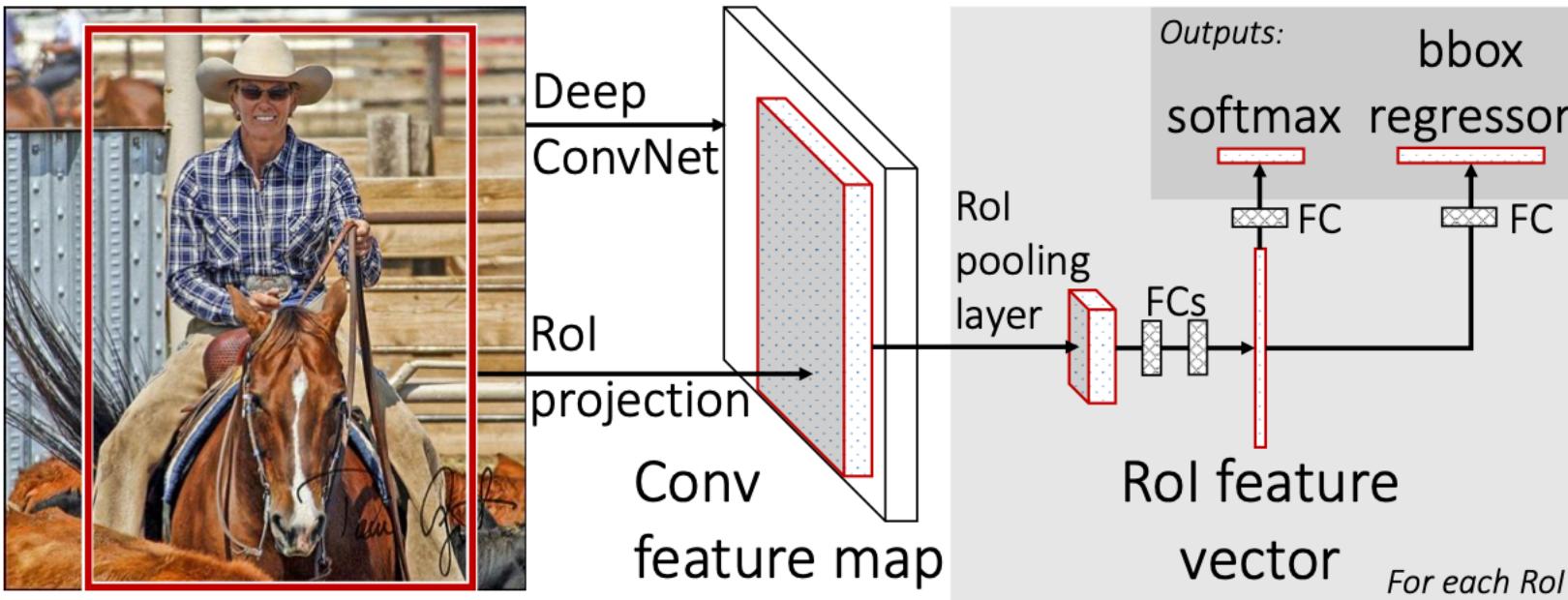


Fast R-CNN



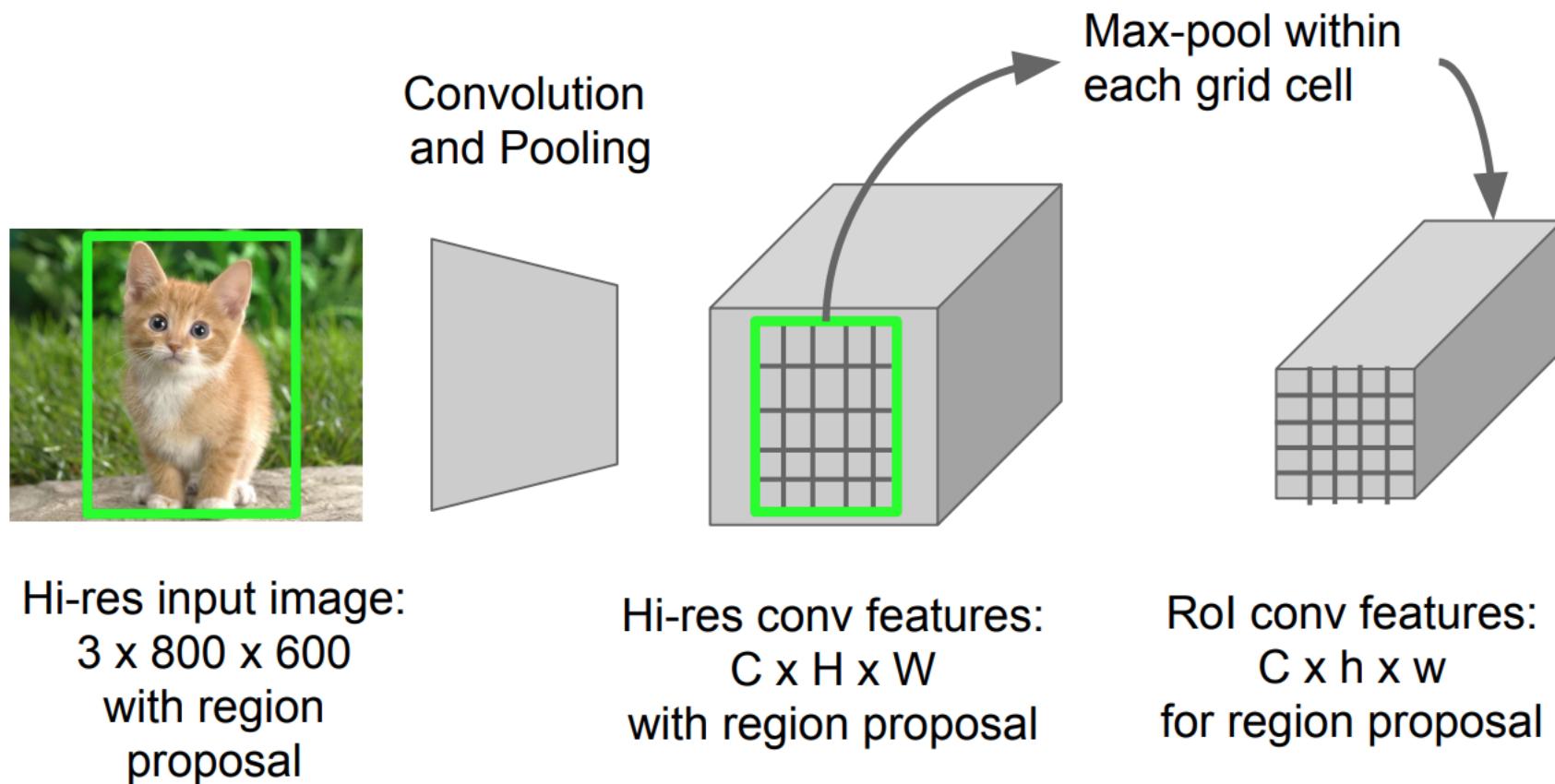
- Кандидаты всё ещё генерируются внешним методом

Fast R-CNN

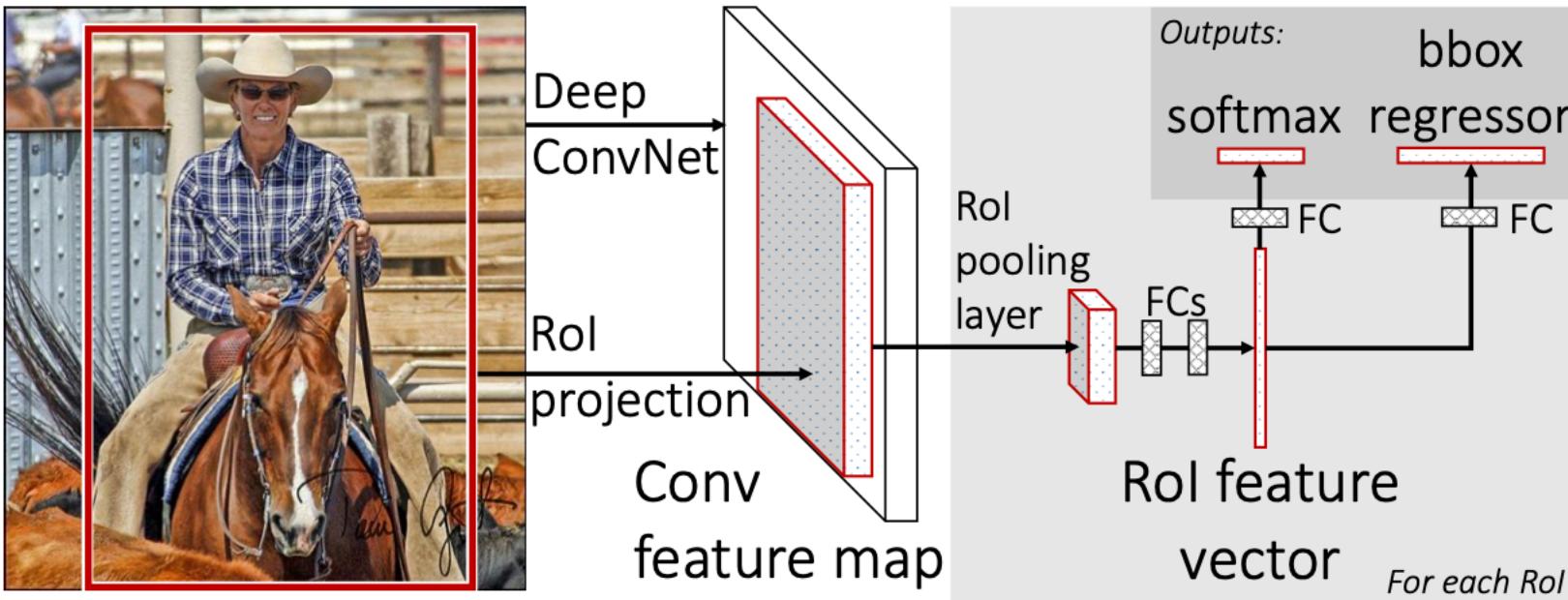


- Для конкретного кандидата извлекаются признаки: вырезаются участки из последнего тензора, режутся на блоки, из каждого блока берётся максимум

RoI Pooling



Fast R-CNN



- Для конкретного кандидата предсказываются вероятности классов и 4 поправки к размерам прямоугольника

Fast R-CNN

индикатор наличия объекта в области

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

CCE

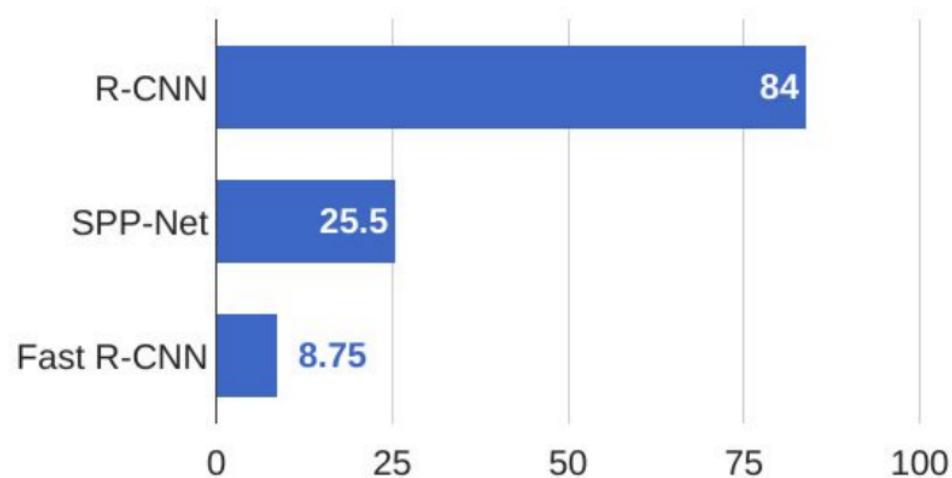
$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x}, \text{y}, \text{w}, \text{h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

in which

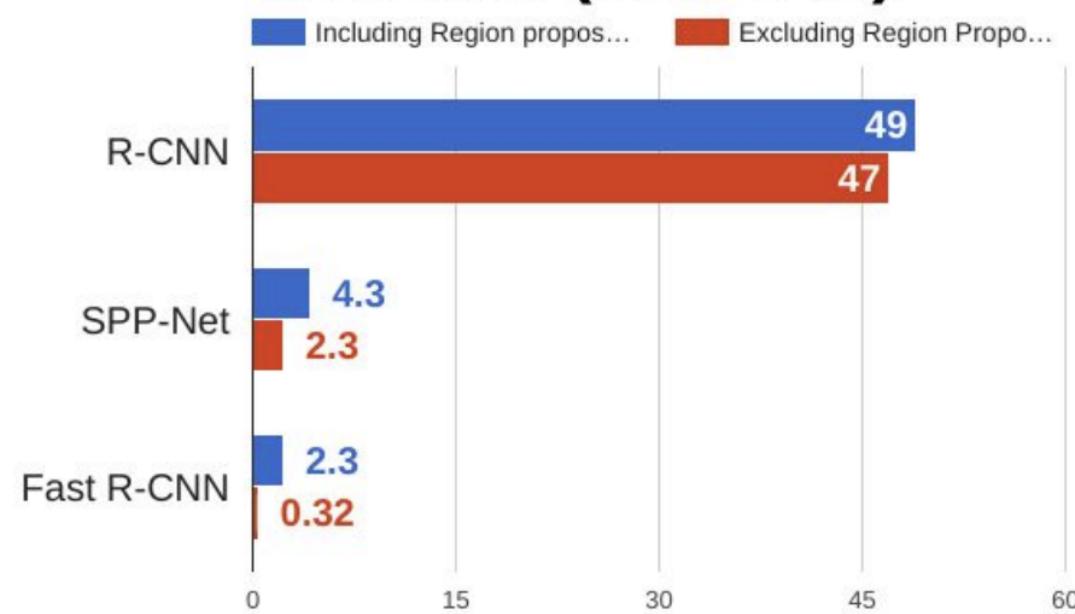
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Fast R-CNN

Training time (Hours)



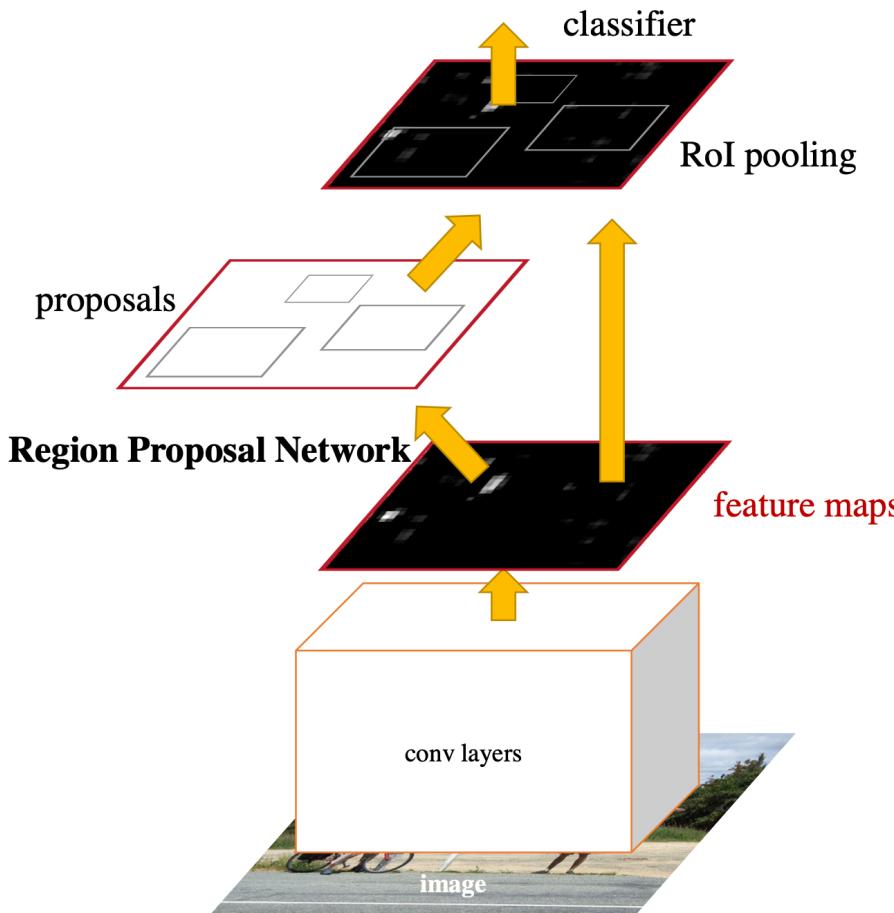
Test time (seconds)



Fast R-CNN

- Исправили почти все проблемы, но теперь генерация кандидатов занимает больше всего времени

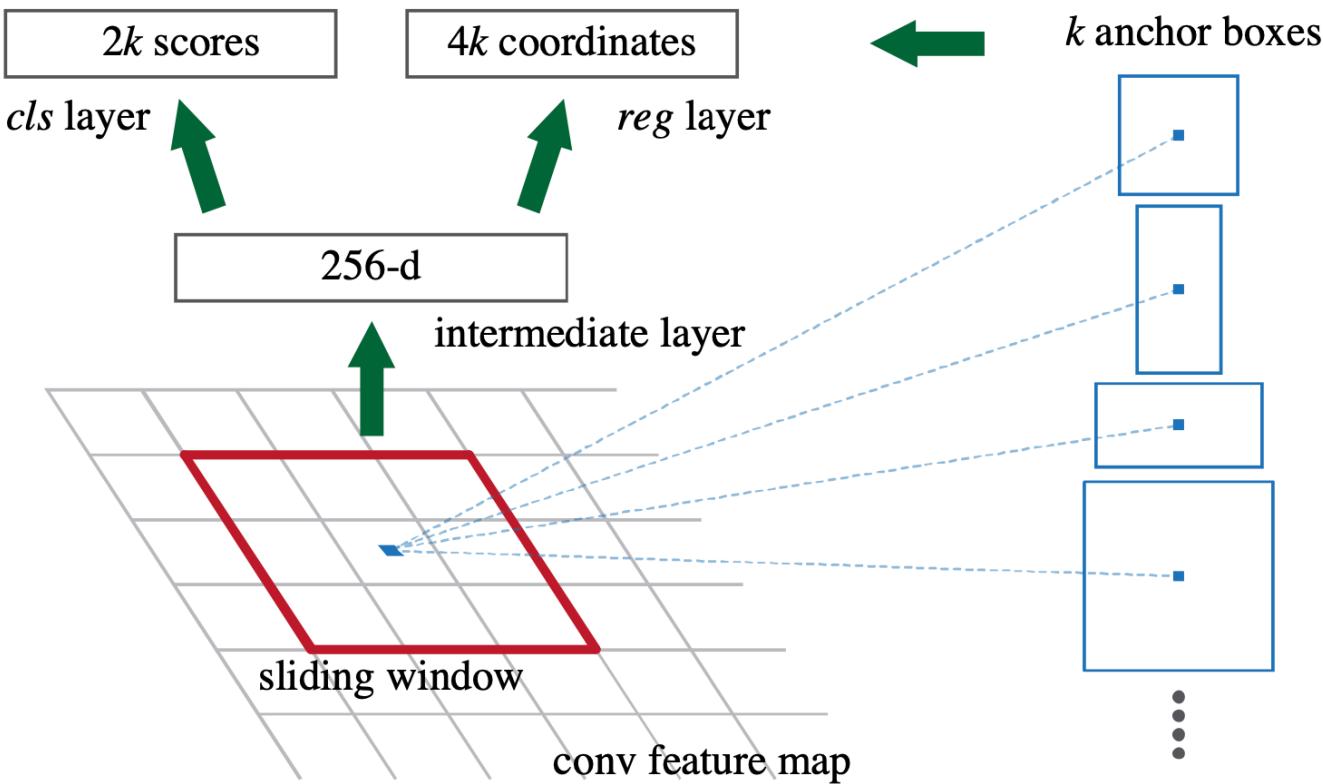
Faster R-CNN



Region Proposal Network

- Берём последний тензор из свёрточной сети
- Применяем свёрточный слой 3x3 с большим количеством каналов (256 или 512)
- Для каждой точки берём 9 прямоугольников с разными длинами сторон и соотношениями сторон
- Для каждого предсказываем:
 - Есть ли там объект
 - 4 поправки к сторонам

Region Proposal Network



Faster R-CNN

Всё обучается совместно, оптимизируем сумму 4 функций потерь:

- Классификация «есть ли объект» в RPN
- Регрессия для корректировки прямоугольника в RPN
- Классификация на K классов в основной модели
- Регрессия для корректировки прямоугольника в основной модели

Faster R-CNN

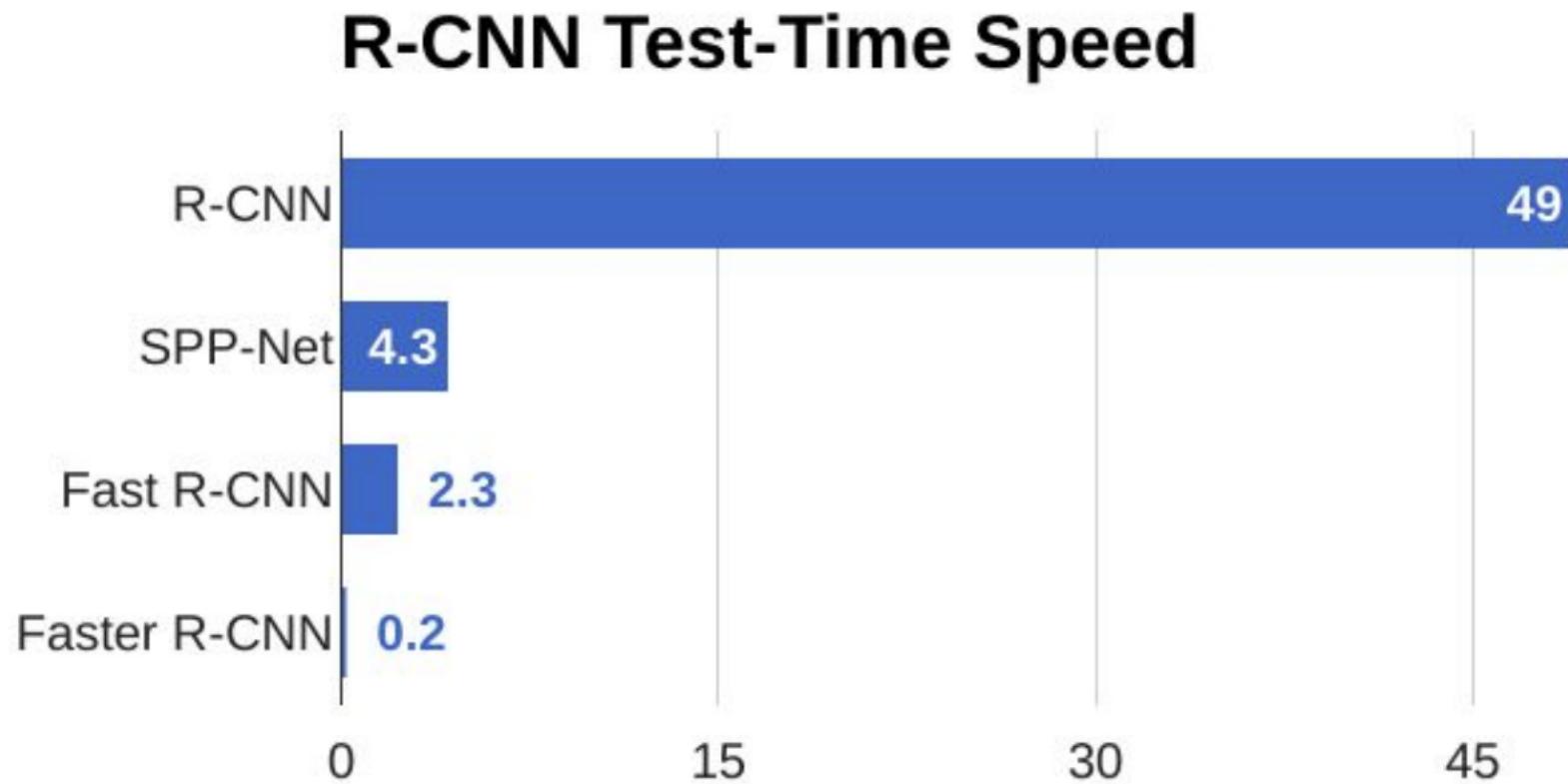
На этапе применения:

- Выделяем самые уверенные регионы из RPN
- Подаём в основную модель для детекции

Faster R-CNN

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

Faster R-CNN



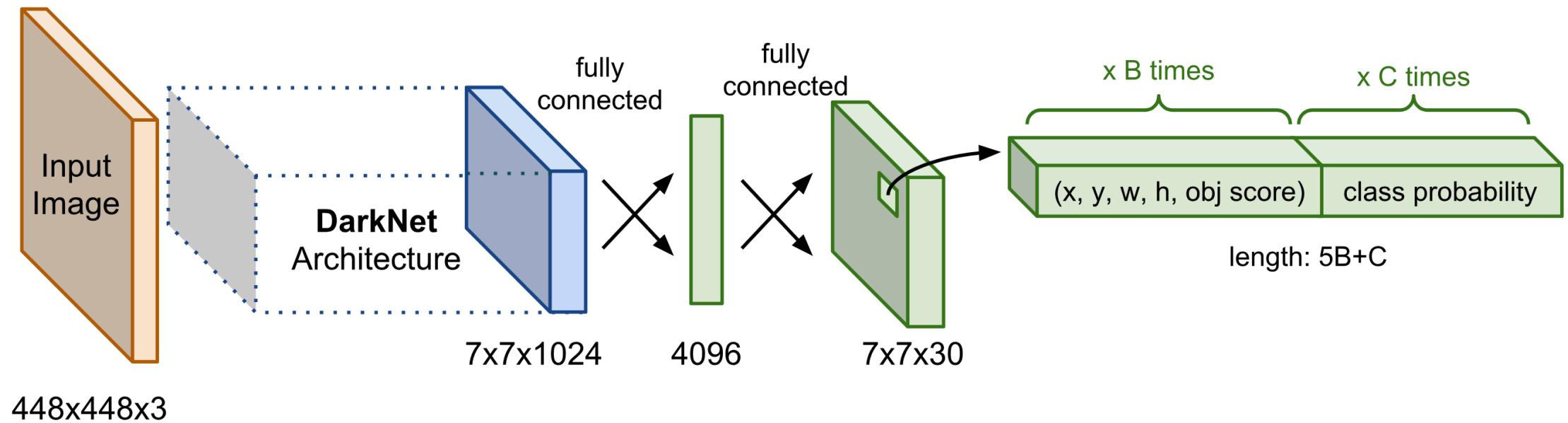
Что ещё?

- Mask R-CNN — instance segmantation

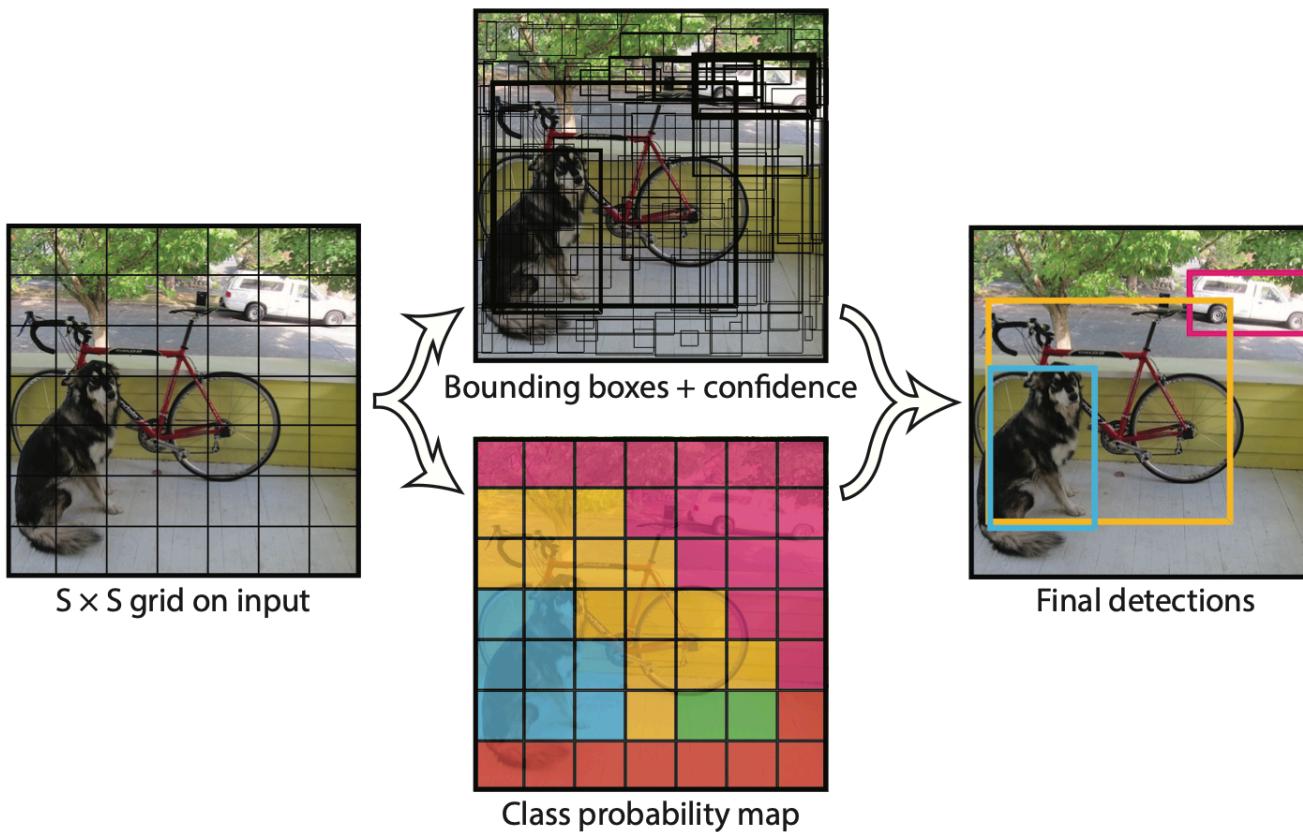
One-shot detection

- Двухэтапные детекторы работают хорошо, но не очень быстро
- Попытаемся одновременно искать кандидатов, и определять их классы

YOLO



YOLO



YOLO

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Что ещё?

- SSD (Single Shot Detector)
- RetinaNet
- YOLOv3