

# **Детекция и сегментация**

**Лекция 5**

**Data Mining in Action / Deep Learning / 2019**

**Эмиль Каюмов  
@emilkayumov**

# План

- Детекция объектов
- Семантическая сегментация
- Сегментация отдельных объектов
- Компьютерное зрение для видео
- Датасеты/реализации/...

# План

- Детекция объектов
- Семантическая сегментация
- Сегментация отдельных объектов
- Компьютерное зрение для видео
- Датасеты/реализации/...

## Classification



CAT

Single object

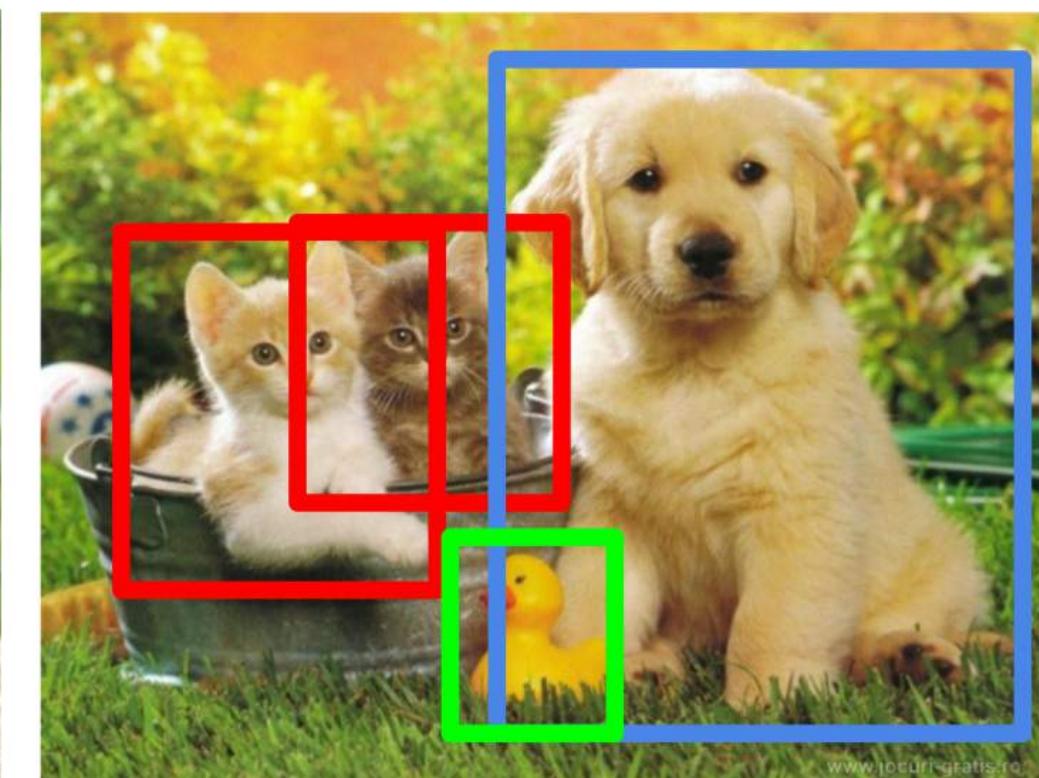
## Classification + Localization



CAT

Single object

## Object Detection



CAT, DOG, DUCK

Multiple objects

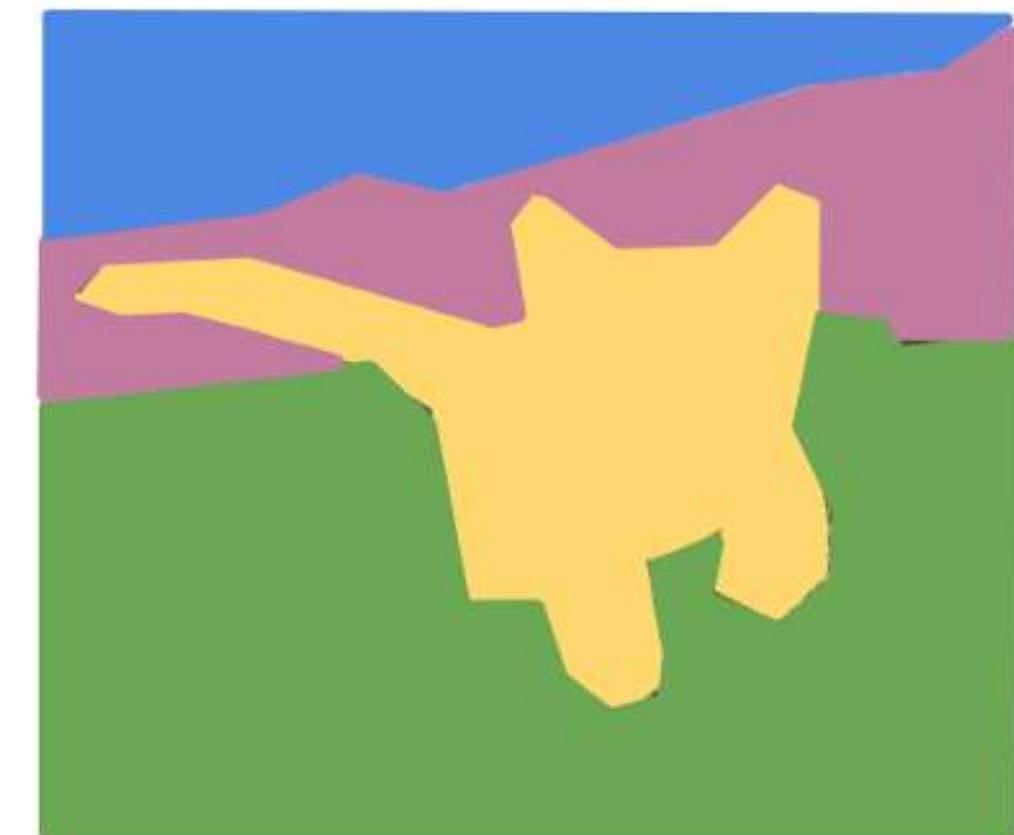
## Instance Segmentation



CAT, DOG, DUCK

Multiple objects

## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

cs231n slides

# Типы задач

- Детекция объектов – **найти все объекты** (обвести рамкой – bounding box / bbox) и **определить класс**
- Семантическая сегментация – определить **класс каждого пикселя** (сделать маски)
- Сегментация отдельных объектов – найти группы **пикселей одного объекта** (тэгировать каждый пиксель объектом или фоном + иногда класс)

# Классификация



This image is CC0 public domain

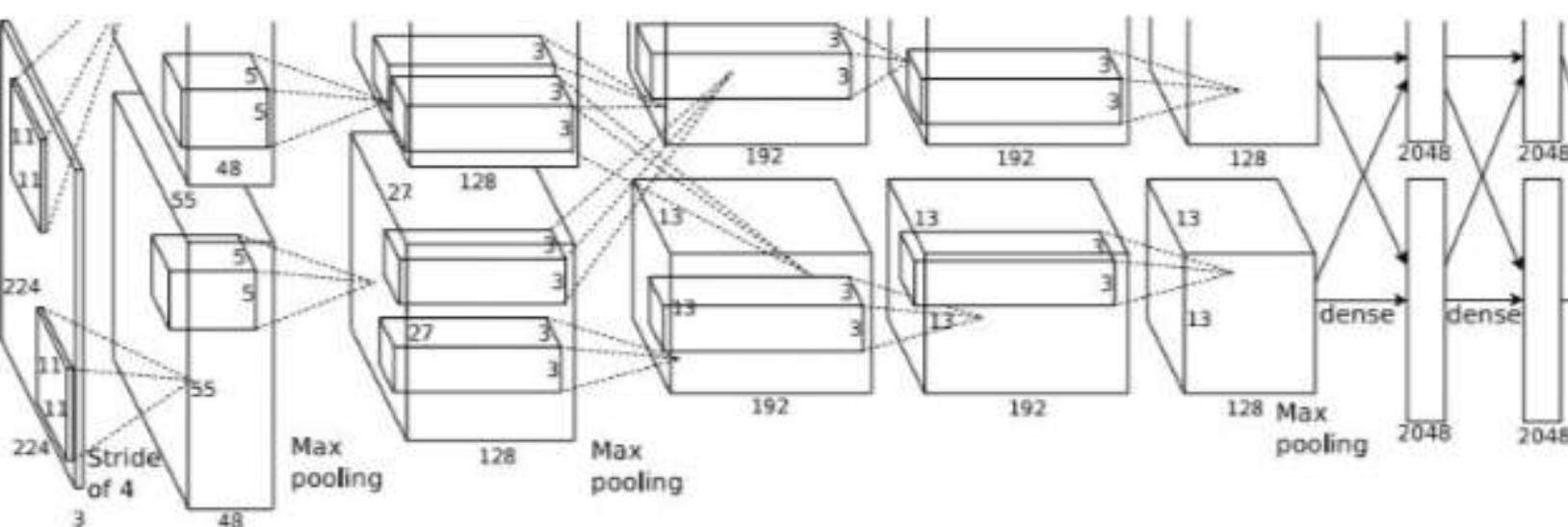


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

**Vector:**  
4096

**Fully-Connected:**  
4096 to 1000

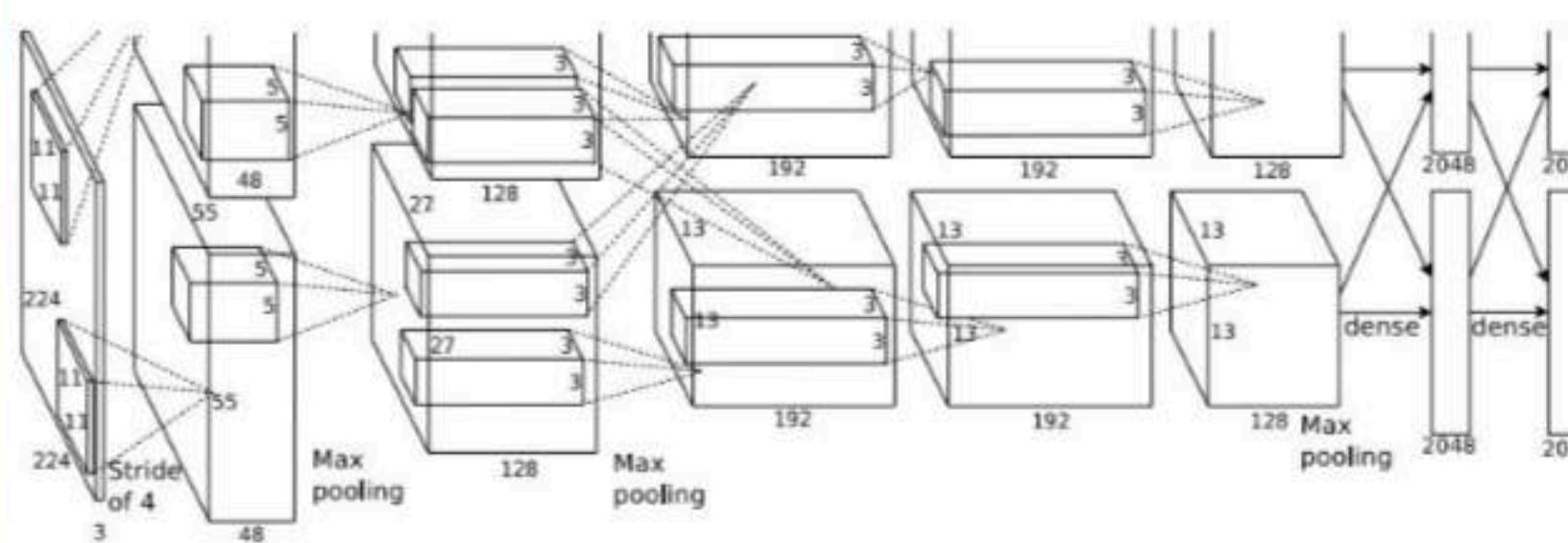
**Class Scores**  
Cat: 0.9  
Dog: 0.05  
Car: 0.01  
...

cs231n slides

# Классификация + локализация



This image is CC0 public domain



**Vector:** 4096  
**Fully Connected:** 4096 to 4

**Class Scores**

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

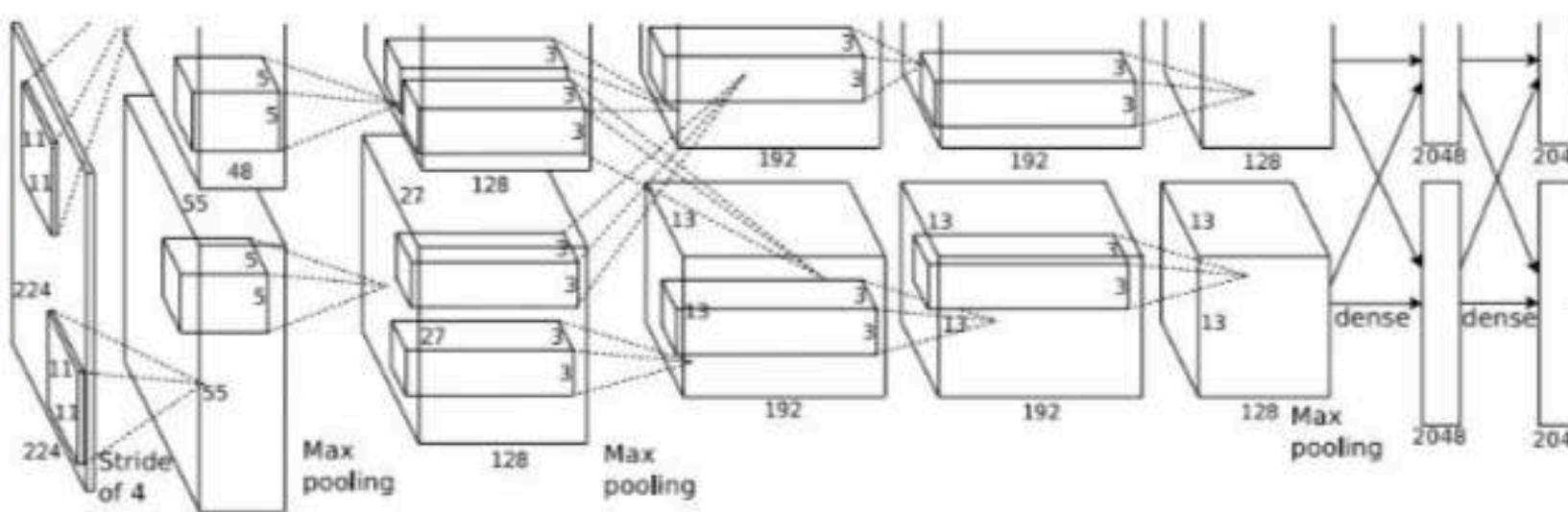
**Box Coordinates**  
( $x, y, w, h$ )

cs231n slides

# Классификация + локализация



This image is CC0 public domain



**Vector:**  
4096

**Fully  
Connected:**  
4096 to 1000

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

**Box  
Coordinates** → **L2 Loss**  
( $x, y, w, h$ )

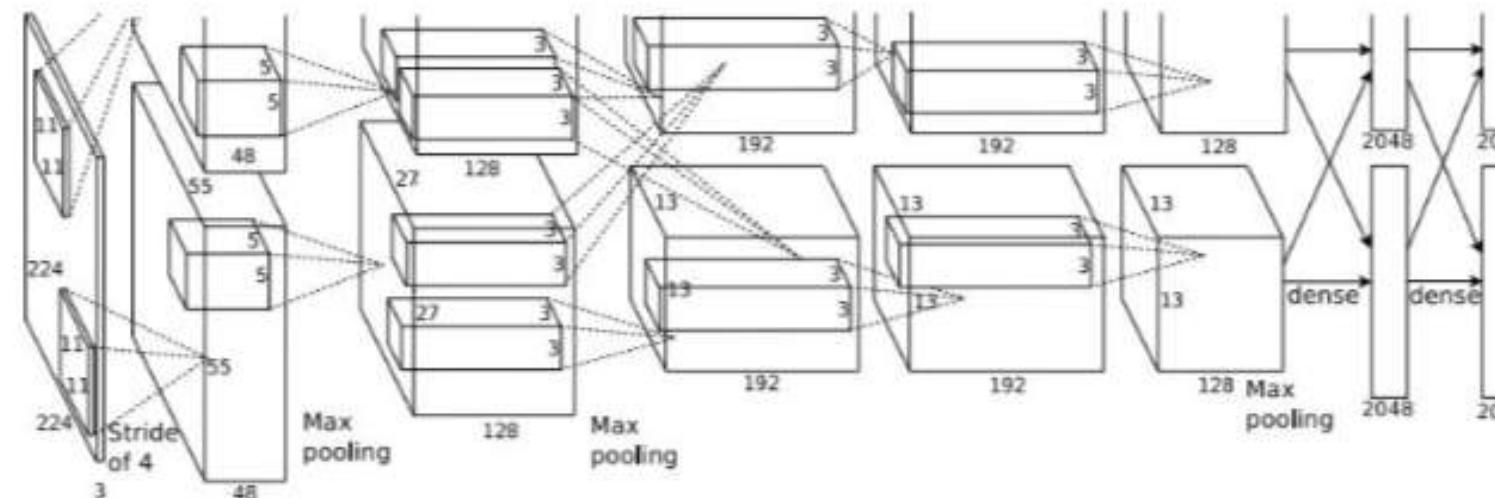
**Correct label:**  
Cat

**Softmax  
Loss**

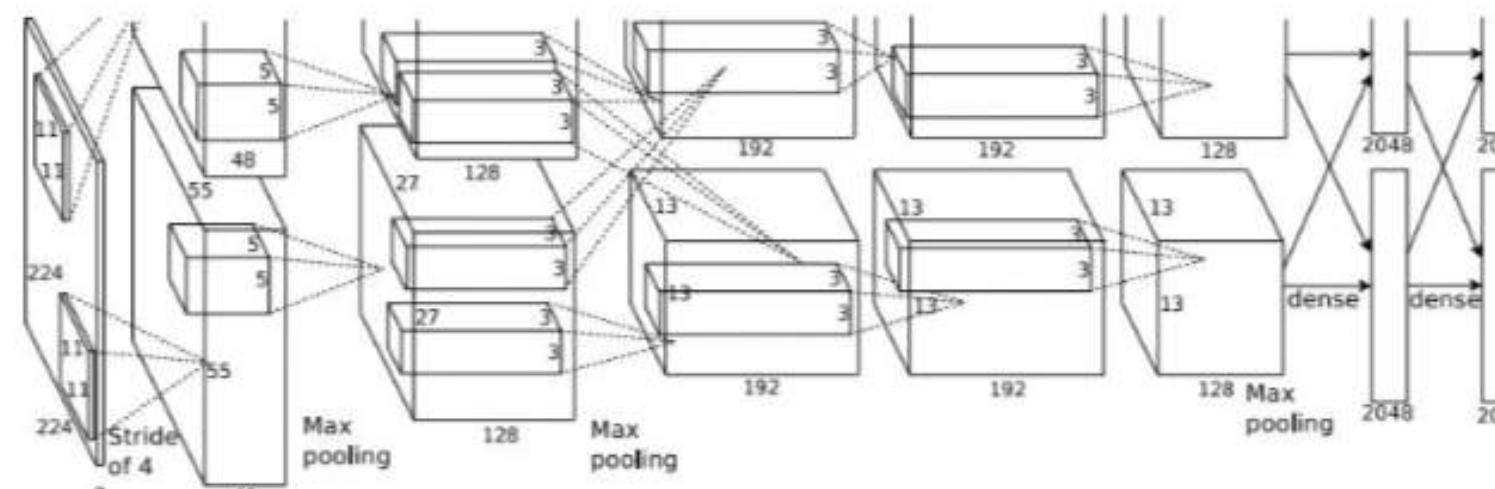
**+** → **Loss**

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

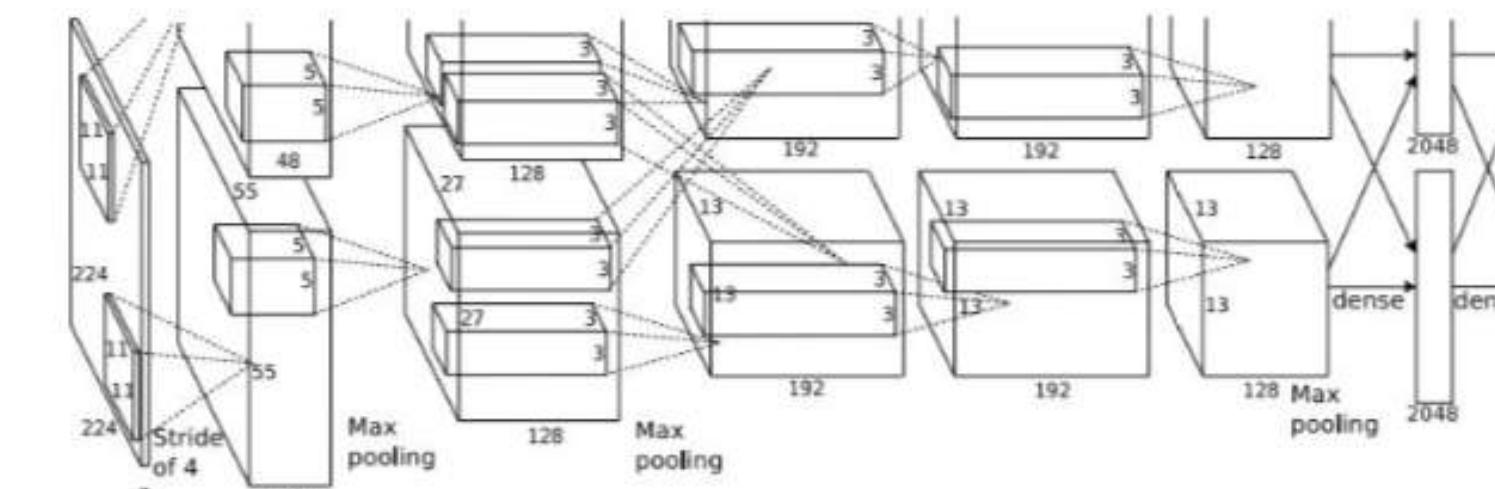
# Детекция объектов



CAT: (x, y, w, h)



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



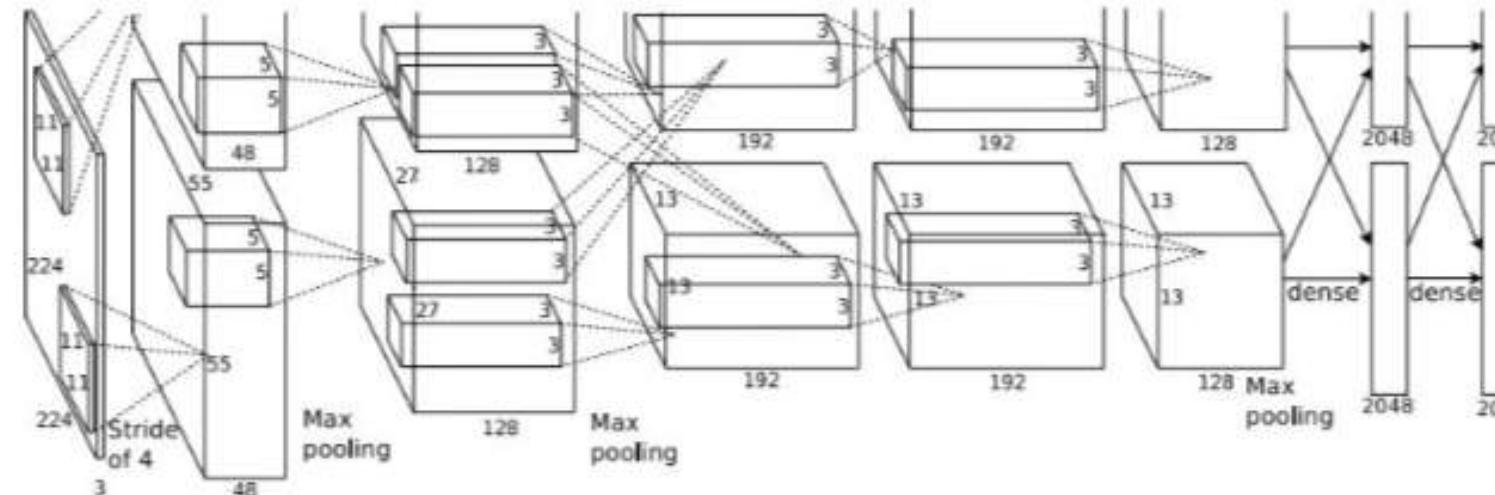
DUCK: (x, y, w, h)  
DUCK: (x, y, w, h)

...

cs231n slides

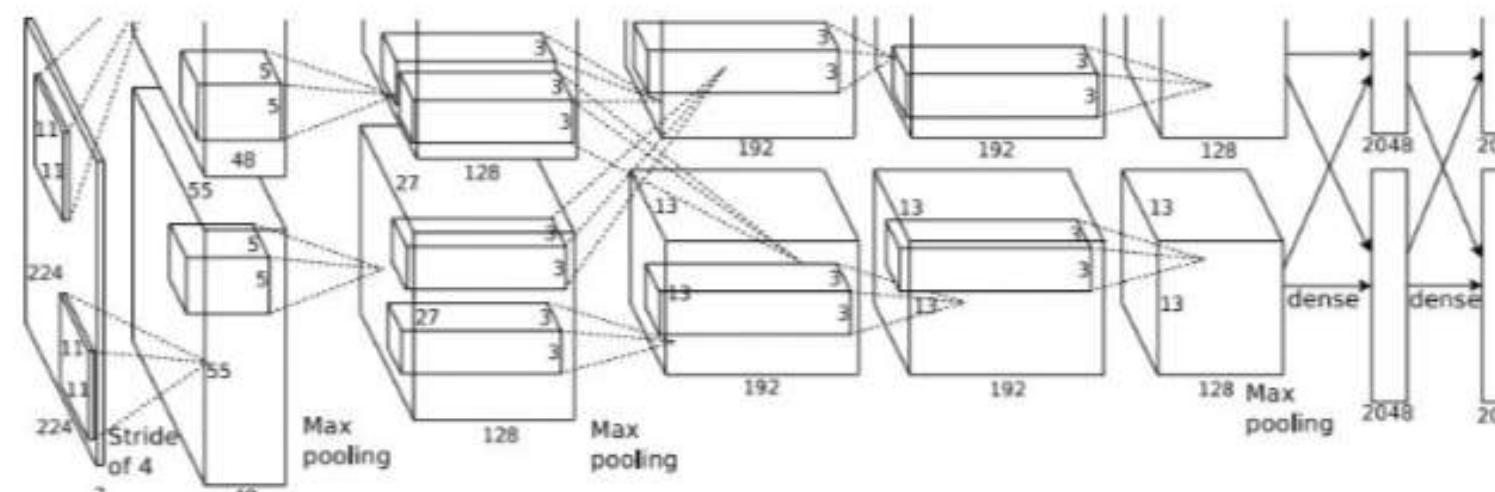
# Детекция объектов

4 выхода

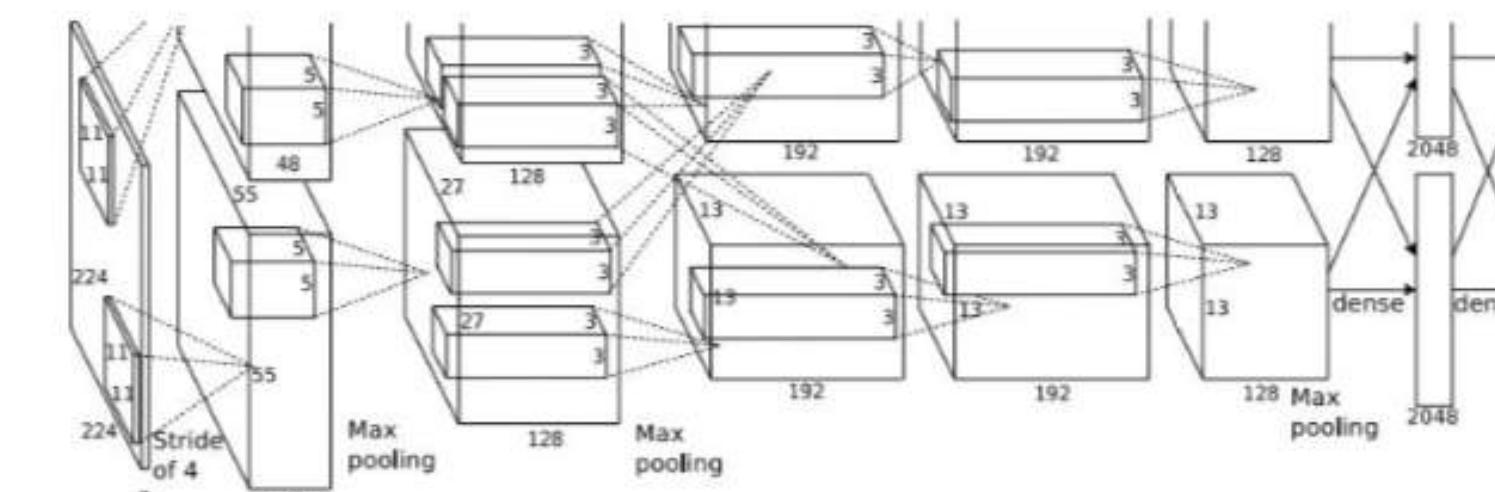


CAT:  $(x, y, w, h)$

12 выходов



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



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

Количество объектов заранее неизвестно!

cs231n slides

# Классификатор + скользящее окно

Имеем классификатор – давайте переиспользовать



**Футболист: 0.3**  
**Болельщик: 0.1**  
**Мяч: 0.1**  
**Фон: 0.8**

# Классификатор + скользящее окно

Имеем классификатор – давайте переиспользовать



**Футболист: 0.9**  
**Болельщик: 0.1**  
**Мяч: 0.1**  
**Фон: 0.2**

# Классификатор + скользящее окно

Имеем классификатор – давайте переиспользовать



**Футболист: 0.9**  
**Болельщик: 0.2**  
**Мяч: 0.1**  
**Фон: 0.2**

# Классификатор + скользящее окно

Имеем классификатор – давайте переиспользовать



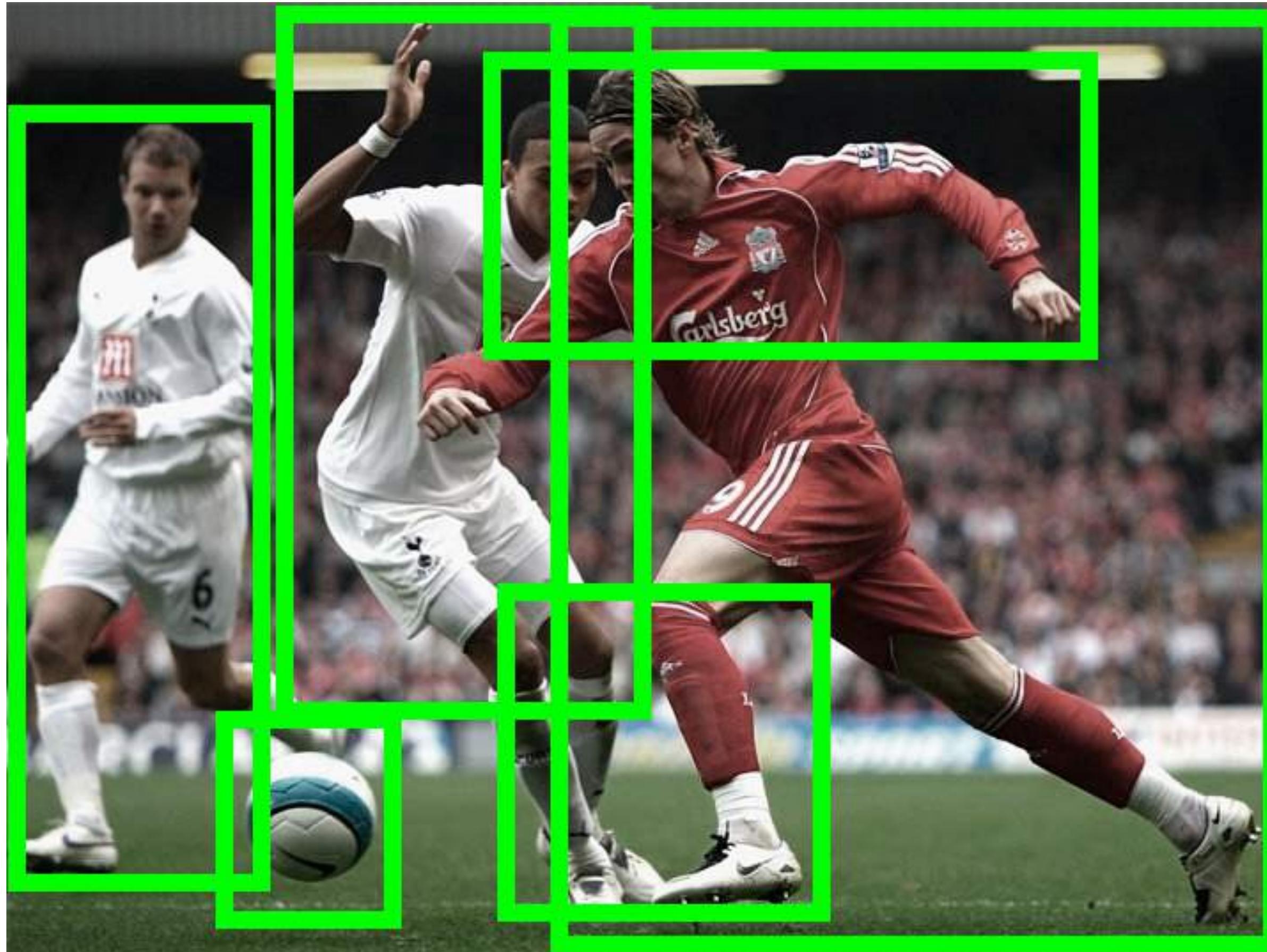
**Футболист: 0.7**  
**Болельщик: 0.2**  
**Мяч: 0.5**  
**Фон: 0.1**

# Классификатор + скользящее окно



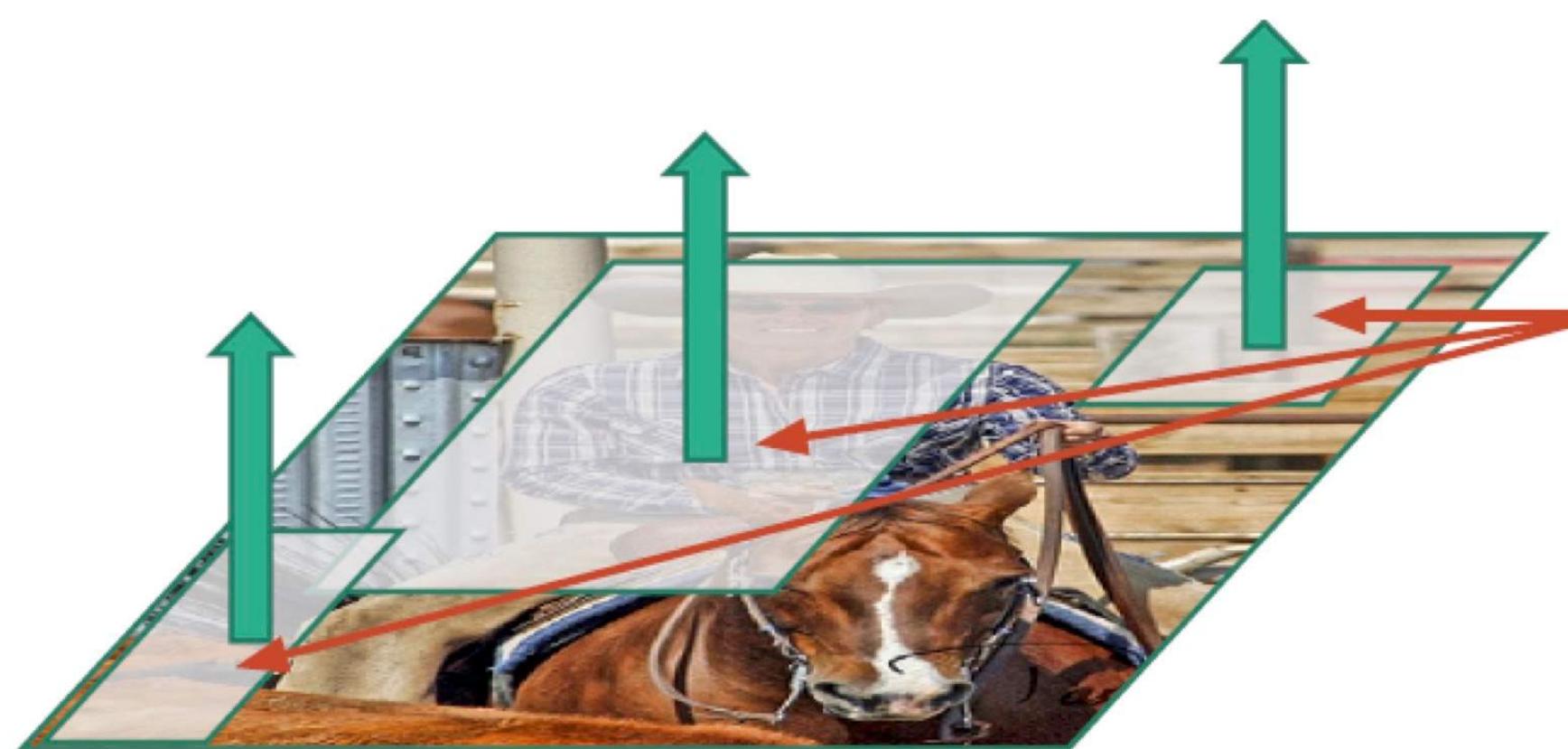
- Разные размеры и масштабы окон — вычислительно тяжело

# Region proposals



- Ищем некоторым алгоритмом зоны-кандидаты, чтобы только на них применить сеть
- Например, сгустки цветов
- ~1000 кандидатов в секунду

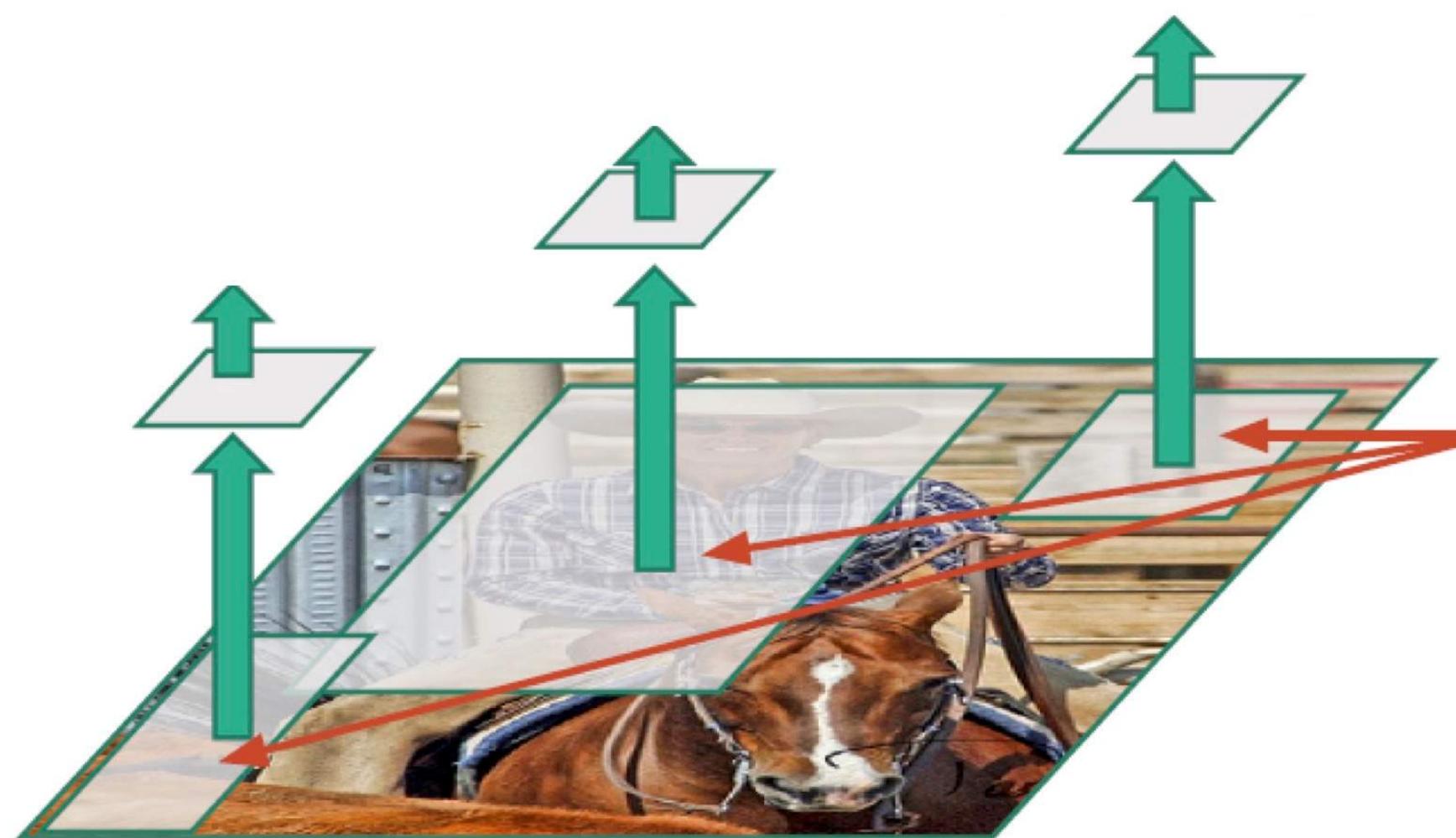
# R-CNN



## 1. Извлечение зон-кандидатов

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

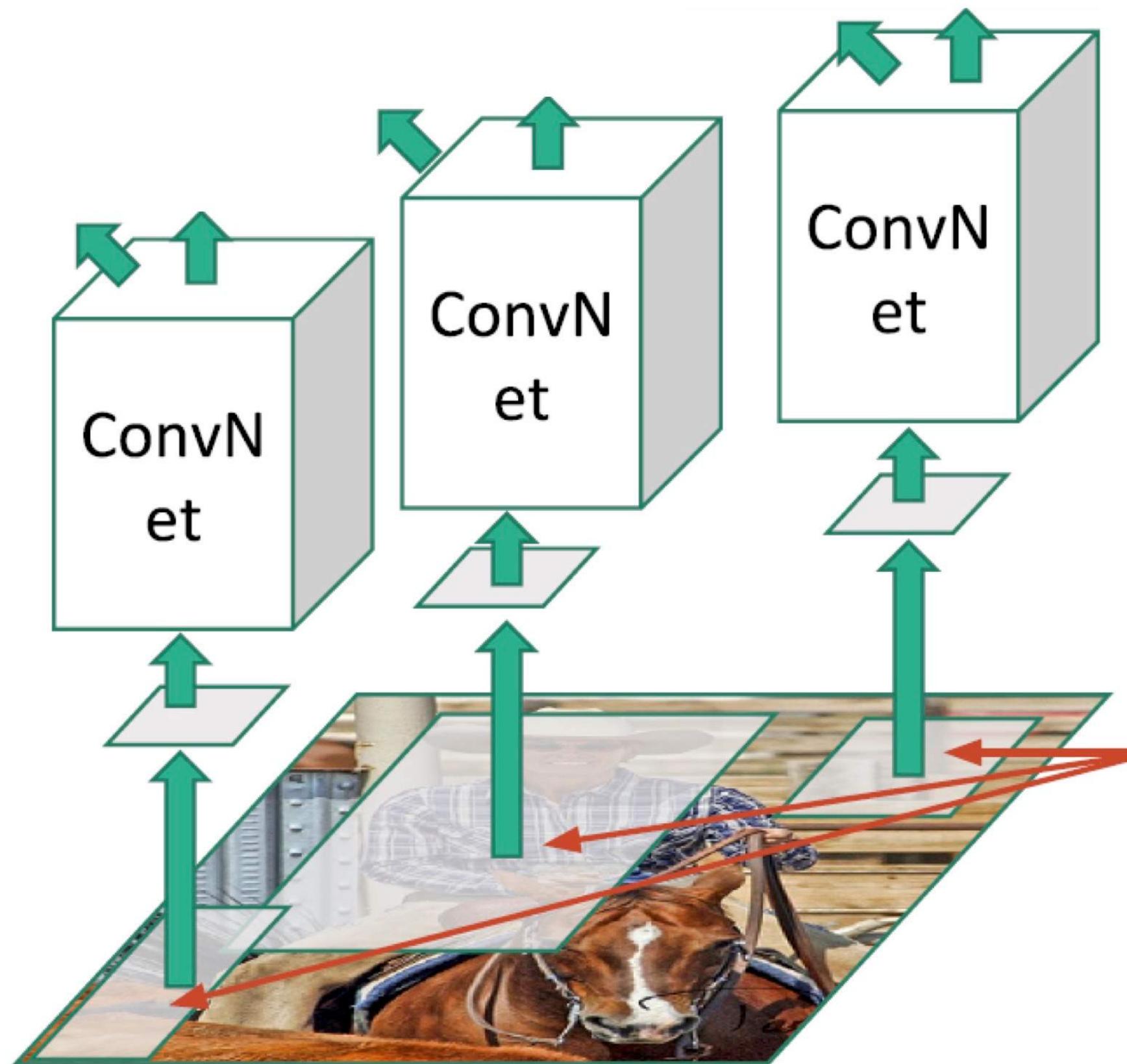
# R-CNN



1. Извлечение зон-кандидатов
2. Преобразование к одному размеру

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

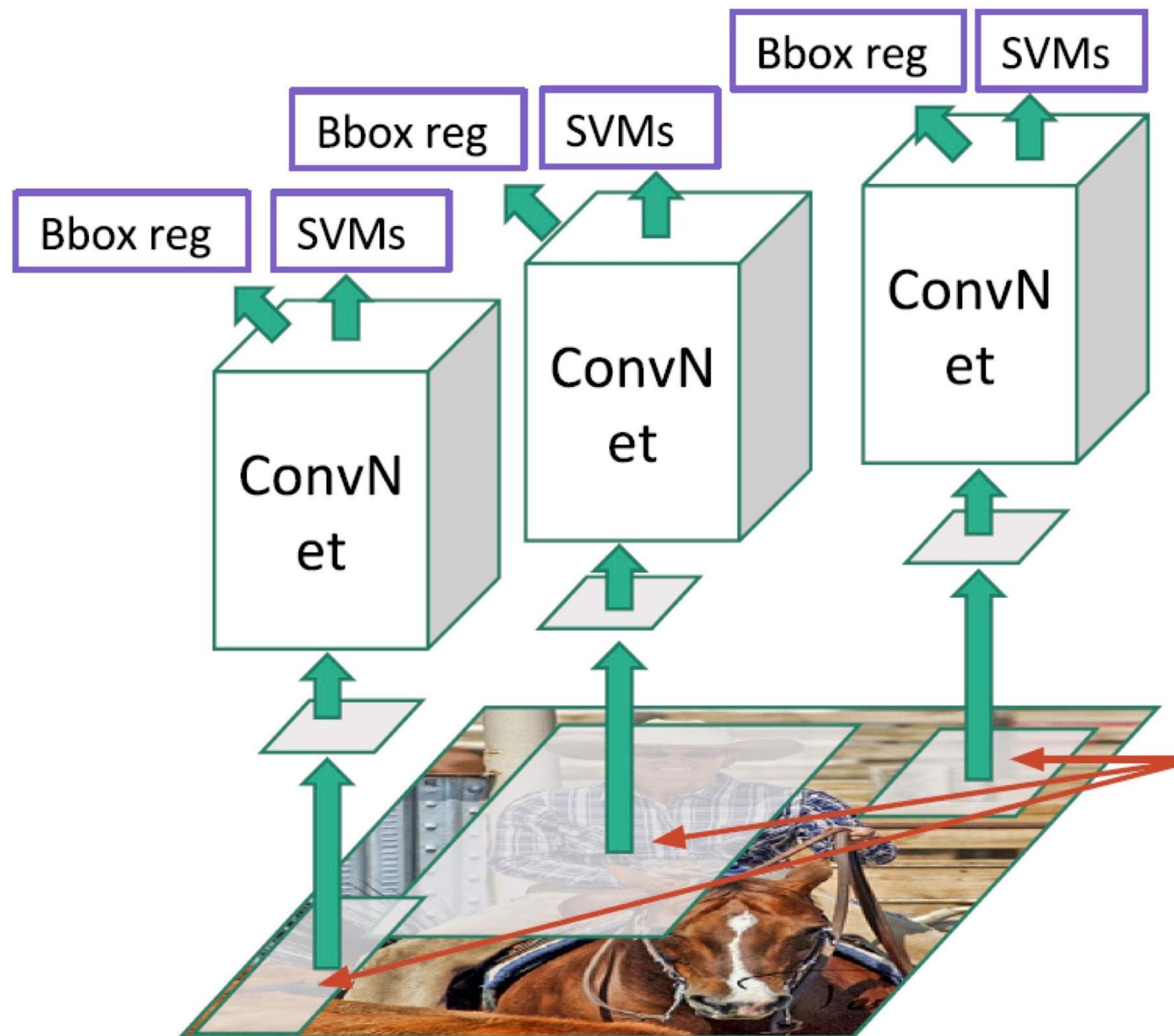
# R-CNN



1. Извлечение зон-кандидатов
2. Преобразование к одному размеру
3. Извлечение признаков сетью

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

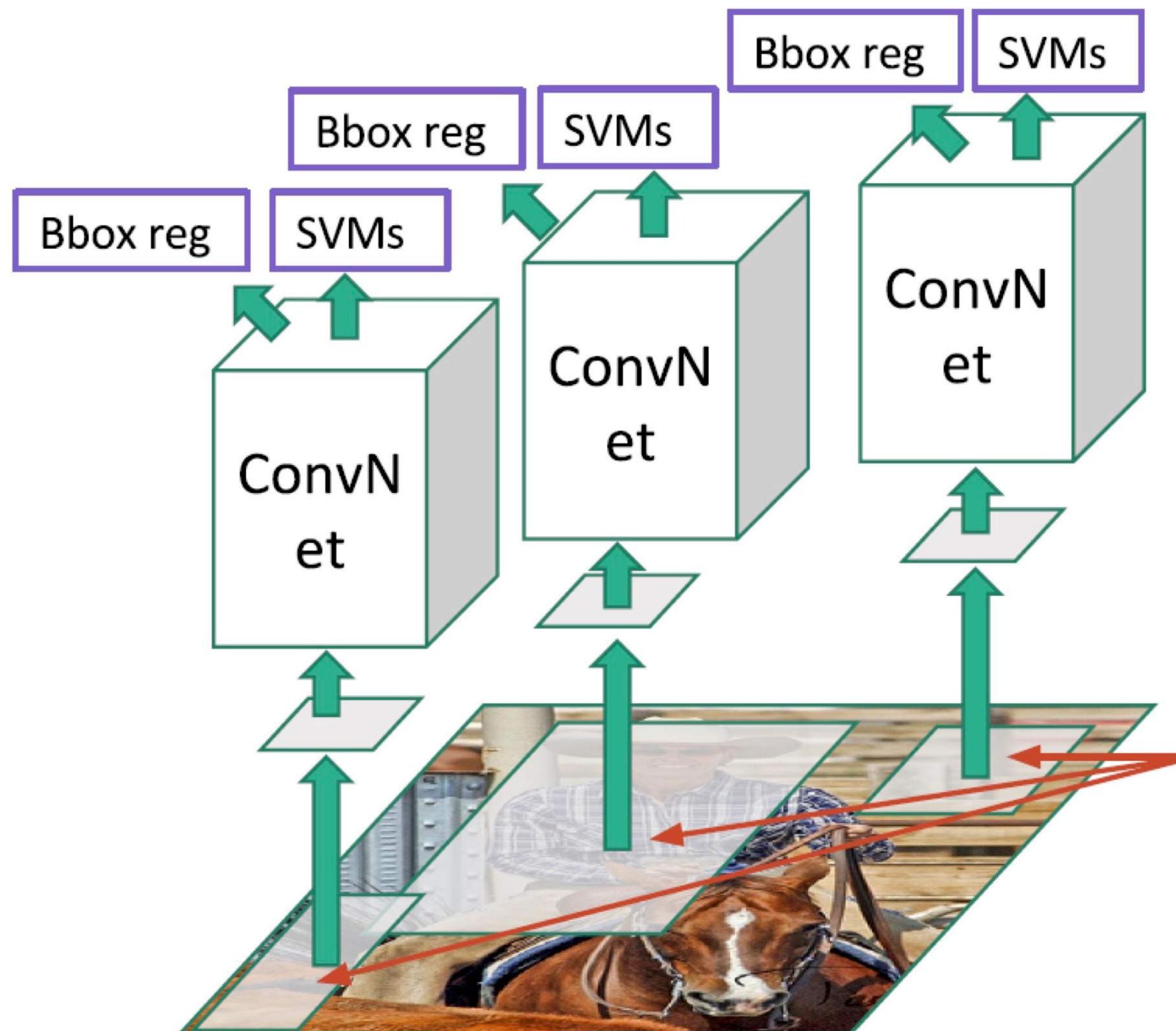
# R-CNN



1. Извлечение зон-кандидатов
2. Преобразование к одному размеру
3. Извлечение признаков сетью
4. Применение модели классификации и коррекции bbox

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

# R-CNN

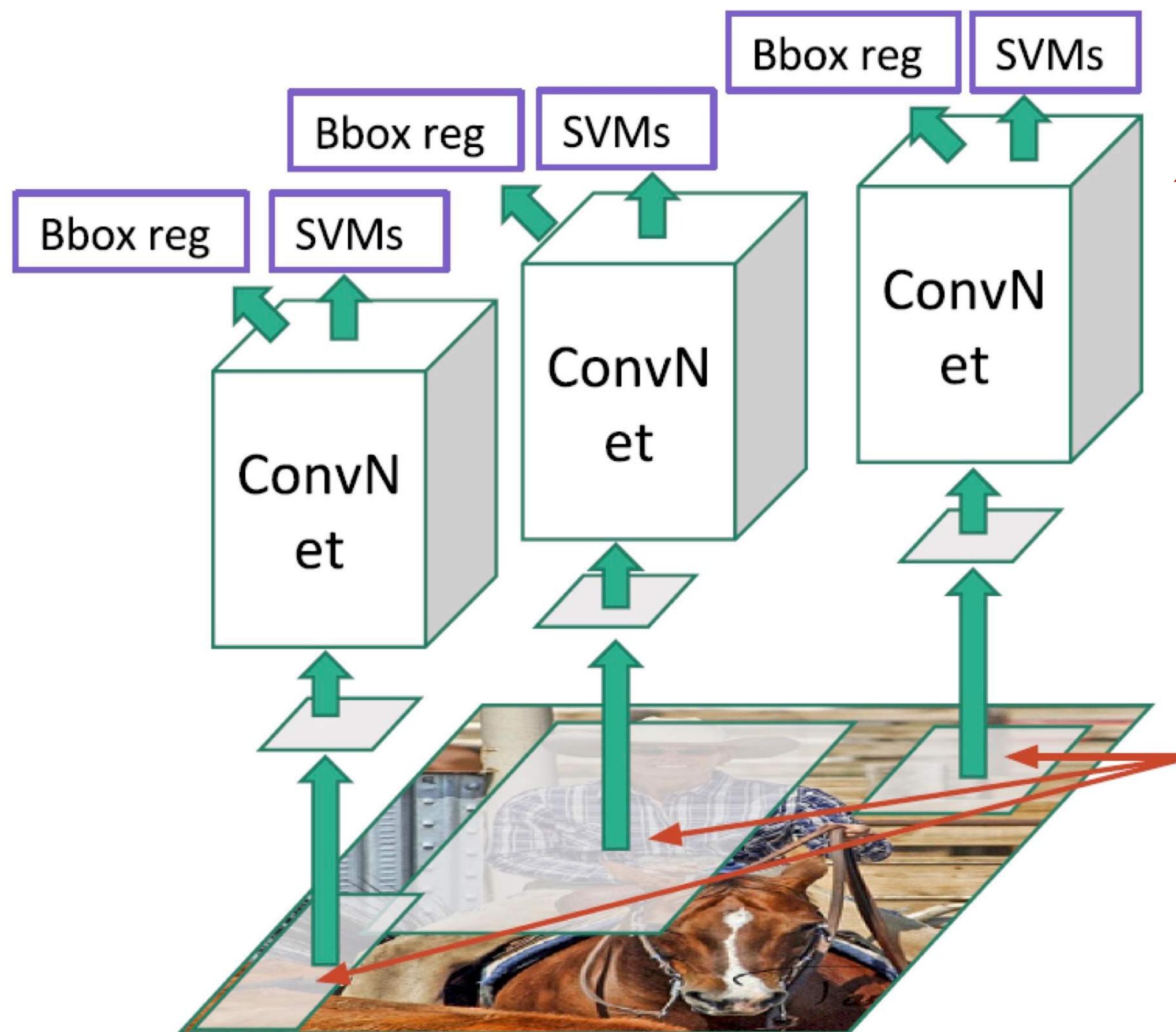


- Много независимых задач обучаются отдельно — теряем в качестве
- Долго учится, долго применяется (почти минуту для VGG16)

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

# R-CNN

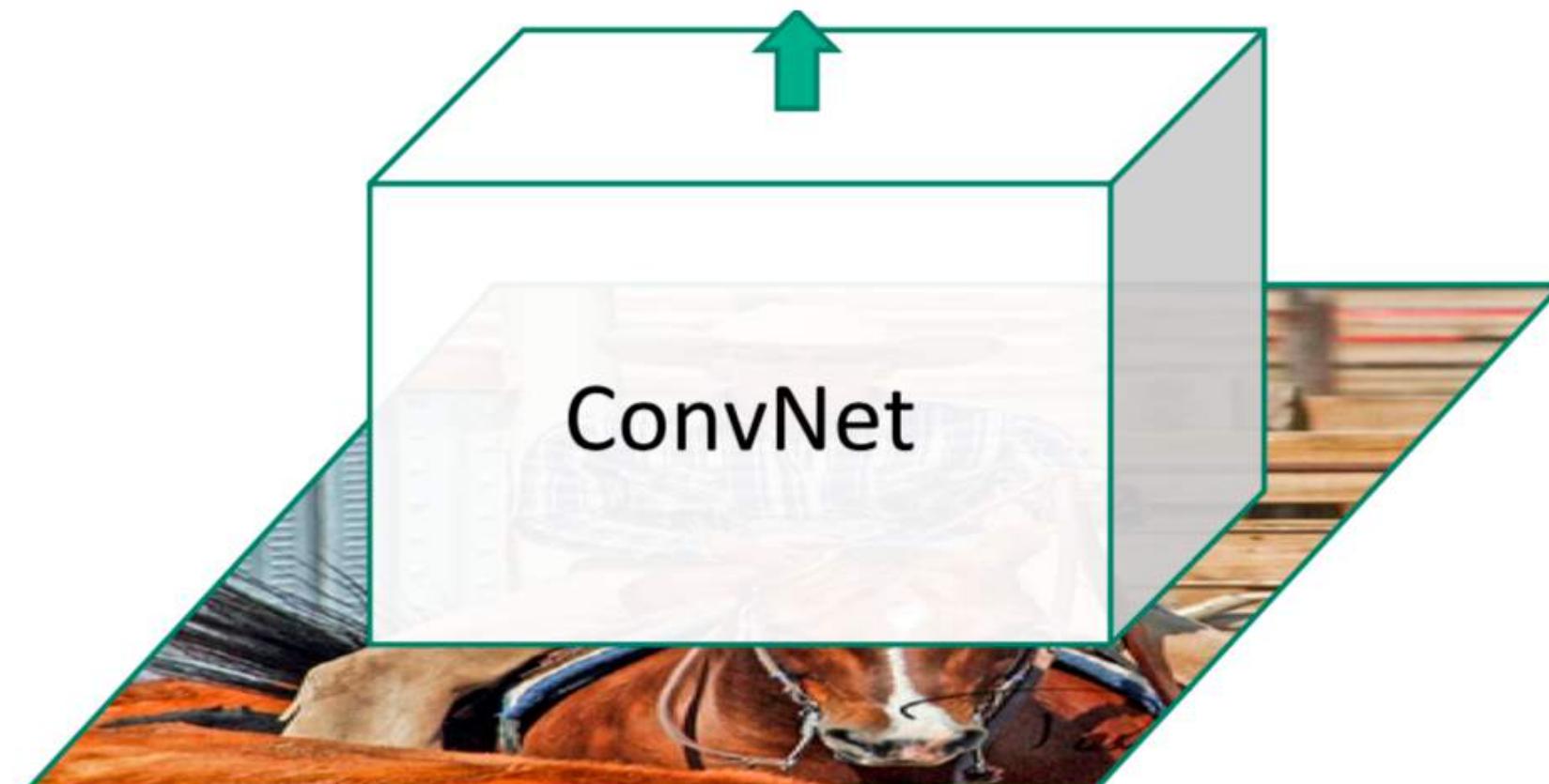
**Используем много раз одну и ту же сеть**



- Много независимых задач обучаются отдельно — теряем в качестве
- Долго учится, долго применяется (почти минуту для VGG16)

[arXiv:1311.2524 Rich feature hierarchies for accurate object detection and semantic segmentation](https://arxiv.org/abs/1311.2524)

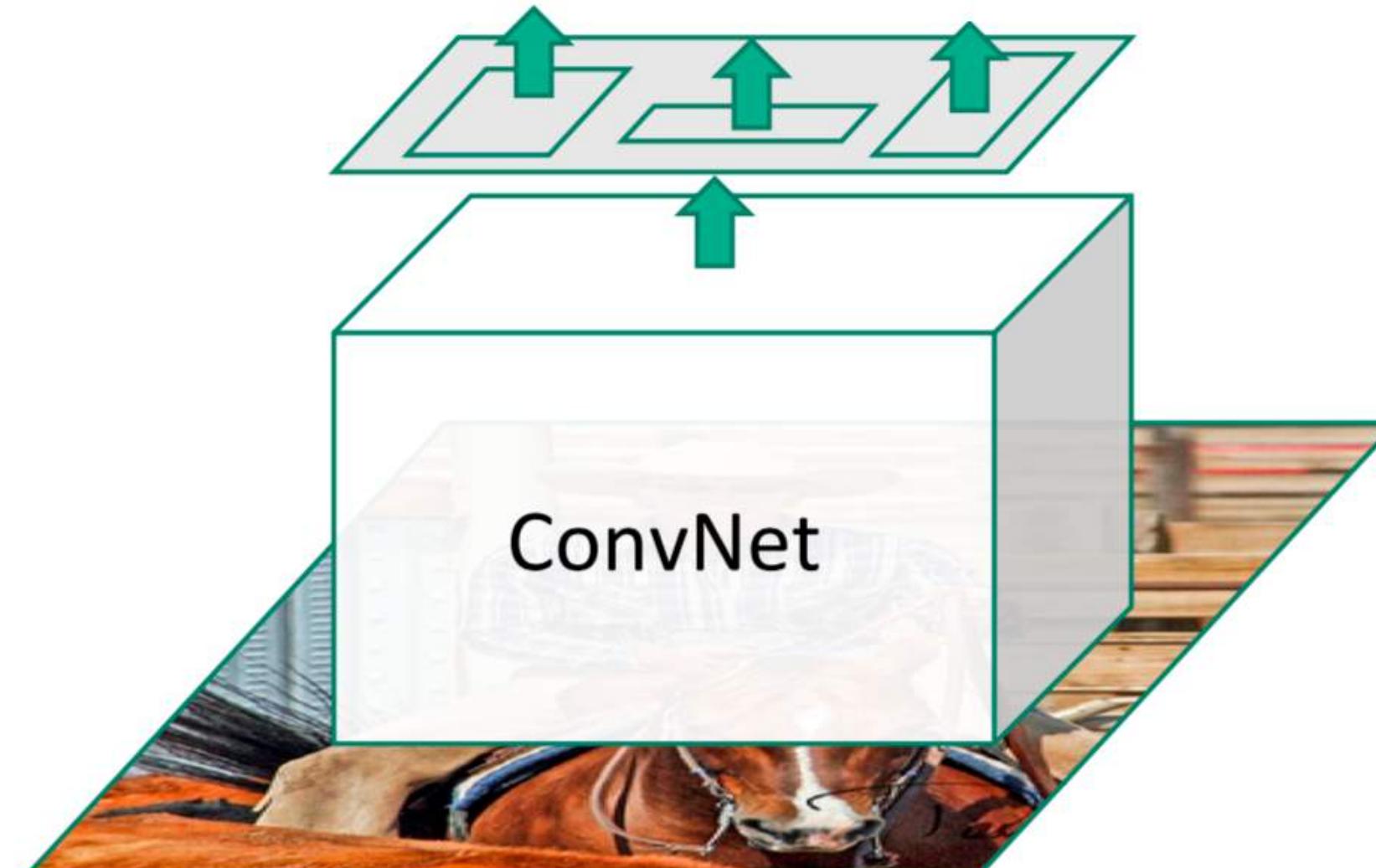
# Fast R-CNN



1. Извлечение признаков **сразу для всего изображения**

[arXiv:1504.08083 Fast R-CNN](https://arxiv.org/abs/1504.08083)

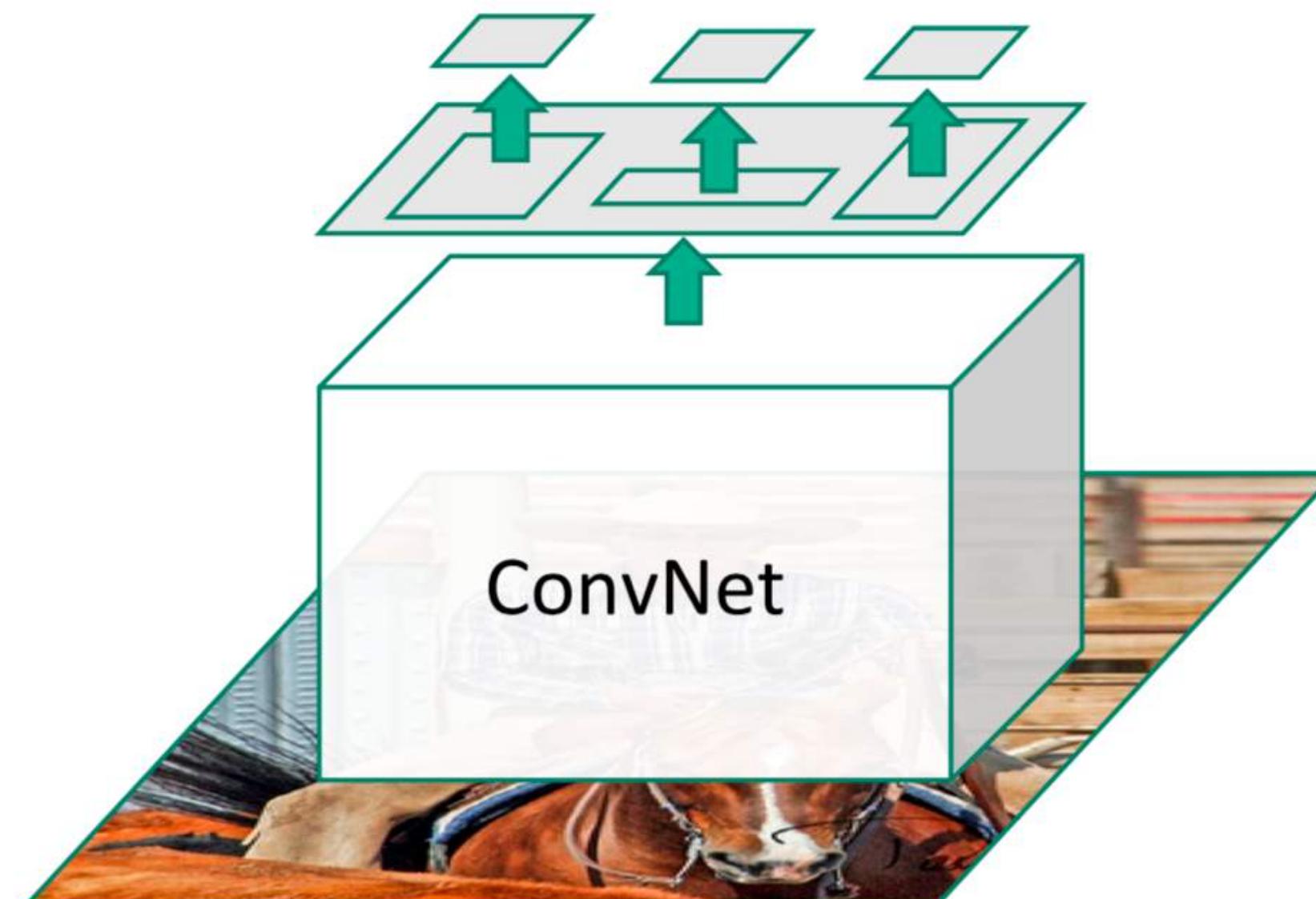
# Fast R-CNN



1. Извлечение признаков **сразу для всего изображения**
2. Вырезаем нужные области признаков

[arXiv:1504.08083 Fast R-CNN](https://arxiv.org/abs/1504.08083)

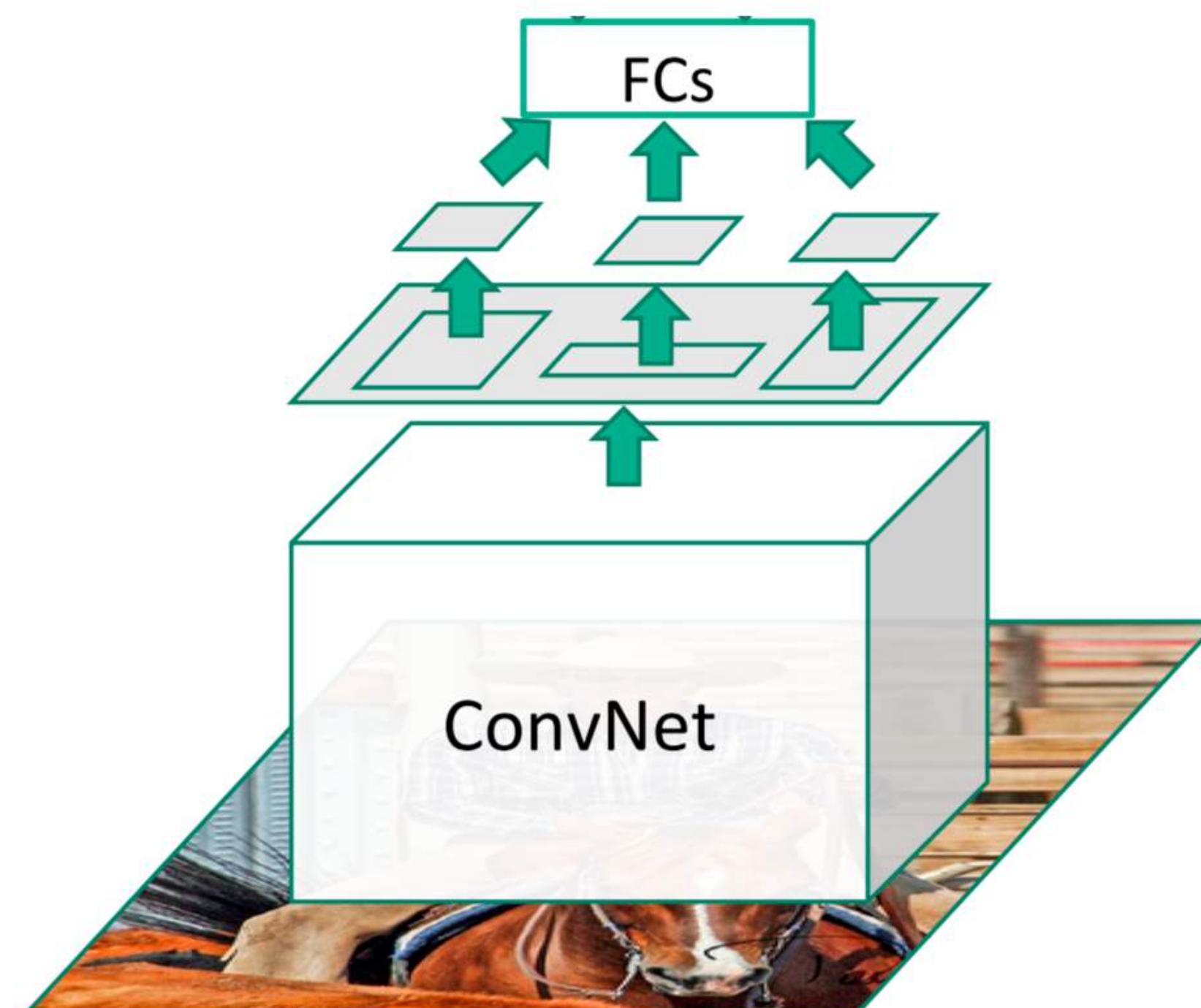
# Fast R-CNN



1. Извлечение признаков **сразу для всего изображения**
2. Вырезаем нужные области признаков
3. Преобразование признаков под фиксированную размерность

[arXiv:1504.08083 Fast R-CNN](https://arxiv.org/abs/1504.08083)

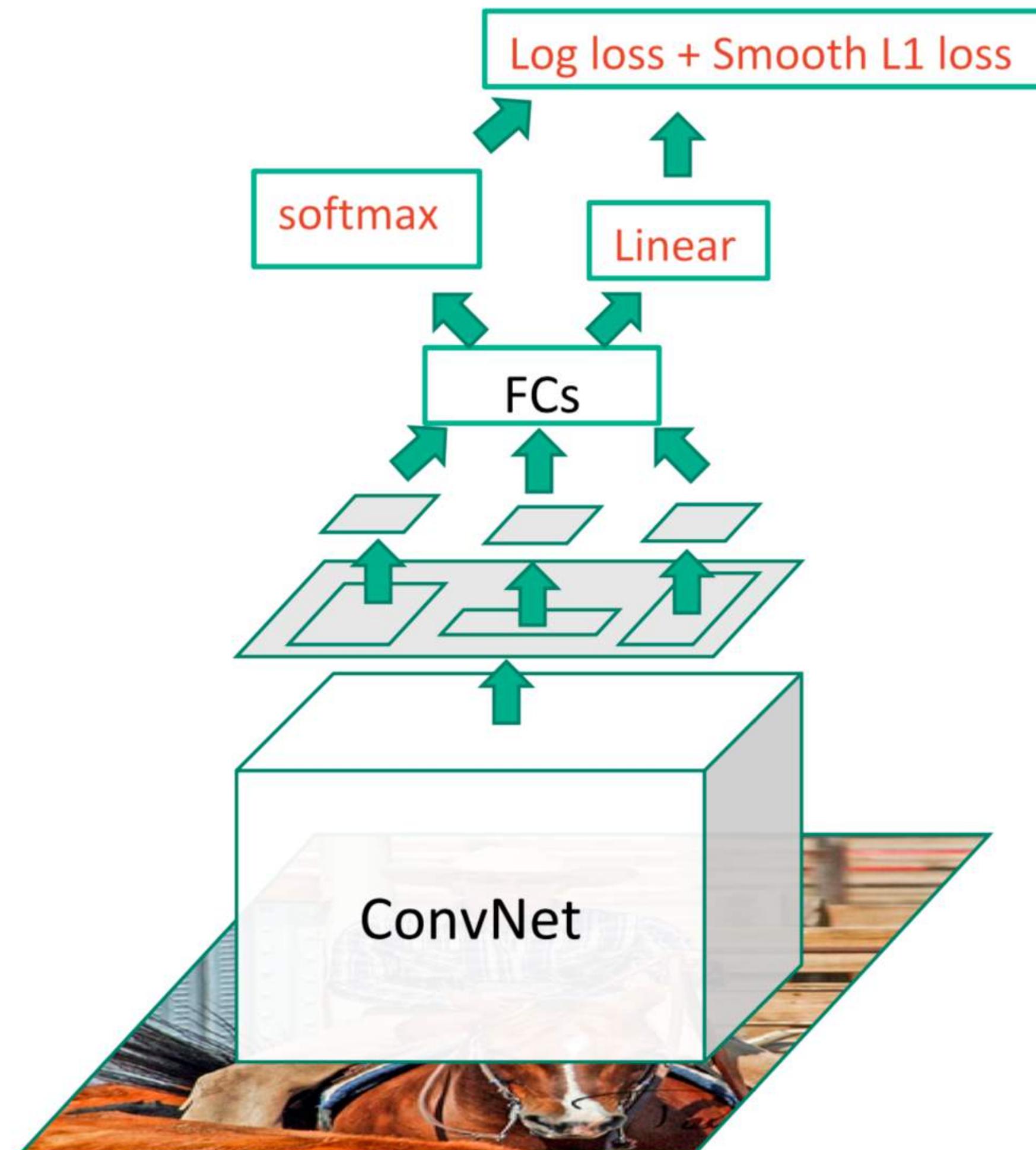
# Fast R-CNN



1. Извлечение признаков **сразу для всего изображения**
2. Вырезаем нужные области признаков
3. Преобразование признаков под фиксированную размерность
4. Прогоняем признаки через полно связные слои

[arXiv:1504.08083 Fast R-CNN](https://arxiv.org/abs/1504.08083)

# Fast R-CNN

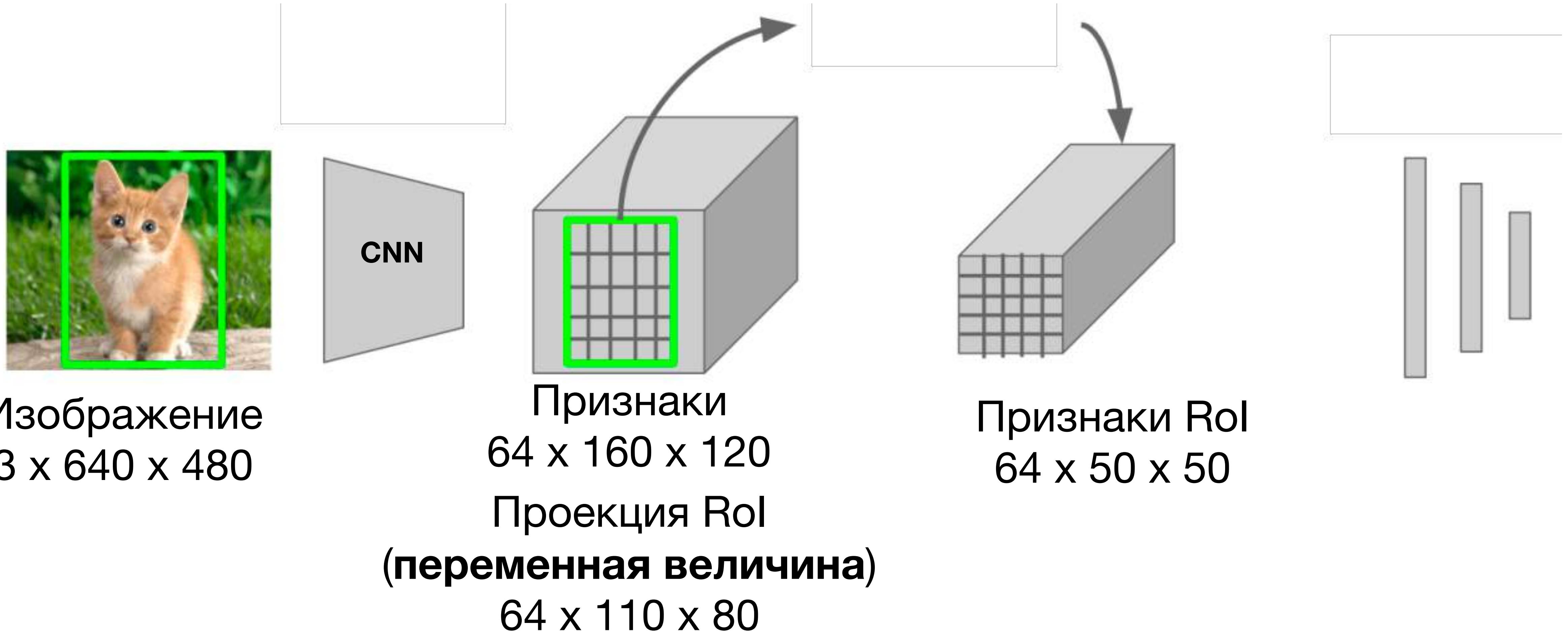


1. Извлечение признаков **сразу для всего** изображения
2. Вырезаем нужные области признаков
3. Преобразование признаков под фиксированную размерность
4. Прогоняем признаки через полно связные слои
5. Делаем предсказания класса и коррекцию bbox

[arXiv:1504.08083 Fast R-CNN](https://arxiv.org/abs/1504.08083)

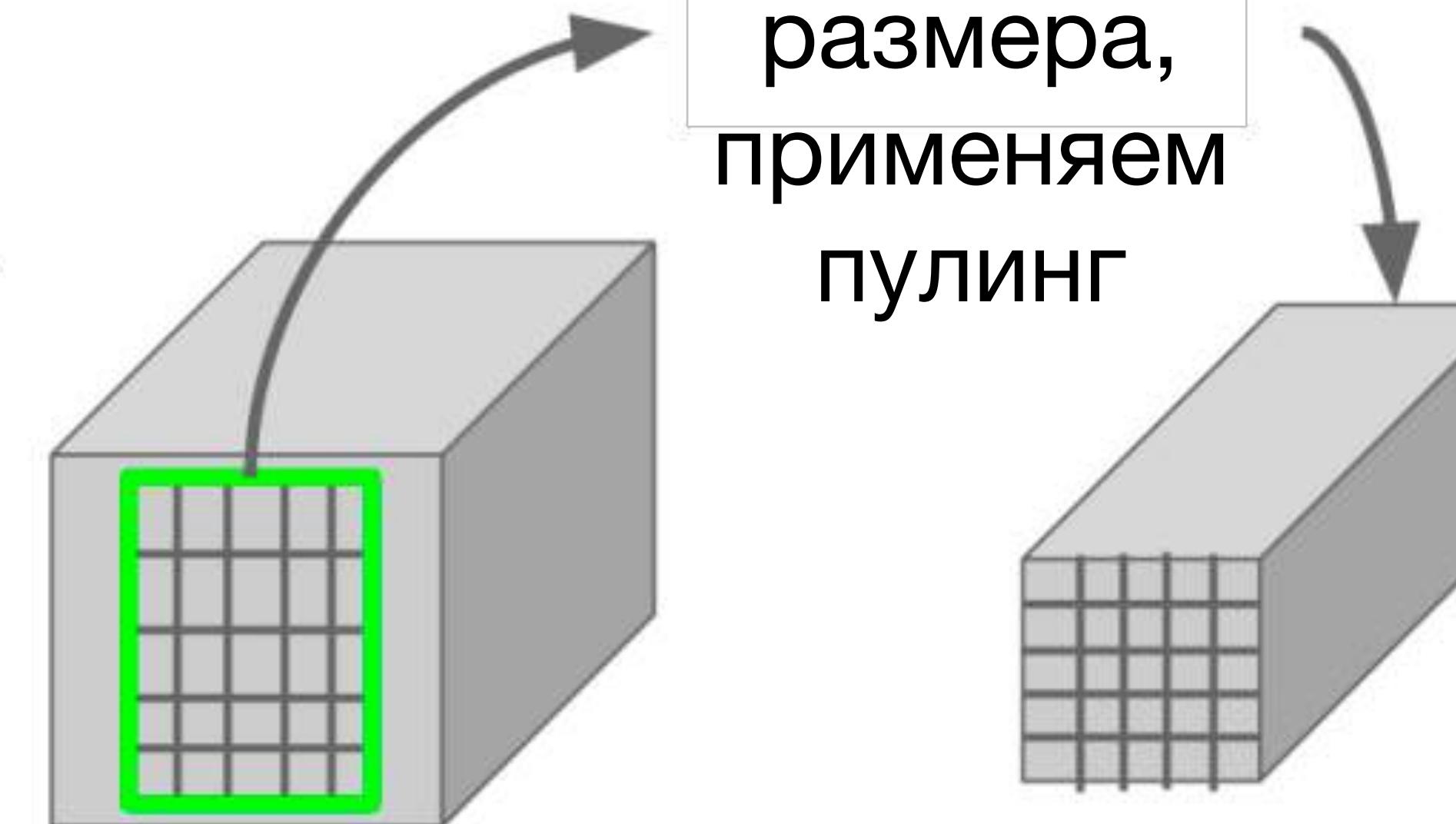
# RoI Pooling

**Проблема:** RoI разных размеров, а полносвязная сеть принимает один размер

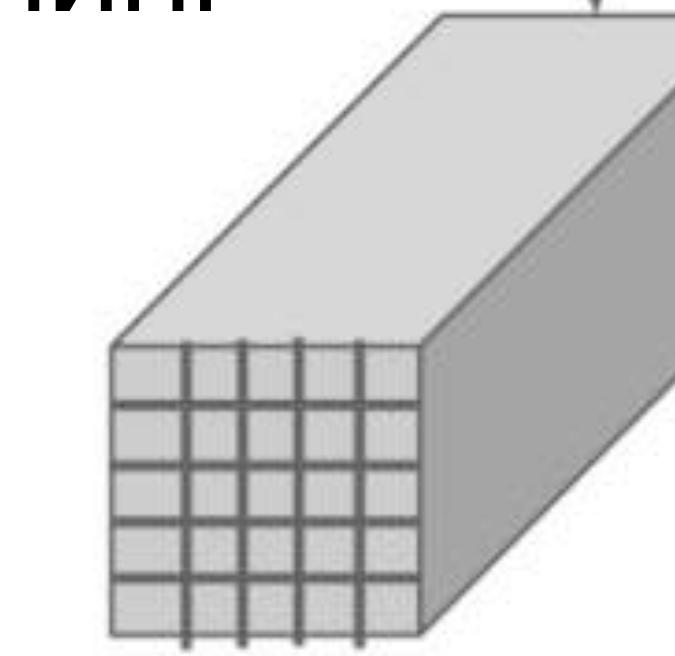


# ROI Pooling

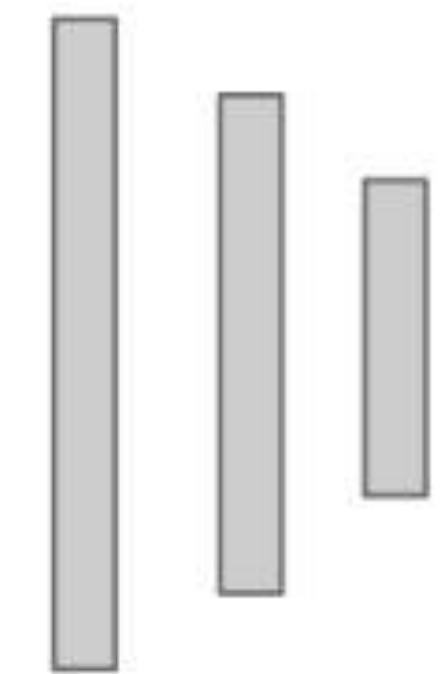
1. Извлечение признаков из изображения



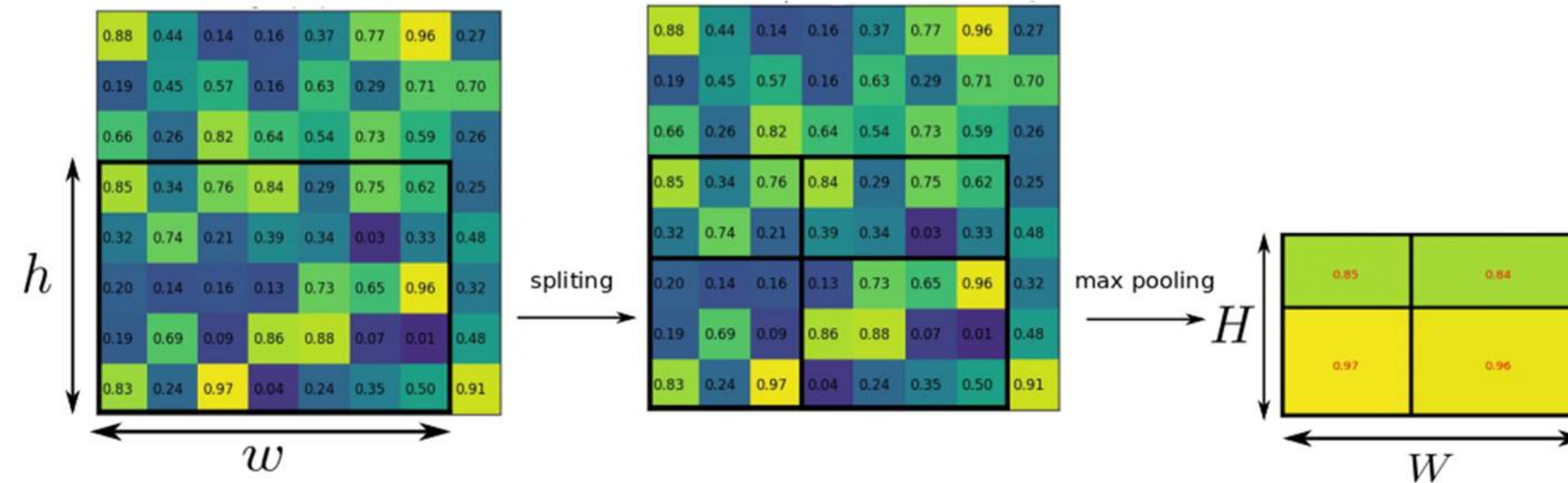
2. Делим на блоки будущего размера, применяем пулинг



3. Работаем дальше с одним размером



# Roi Pooling

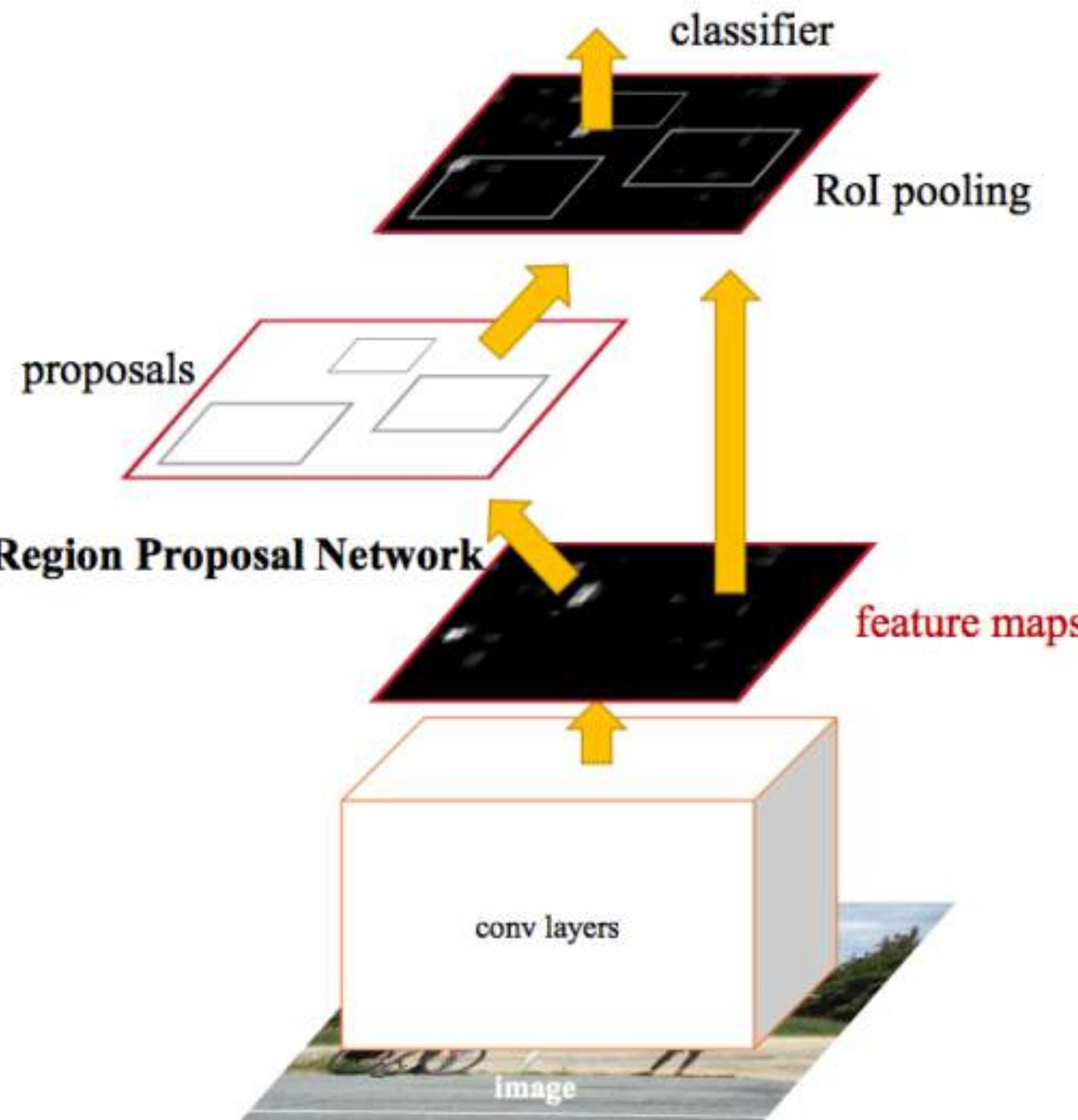


Проекция  
признаков Roi

Деление на блоки  
(количество блоков =  
нужная размерность)

Pooling

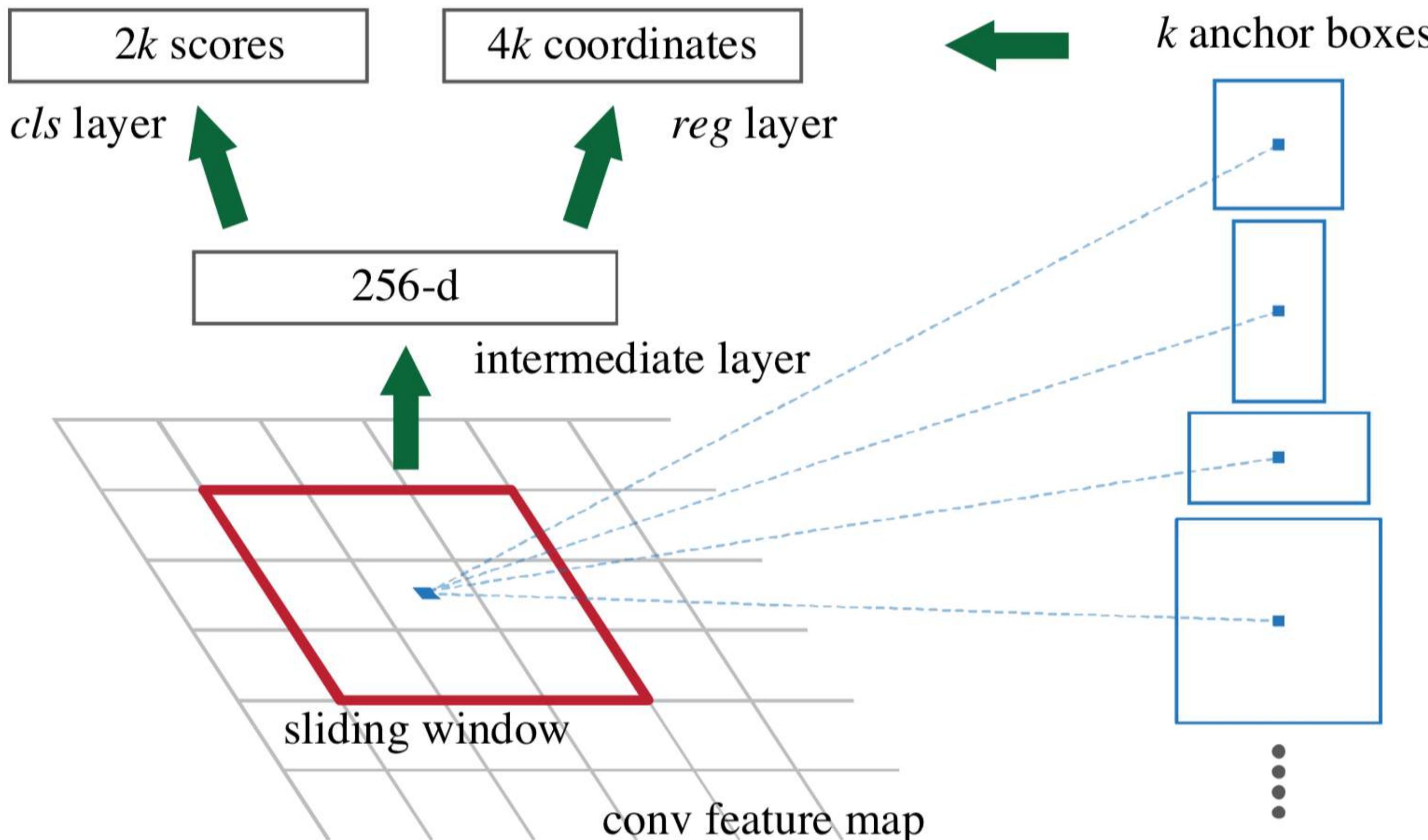
# Faster R-CNN



- Генерация RoI — последний тяжёлый рубеж
- Хотим генерировать сетью на основе извлечённых признаков

[arXiv:1506.01497 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](https://arxiv.org/abs/1506.01497)

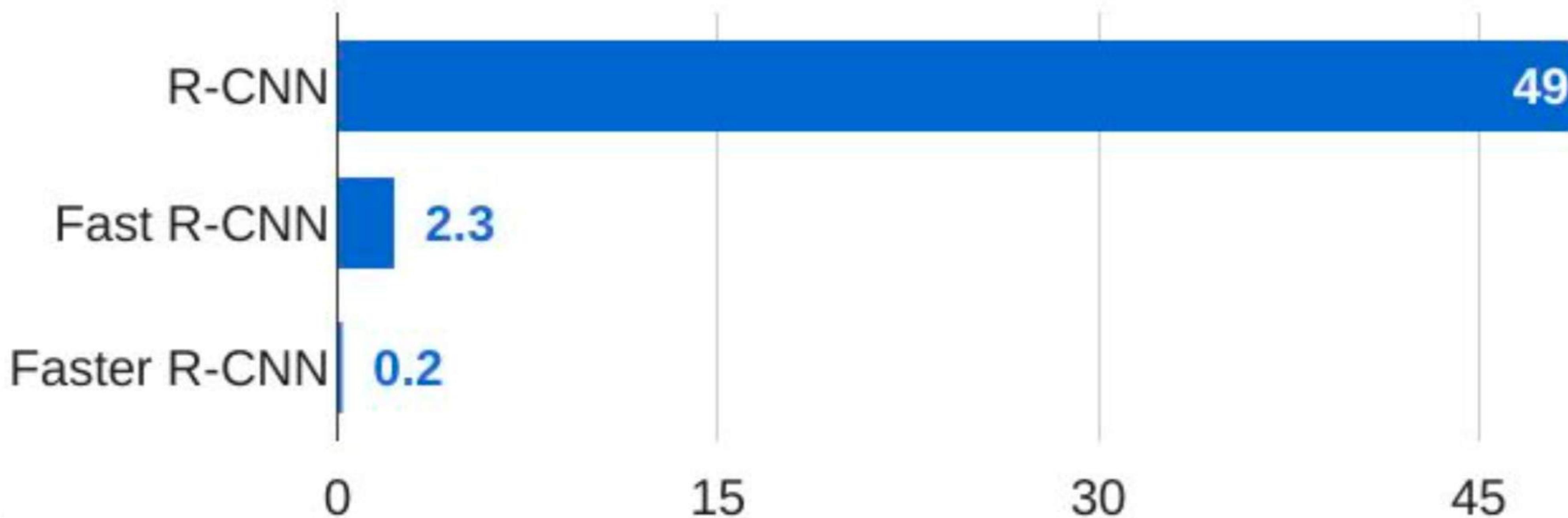
# Region proposal network



1. Небольшим скользящим окном двигаемся по признакам-активациям
2. Оцениваем шансы на объект + корректируем рамки
3. Окно разных размеров и соотношений
4. Из 17000 оставим 300 proposal

[arXiv:1506.01497 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](https://arxiv.org/abs/1506.01497)

## R-CNN Test-Time Speed

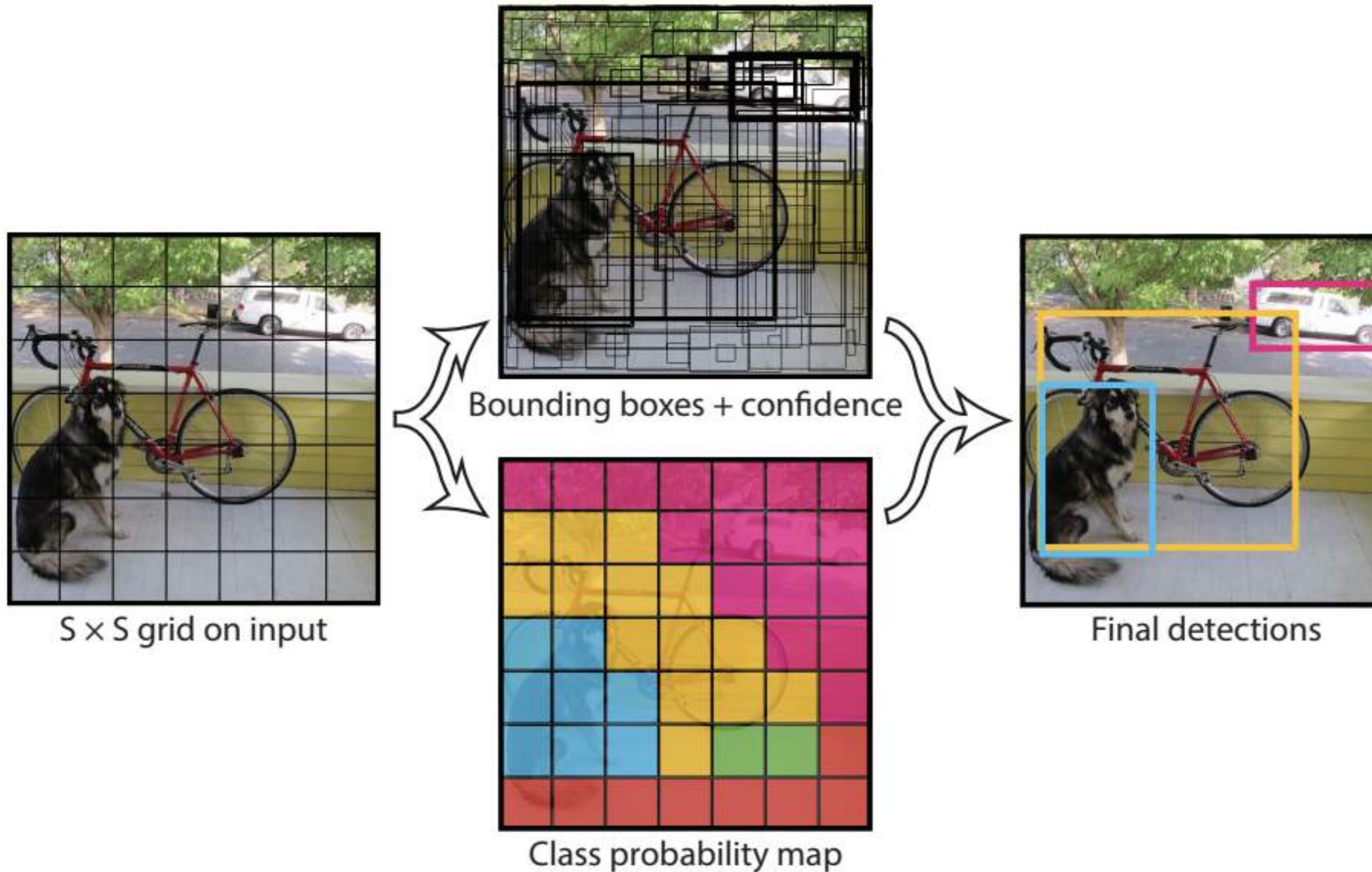


Прямой проход в секундах

# Proposal-подходы vs real-time

- Прямой проход Faster-RCNN занимает «всего» 0.2 секунды — **всего лишь 5 FPS**
- Альтернатива: детектирование **без поиска зон** одним прямым проходом сети

# YOLO: You only Look Once



- Делим изображение на блоки
- Для каждого блока предсказывается распределение классов + координаты В штук bbox с уверенностями
- Фильтруем и находим основные

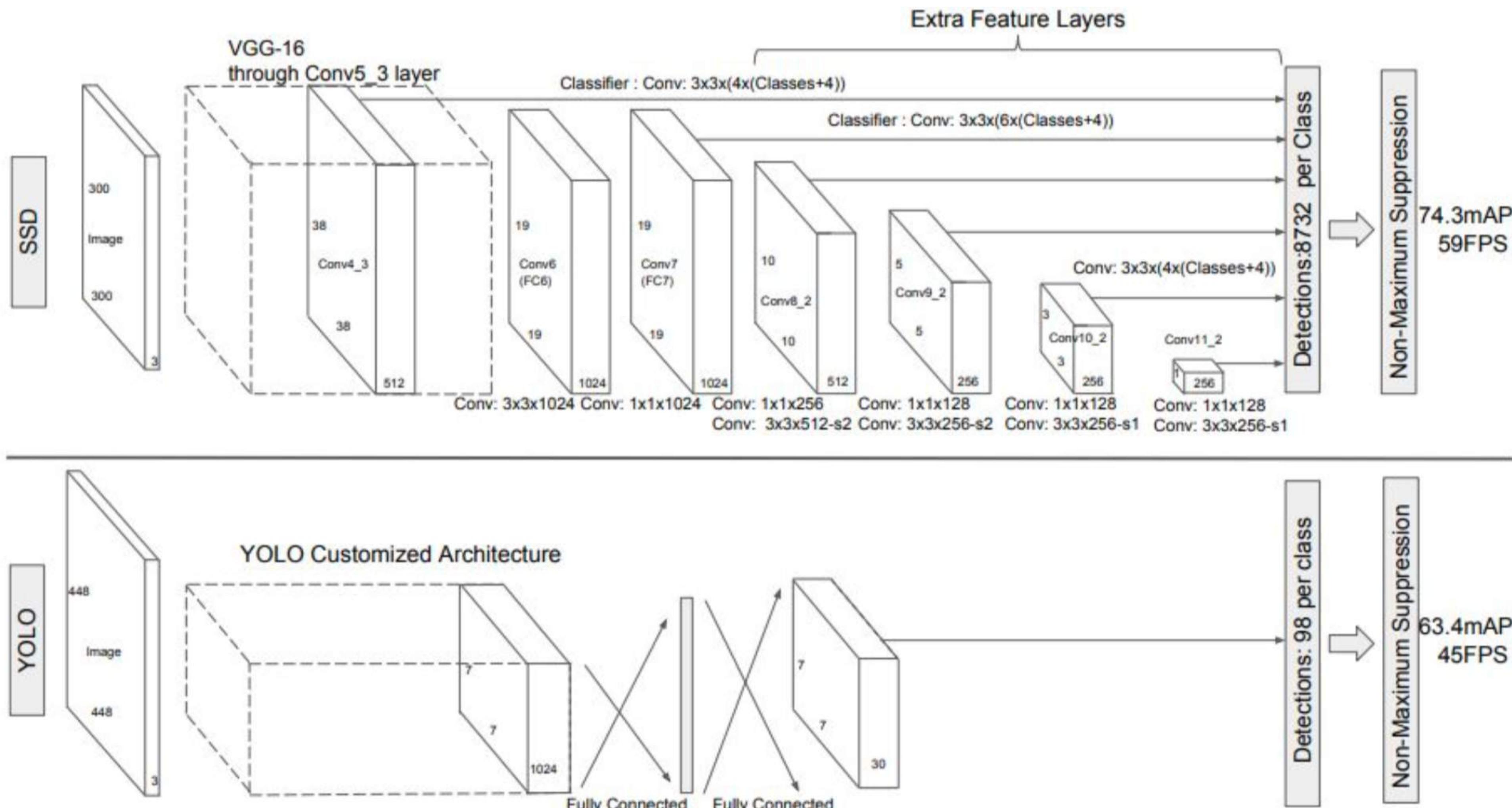
[arXiv:1506.02640 You Only Look Once: Unified, Real-Time Object Detection](https://arxiv.org/abs/1506.02640)

# SSD: The Single-Shot MultiBox Detector

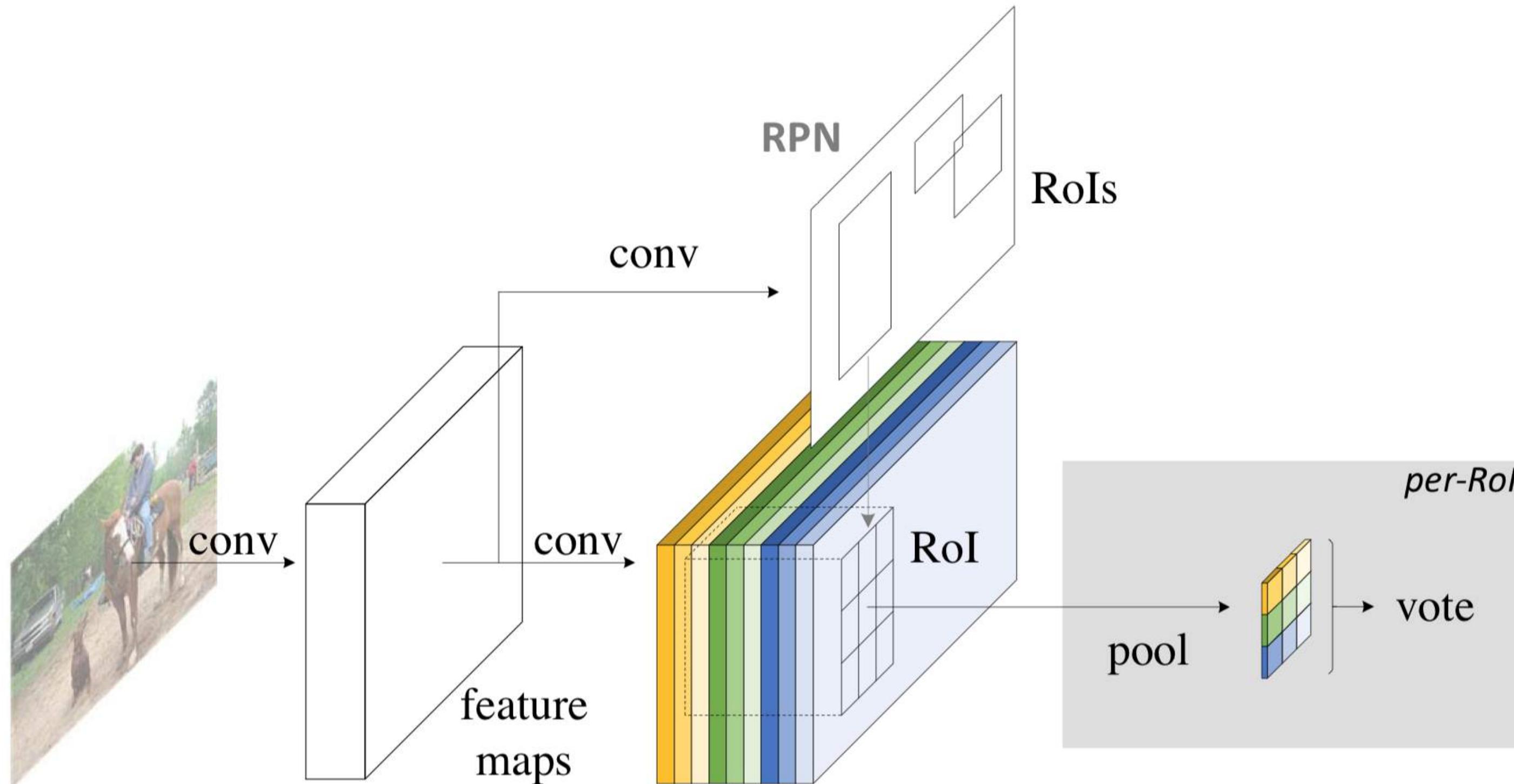
- Фиксированное количество bbox на каждый блок изображения (как у YOLO)
- Предсказание вероятностей классов для каждого bbox (не как у YOLO)
- Фичи с разных масштабов сети (не как у YOLO) — лучше находит крупные и мелкие объекты

[arXiv:1512.02325 SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325)

# SSD vs YOLO



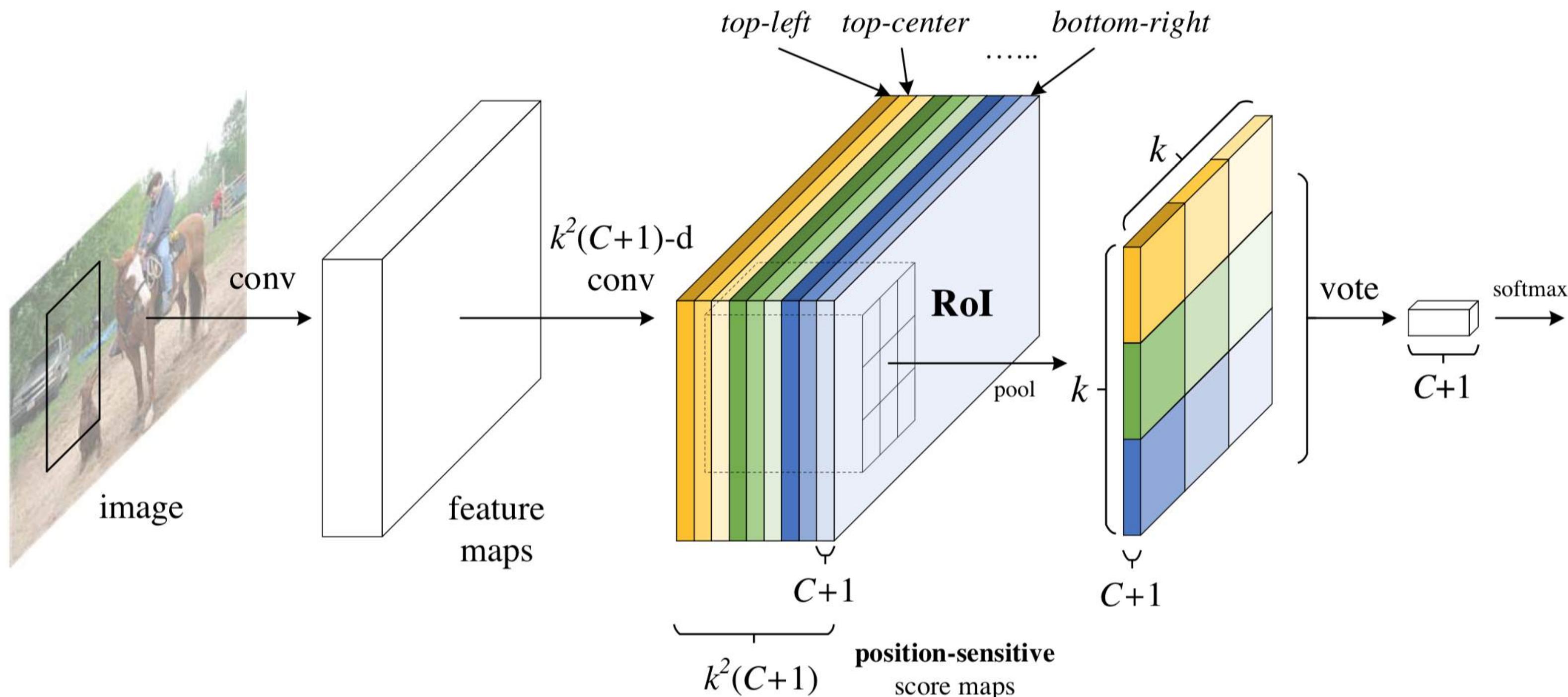
# R-FCN



- Аналогично Faster R-CNN есть RPN для получения зон-кандидатов
- Но предсказываем без полно связных слоёв из признаков – получается заметно быстрее

[arXiv:1605.06409 R-FCN: Object Detection via Region-based Fully Convolutional Networks](https://arxiv.org/abs/1605.06409)

# R-FCN



- Для каждого RoI получаем карту  $K \times K \times C$  признаков для голосования

[arXiv:1605.06409 R-FCN: Object Detection via Region-based Fully Convolutional Networks](https://arxiv.org/abs/1605.06409)

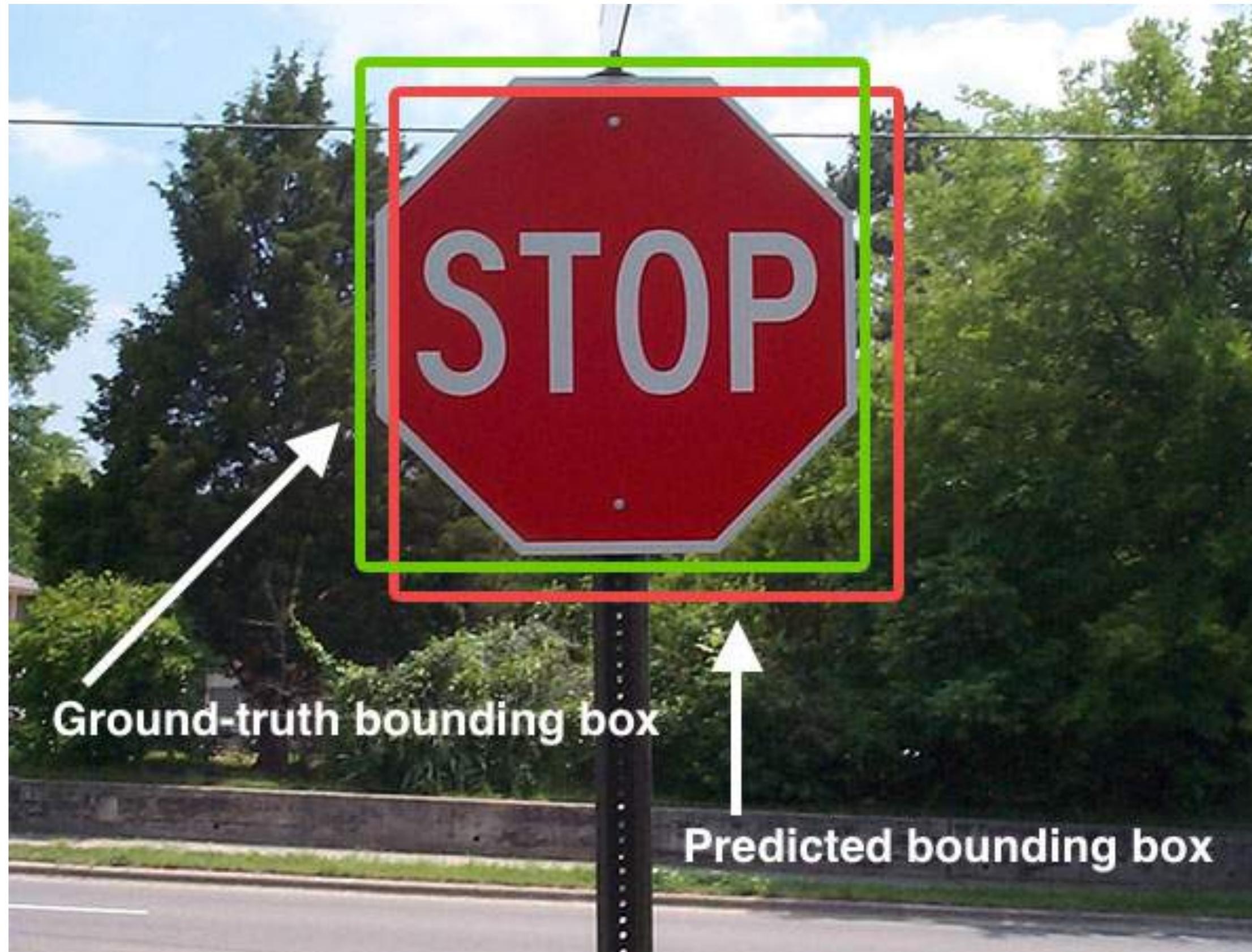
# Non-Maximum Suppression



От детектора получается много пересекающихся bbox, покрывающие одни и те же объекты:

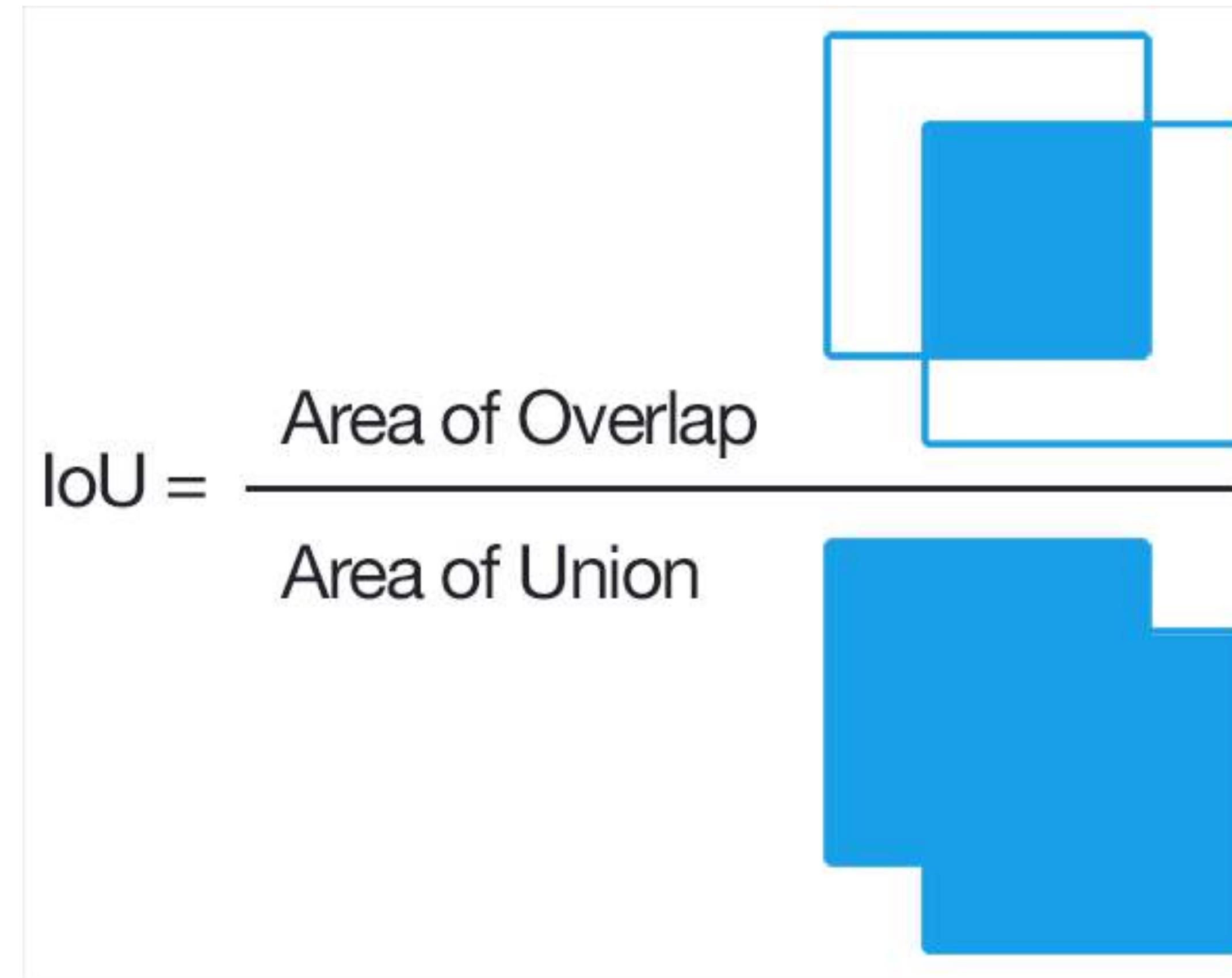
1. Фильтруем по убыванию вероятностей
2. Идём по порядку и удаляем все прочие bbox, пересекающие с текущим на  $>50\%$

# Метрики оценки



**Как сравнивать совпадение предсказанных bbox с реальными?**

# Метрики оценки



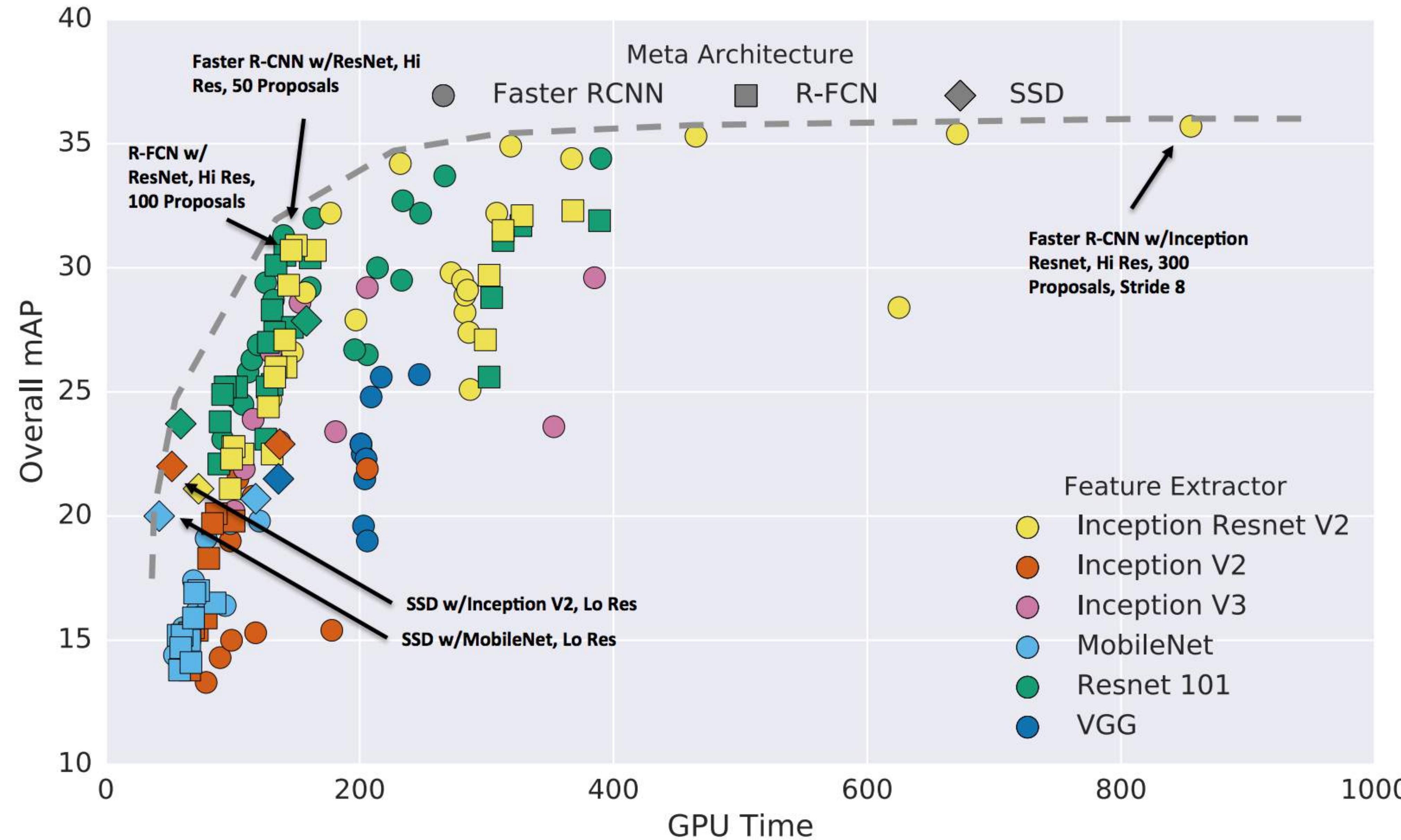
IoU (Intersection Over Union) – доля пересечения двух bbox  
(соответствующего таргета и предсказания)

# Метрики оценки

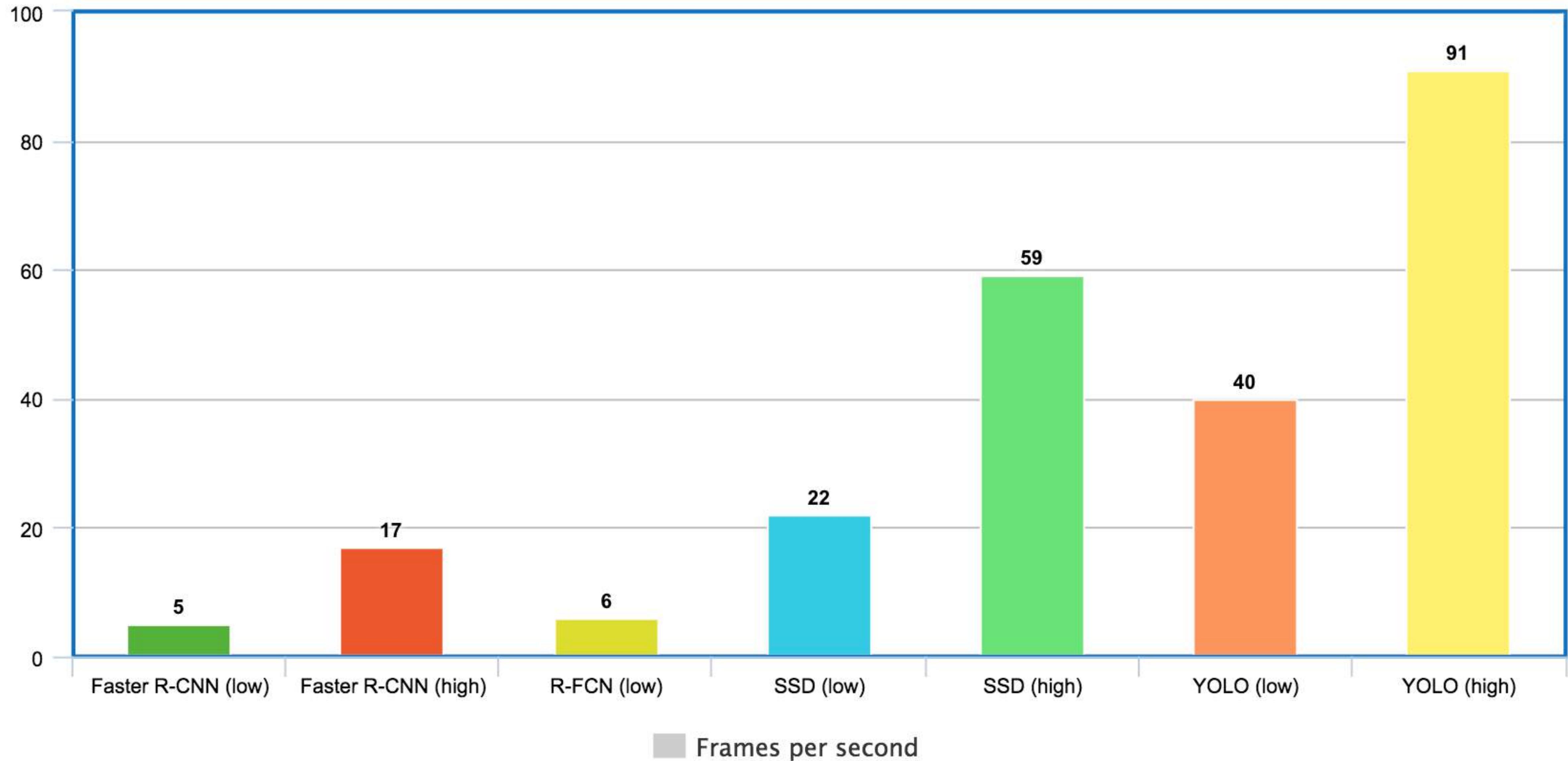
- IoU (Intersection Over Union) – доля пересечения двух bbox (соответствующего таргета и предсказания)
- Сравнение IoU с некоторым порогом (0.5..0.95) показывает **корректность детекции** (рамка вряд ли совпадёт в точности, но должна совпасть хоть как-то)
- Сводим оценку качества к оценке качества классификации (бинарной или многоклассовой)

# Метрики оценки

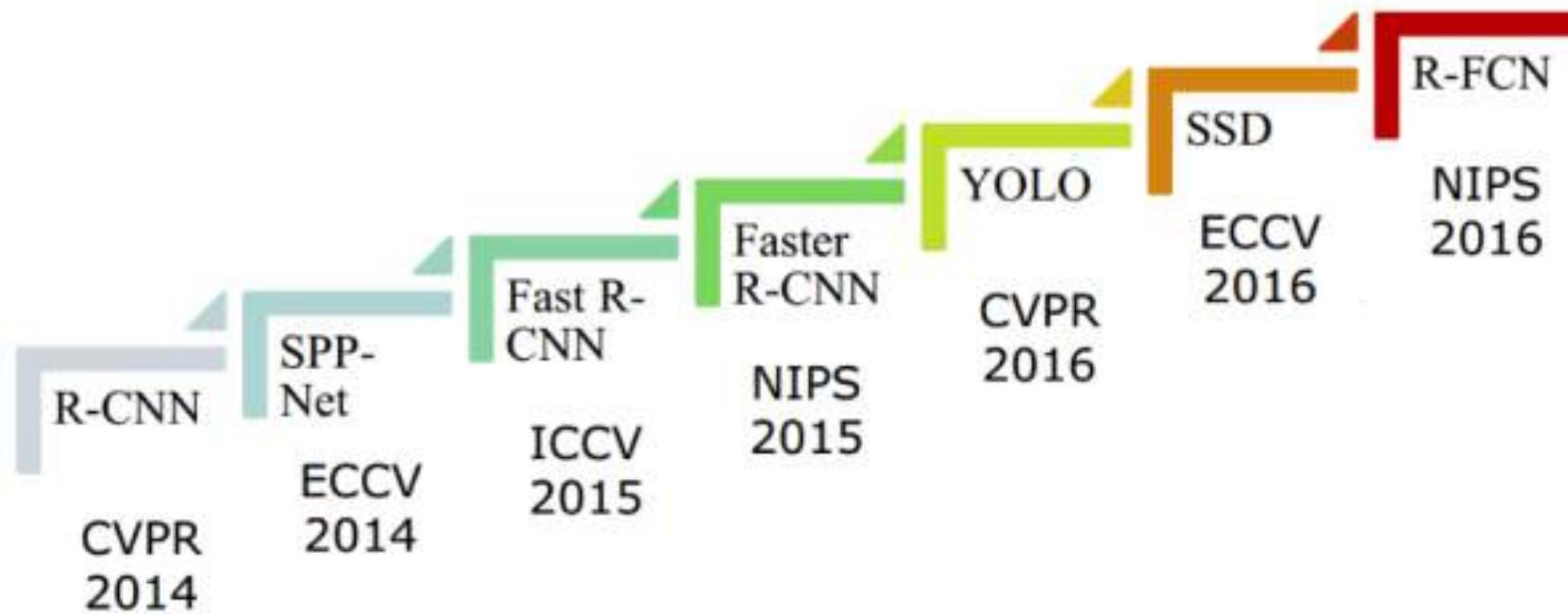
- Средний PR-AUC (average precision) по классам – mAP
- Иногда 11-point interpolation (средний precision по 11 значениям recall) – тоже mAP
- mAP усредняют также по различным порогам для IoU



Speed/accuracy trade-offs for modern convolutional object detectors



# Finally

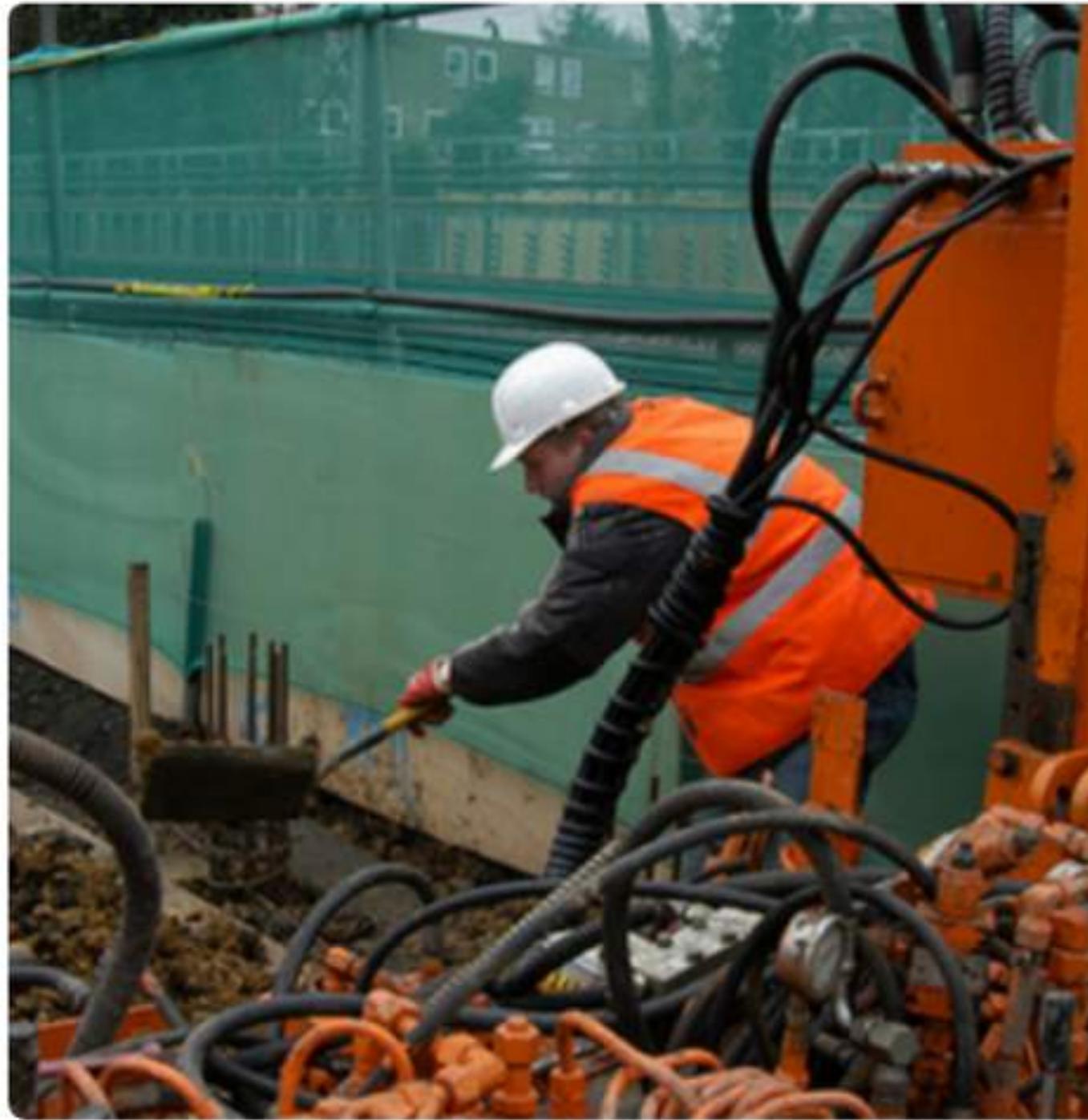


- Наилучшее качество: Faster R-CNN с тяжёлым экстрактором (например, Inception Resnet V2)
- Скорость/real-time: YOLO или SSD с лёгким экстрактором (например, MobileNet)

# Image captioning



"man in black shirt is playing guitar."

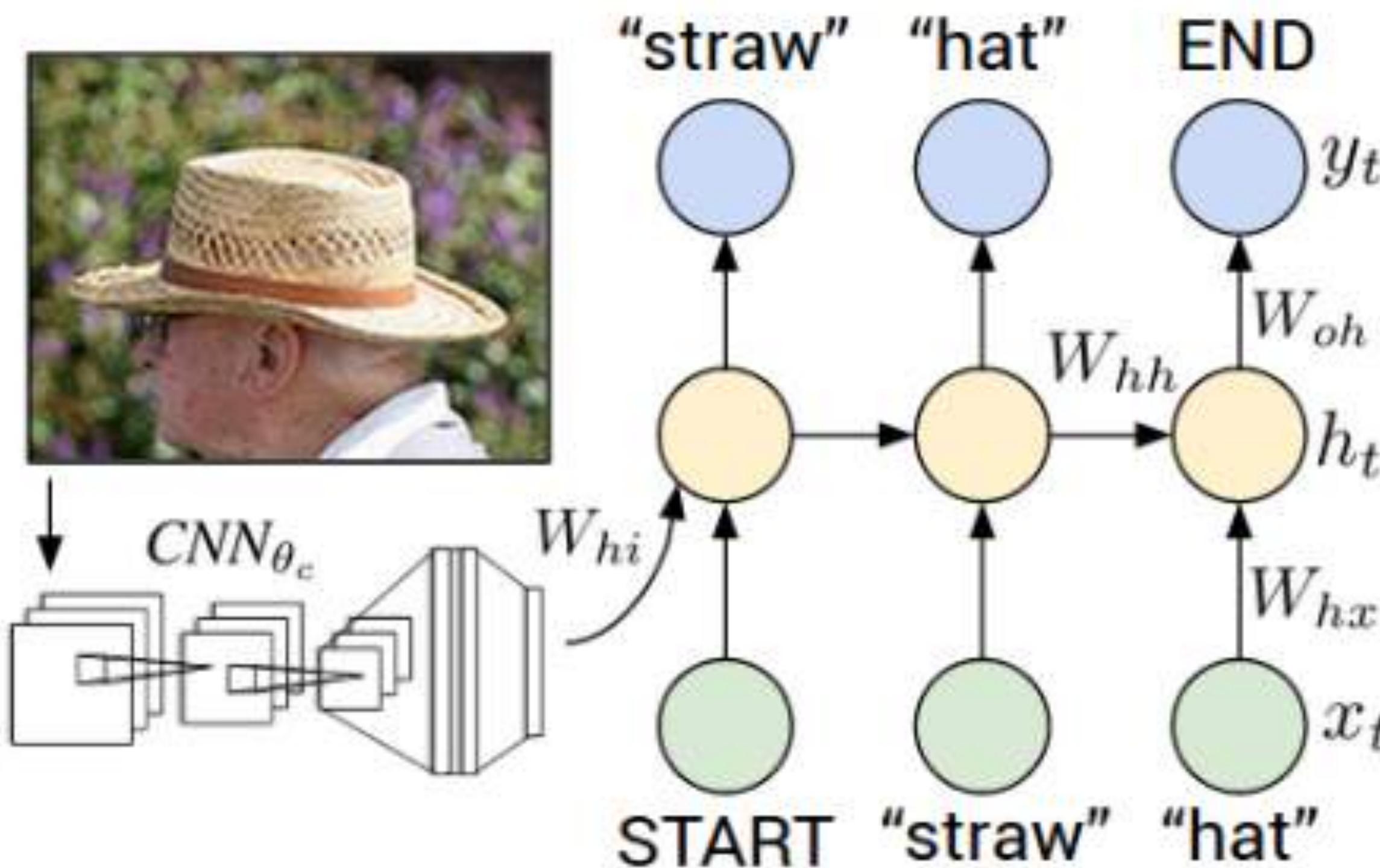


"construction worker in orange safety vest is working on road."



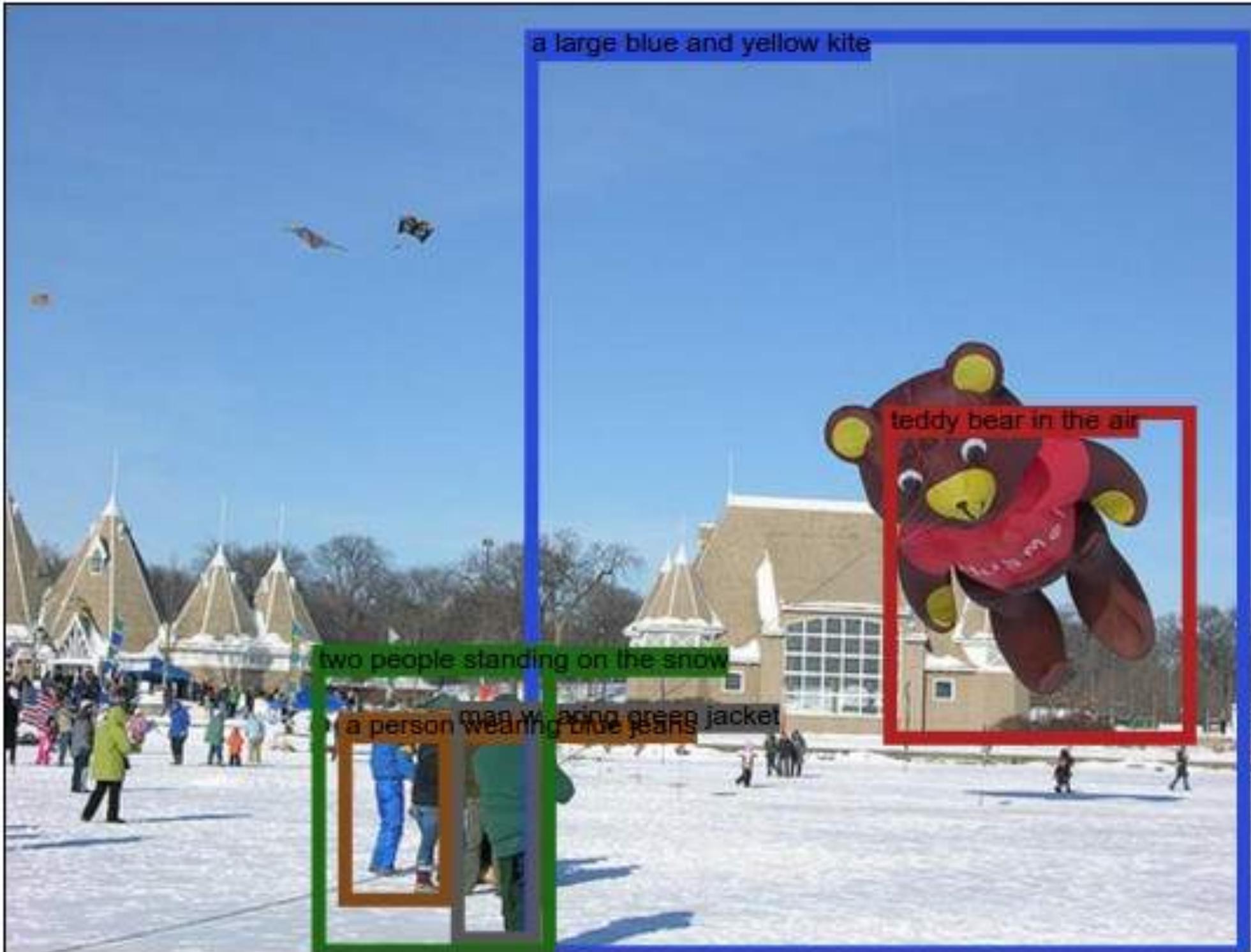
"two young girls are playing with lego toy."

# Image captioning



- Любая предобученная свёрточная сеть для извлечения признаков
- Рекуррентная сеть для генерации описаний

# Dense captioning



**teddy bear in the air.** a large blue and yellow kite.  
man wearing green jacket. **two people standing on the snow.** a person wearing blue jeans.

- Можно скрестить с детекцией объектов — генерировать подпись для каждого объекта
- То есть каждый найденный объект отдаём генератору подписей

[arXiv:1511.07571 DenseCap: Fully Convolutional Localization Networks for Dense Captioning](https://arxiv.org/abs/1511.07571)

# План

- Детекция объектов
- **Семантическая сегментация**
- Сегментация отдельных объектов
- Компьютерное зрение для видео
- Датасеты/реализации/...

# Семантическая сегментация

- Классификация – только класс
- Детекция объектов – класс + положение каждого объекта
- Семантическая сегментация – **класс каждого пикселя**

## Classification



CAT

Single object

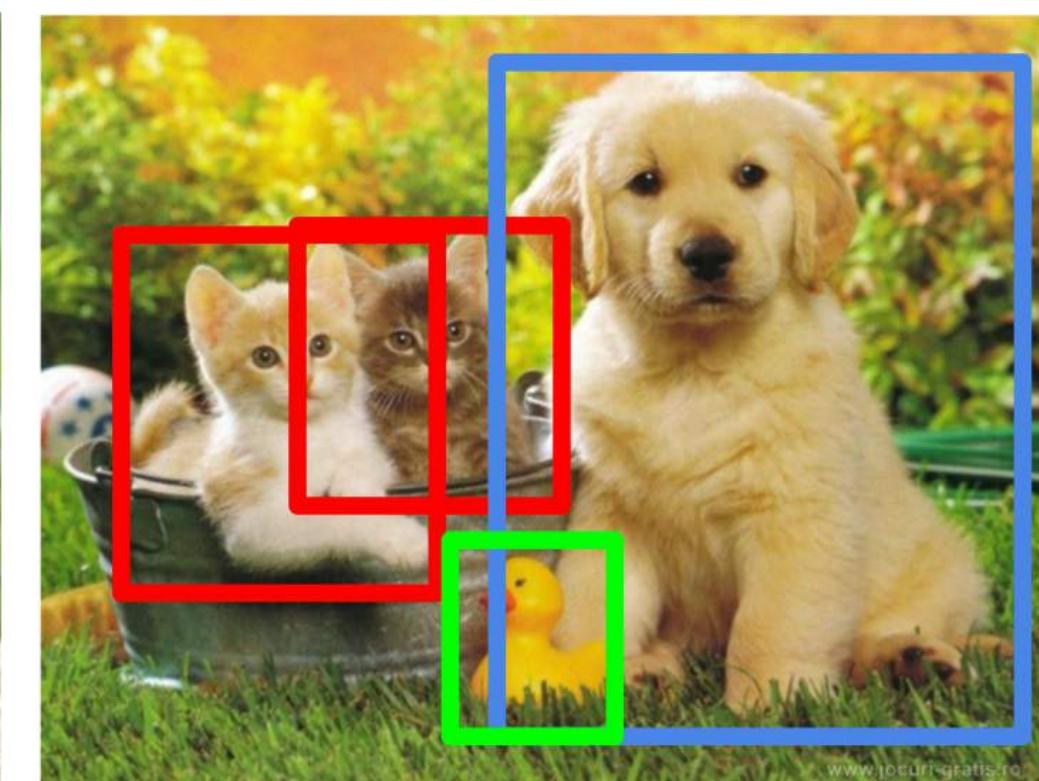
## Classification + Localization



CAT

Single object

## Object Detection



CAT, DOG, DUCK

Multiple objects

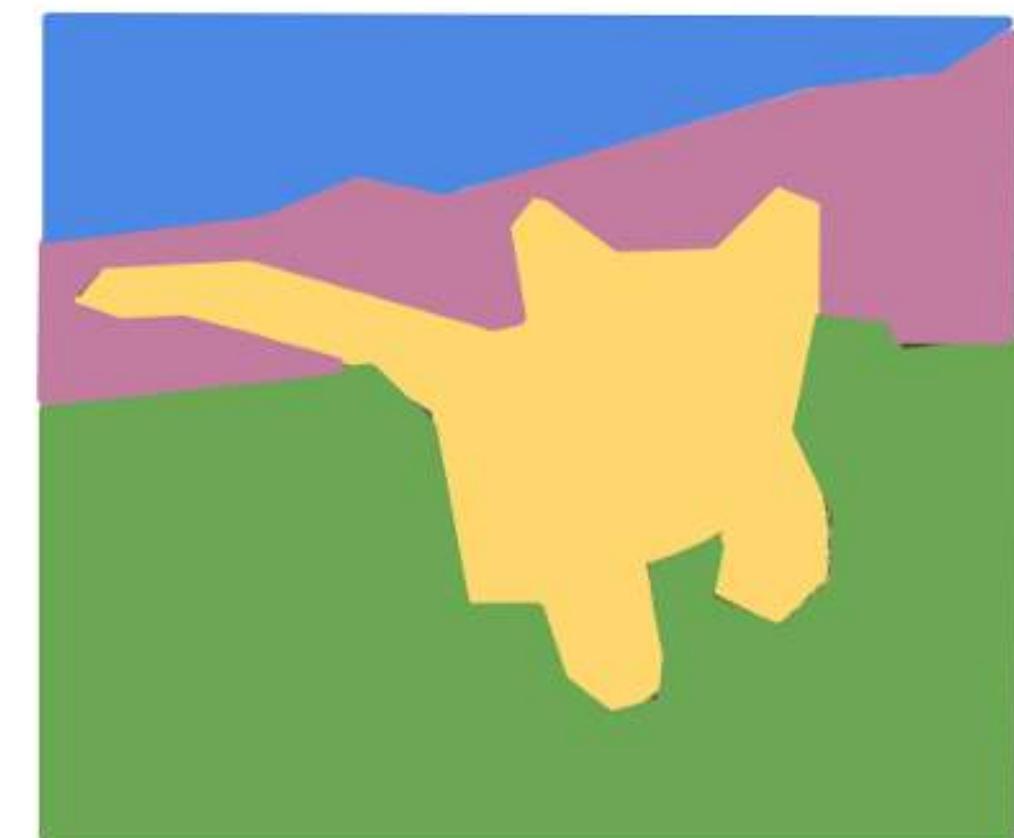
## Instance Segmentation



CAT, DOG, DUCK

Multiple objects

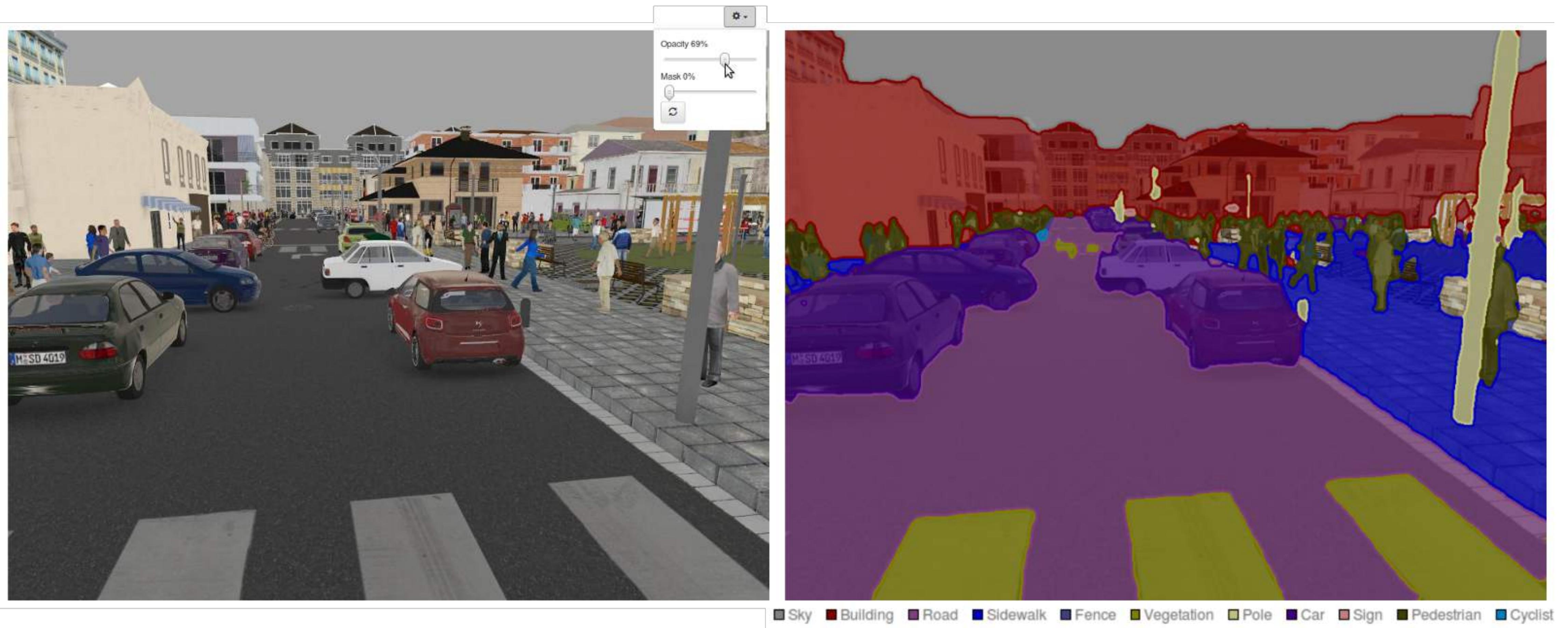
## Semantic Segmentation

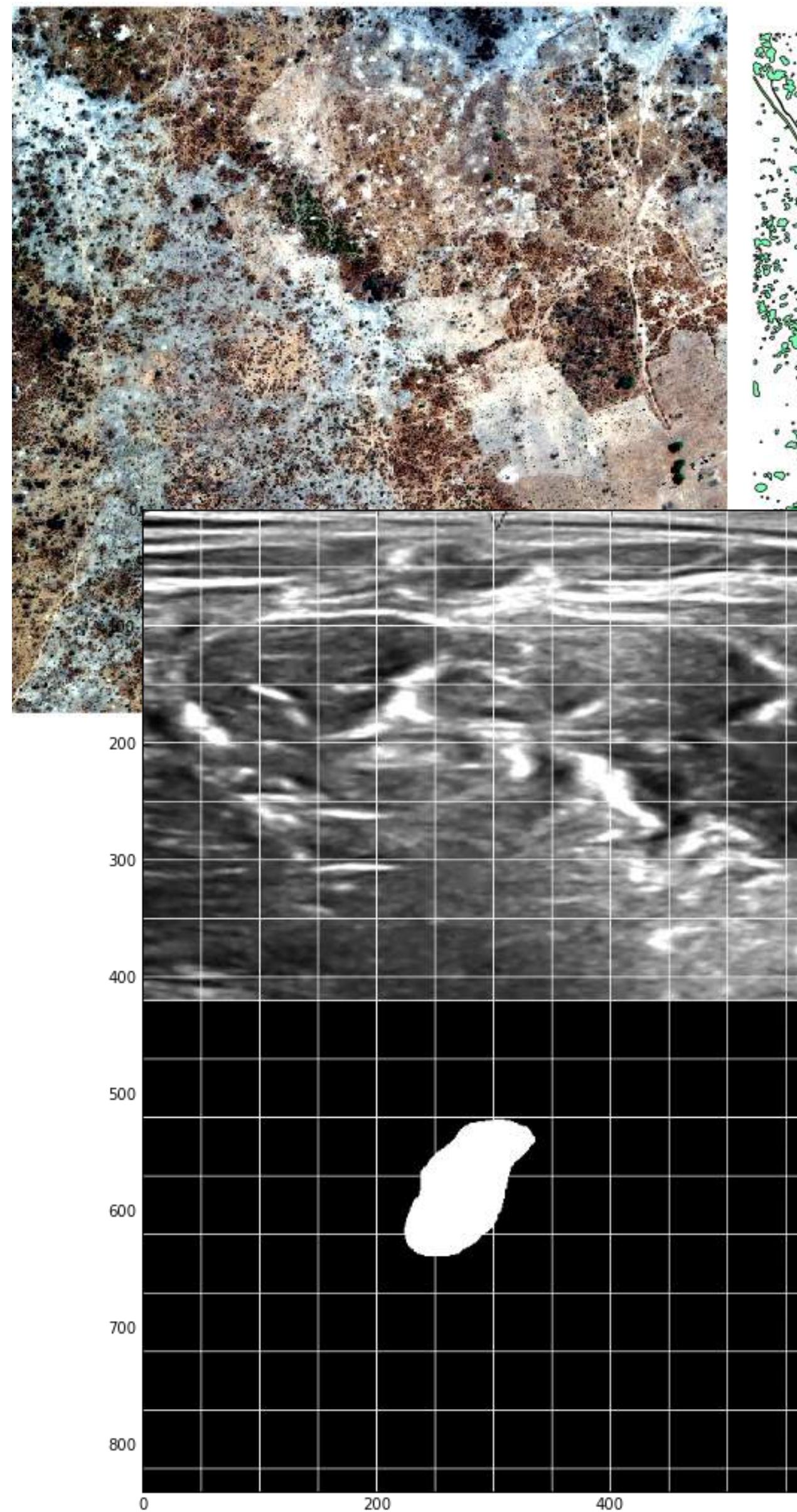


GRASS, CAT,  
TREE, SKY

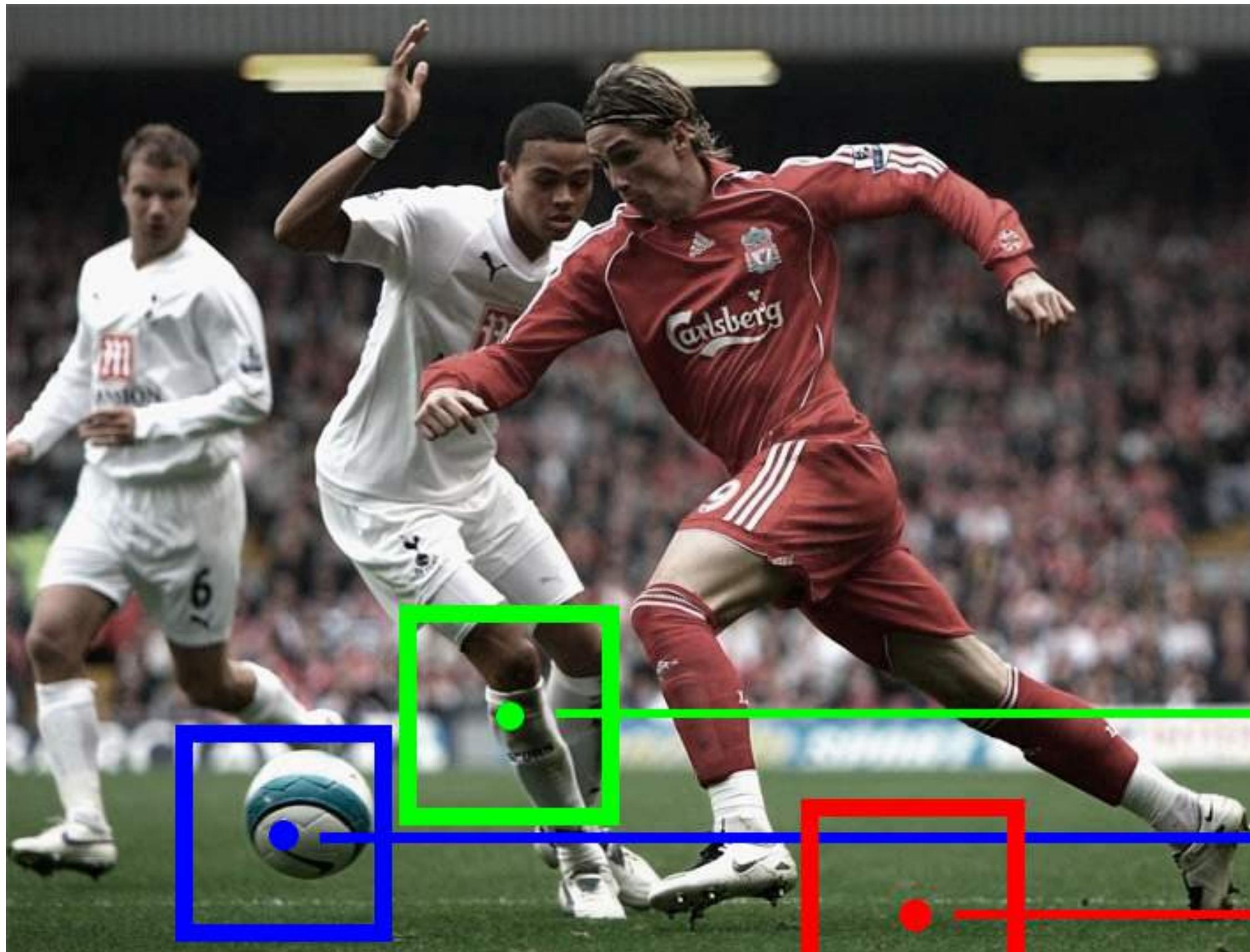
No objects, just pixels

# Семантическая сегментация





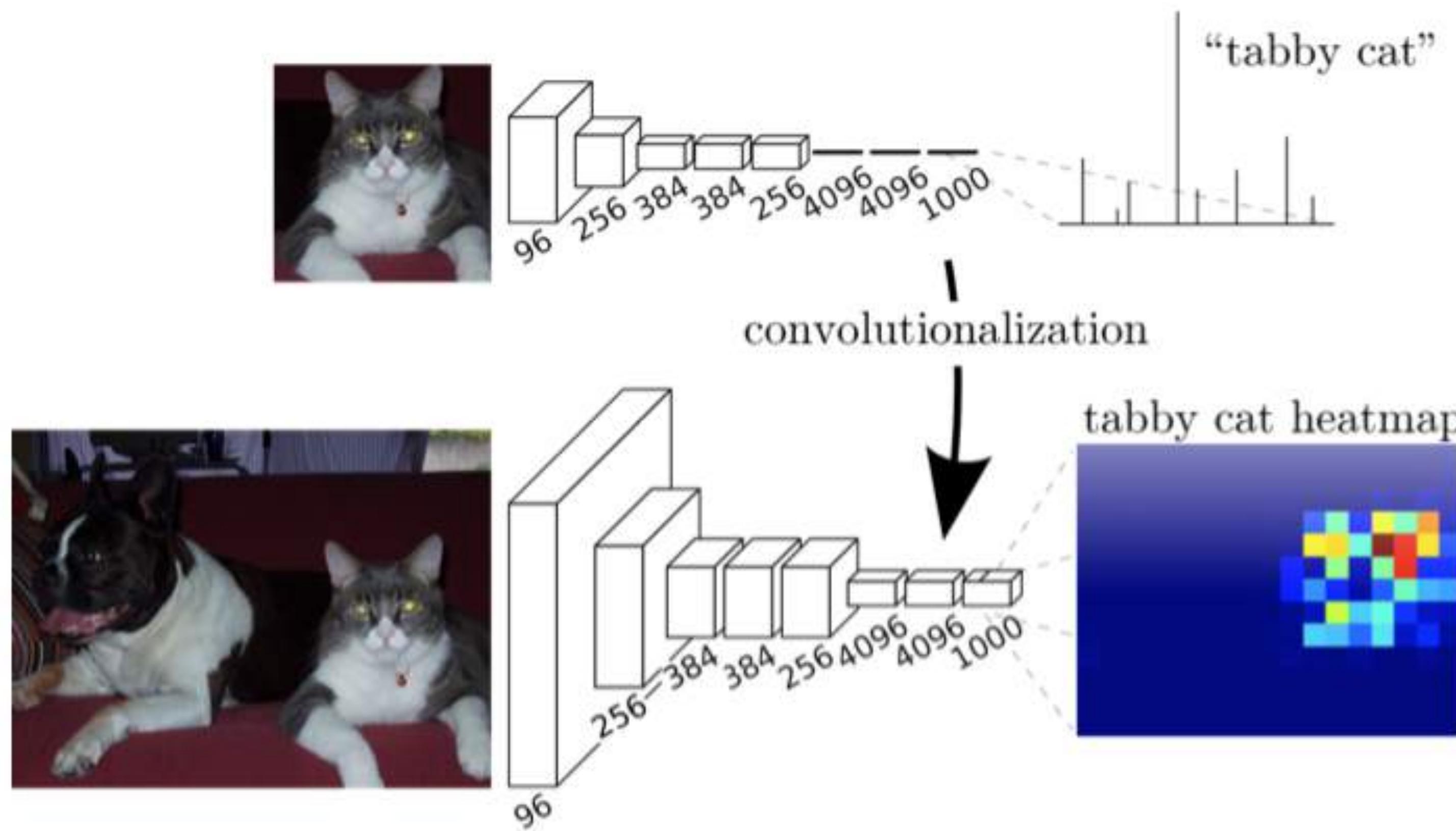
# Классификатор + скользящее окно



- Можно не париться о размере входа (по сравнению с детекцией)
- Те же грабли с вычислительной сложностью

**Футболист**  
**Мяч**  
**Трава**

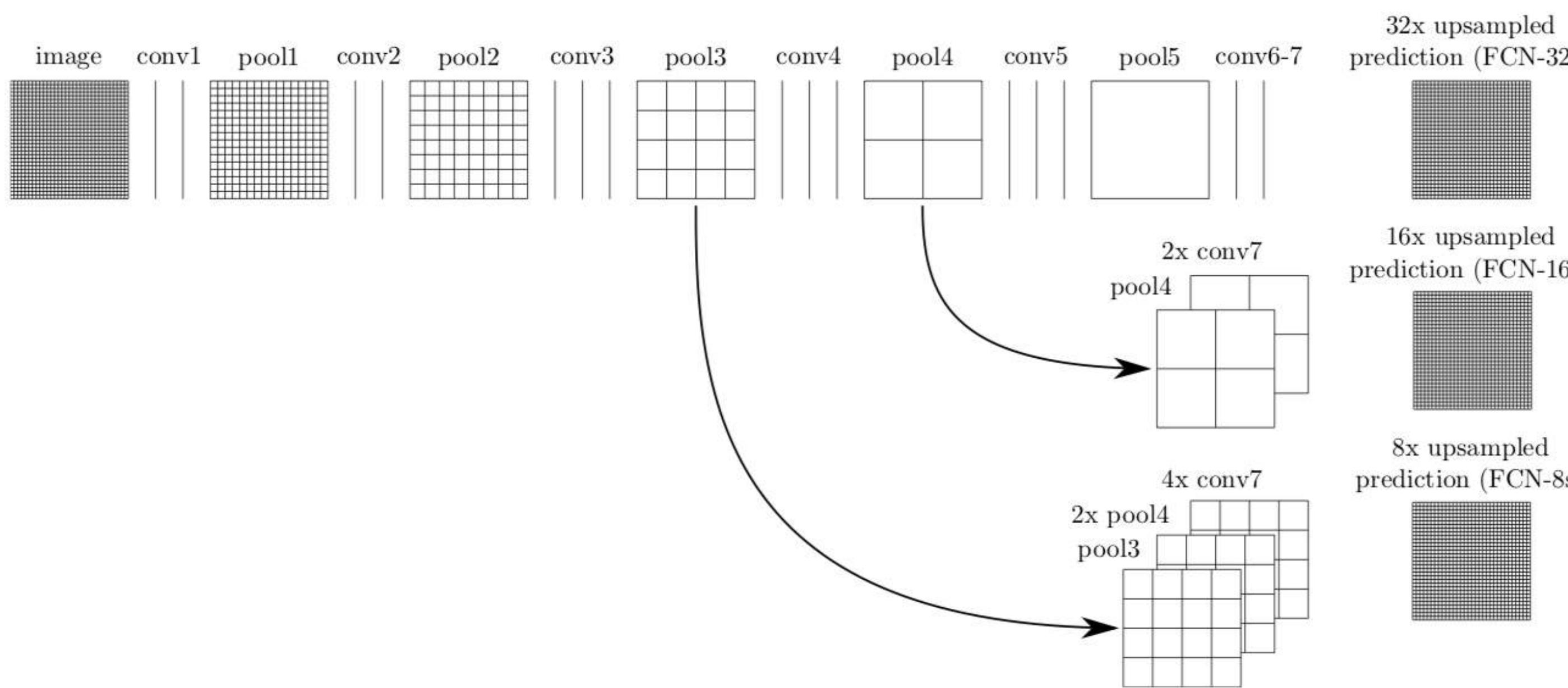
# Fully Convolutional Network



- Аналогично Fast R-CNN будем сразу извлекать свёртками признаки **по всему изображению**
- **Convolutionalization** – разворачиваем полно связные слои в свёртки  $1 \times 1$
- На самом деле получаем уменьшенное изображение

[arXiv:1411.4038 Fully Convolutional Networks for Semantic Segmentation](https://arxiv.org/abs/1411.4038)

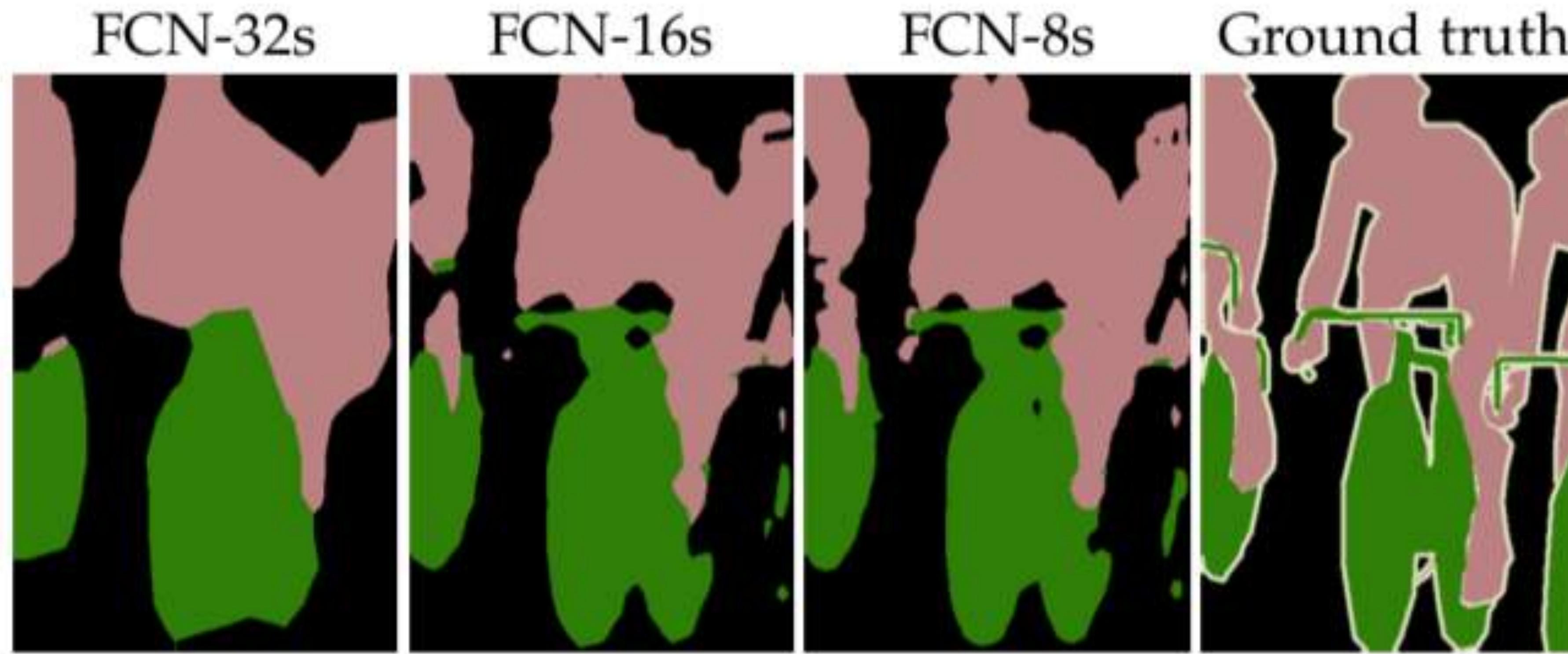
# Fully Convolutional Network



- Для получения исходной размерности надо всё равно делать `upsample 32x`
- Из-за `pooling`-операций потеряли детали изображения
- Можно уточнять разметку признаками с ранних слоёв

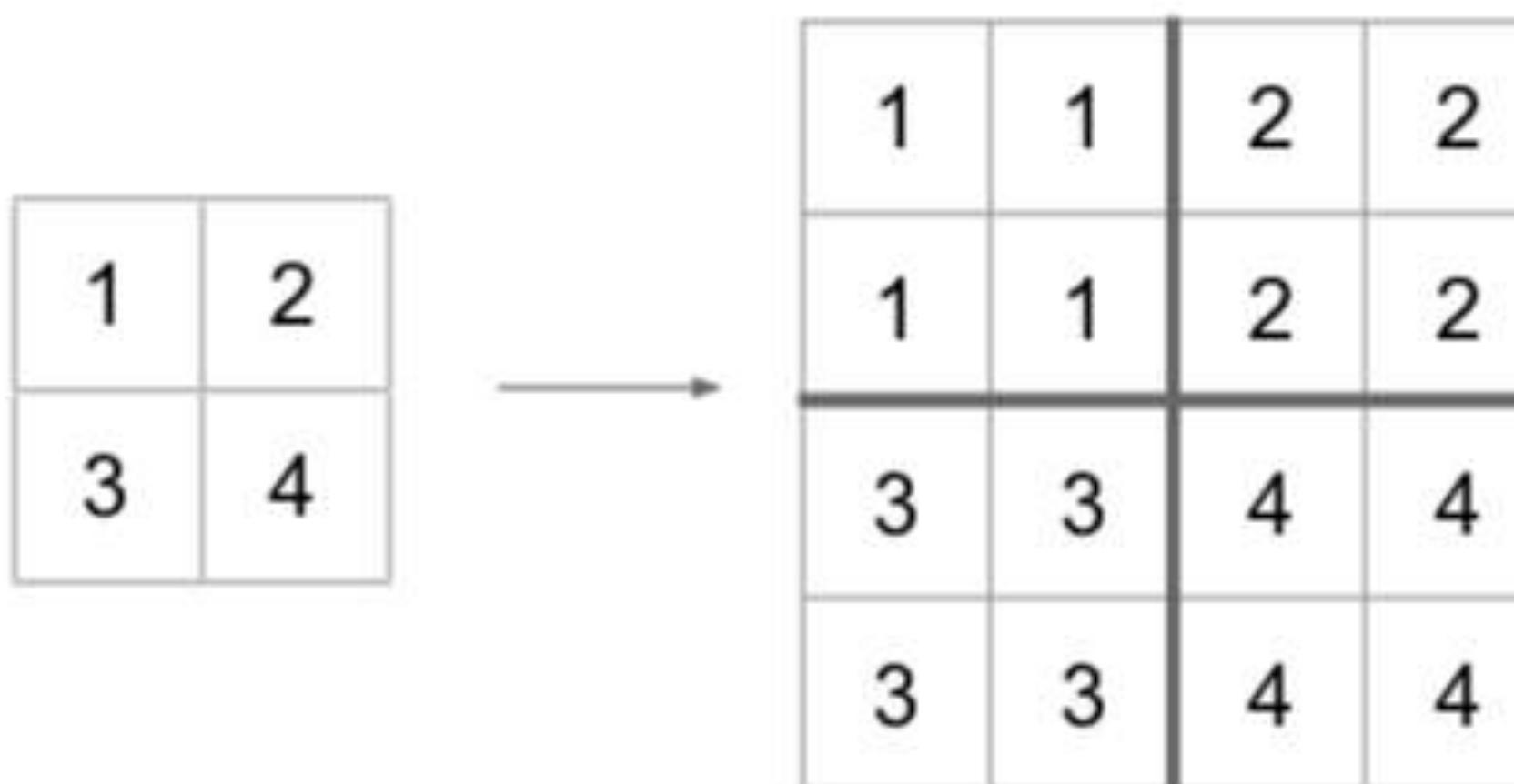
[arXiv:1411.4038 Fully Convolutional Networks for Semantic Segmentation](https://arxiv.org/abs/1411.4038)

# Upsample с ранними признаками

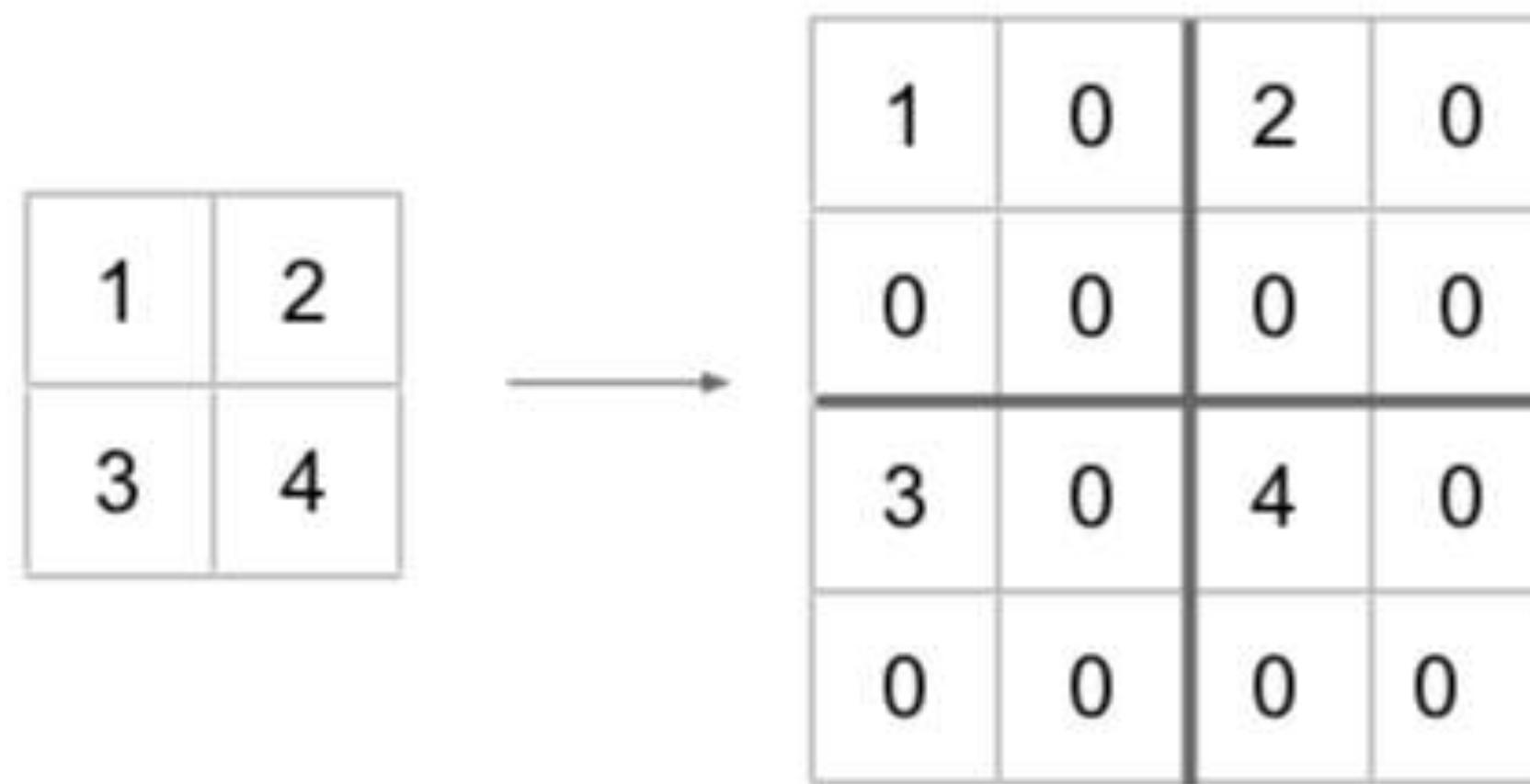


[arXiv:1411.4038 Fully Convolutional Networks for Semantic Segmentation](https://arxiv.org/abs/1411.4038)

# Upsampling



- Ближайшим соседом

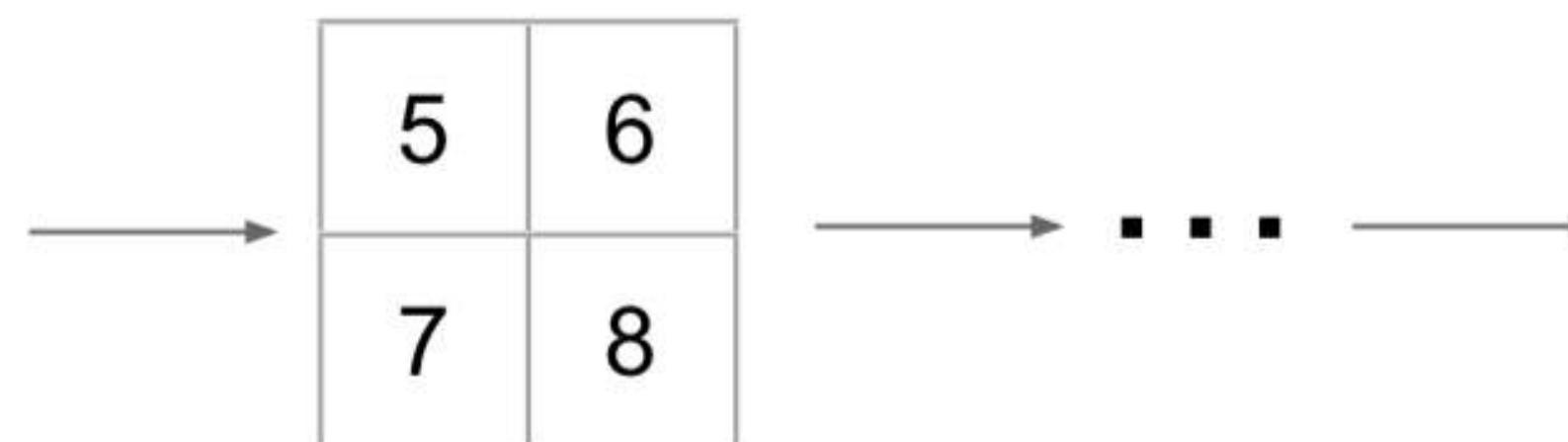


- «Bed of nails»

# Max unpooling

Каждому max pooling слою  
соответствующий max unpooling

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



**Max pooling**  
запоминаем  
индексы  
максимумов

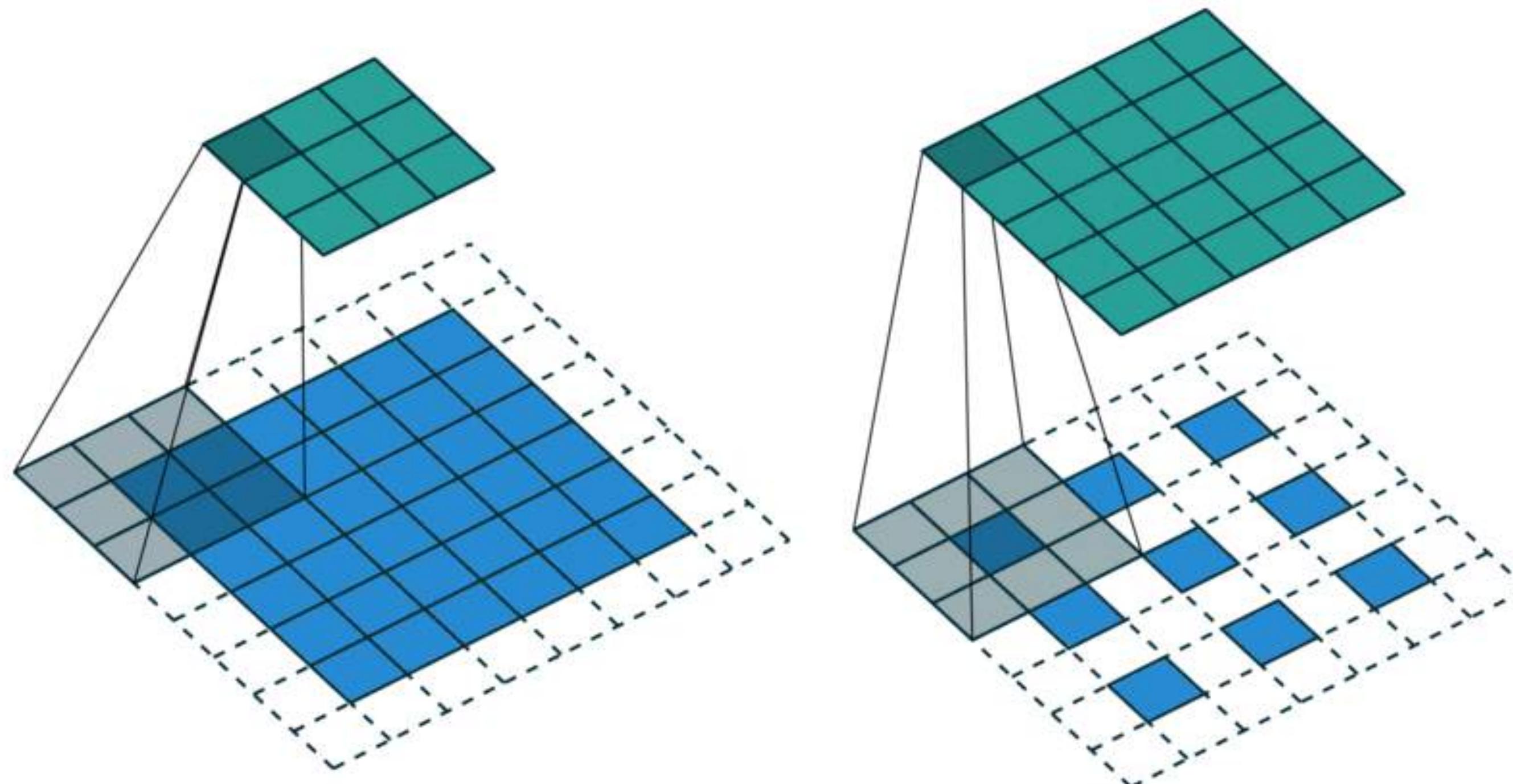
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

**Max unpooling**  
используем  
индексы  
максимумов

# Обучаемый upsampling

- Unpooling / Max unpooling – простые хаки
- Можно делать обучаемый upsampling
- Transpose Convolution / Upconvolution / Fractionally strided convolution – одно и то же

# Fractionally strided convolutions

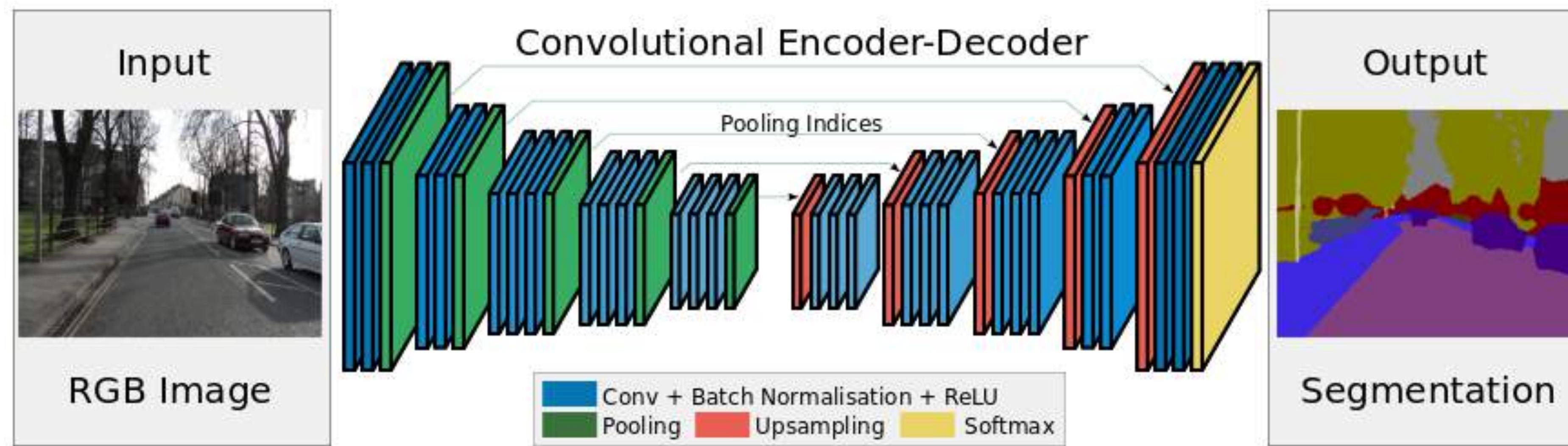


Обычная свёртка

Fractionally  
strided свёртка

- Такой же свёрточный слой,  
но увеличиваем  
размерность

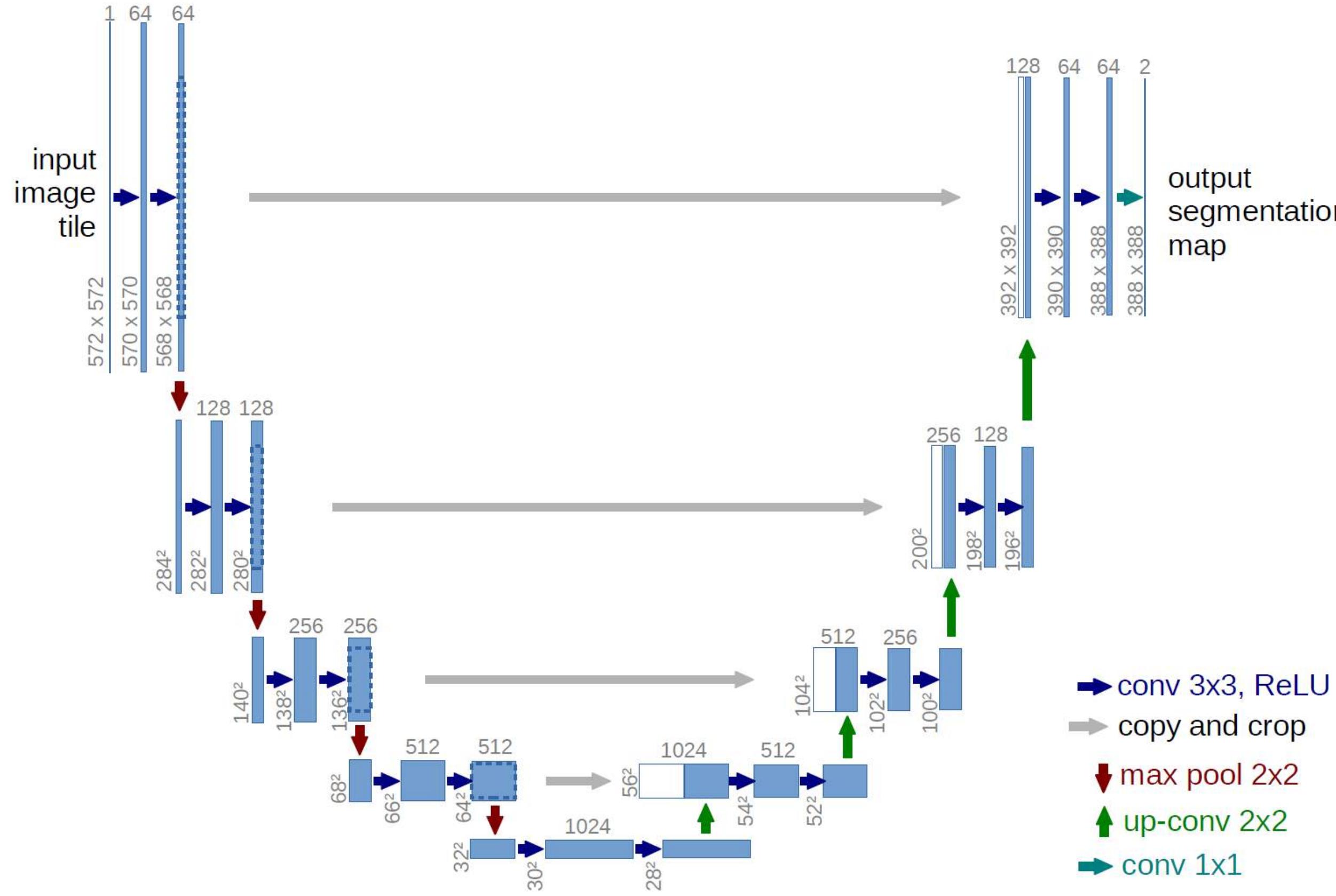
# SegNet



- Архитектура с кодировщиком и декодировщиком
- Нужен upsampling
- Из-за отсутствия полносвязных слоёв получается лёгкой

[arXiv:1511.00561 SegNet: A Deep Convolutional Encoder-Decoder for Image Segmentation](https://arxiv.org/abs/1511.00561)

# U-Net



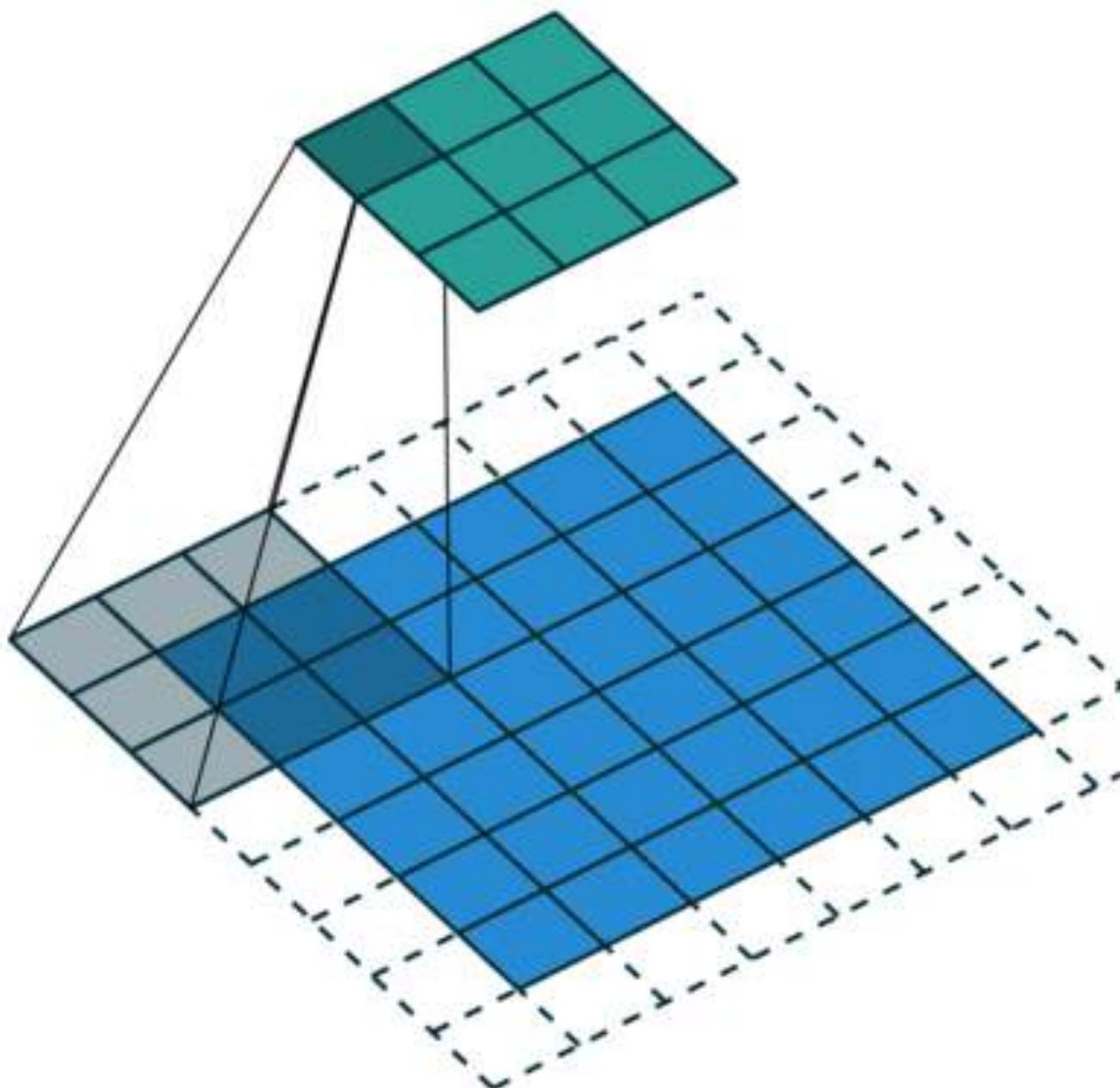
- Промежуточные выходы кодировщика объединяются с промежуточными входами декодировщика

- Upsampling

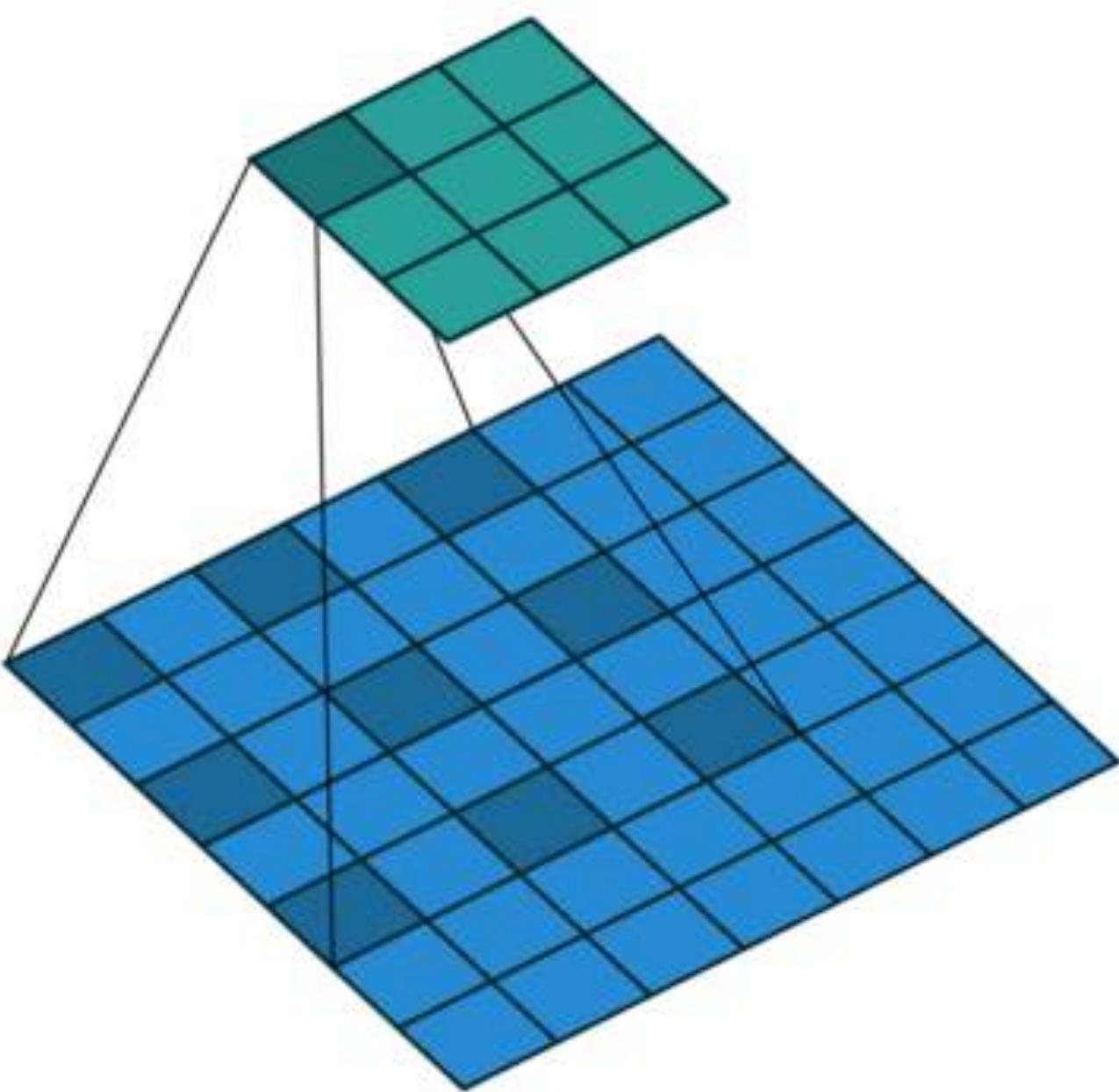
- Зарекомендовал себя на практике

[arXiv:1505.04597 U-Net: Convolutional Networks for Biomedical Image Segmentation](https://arxiv.org/abs/1505.04597)

# Dilated / Atrous convolutions



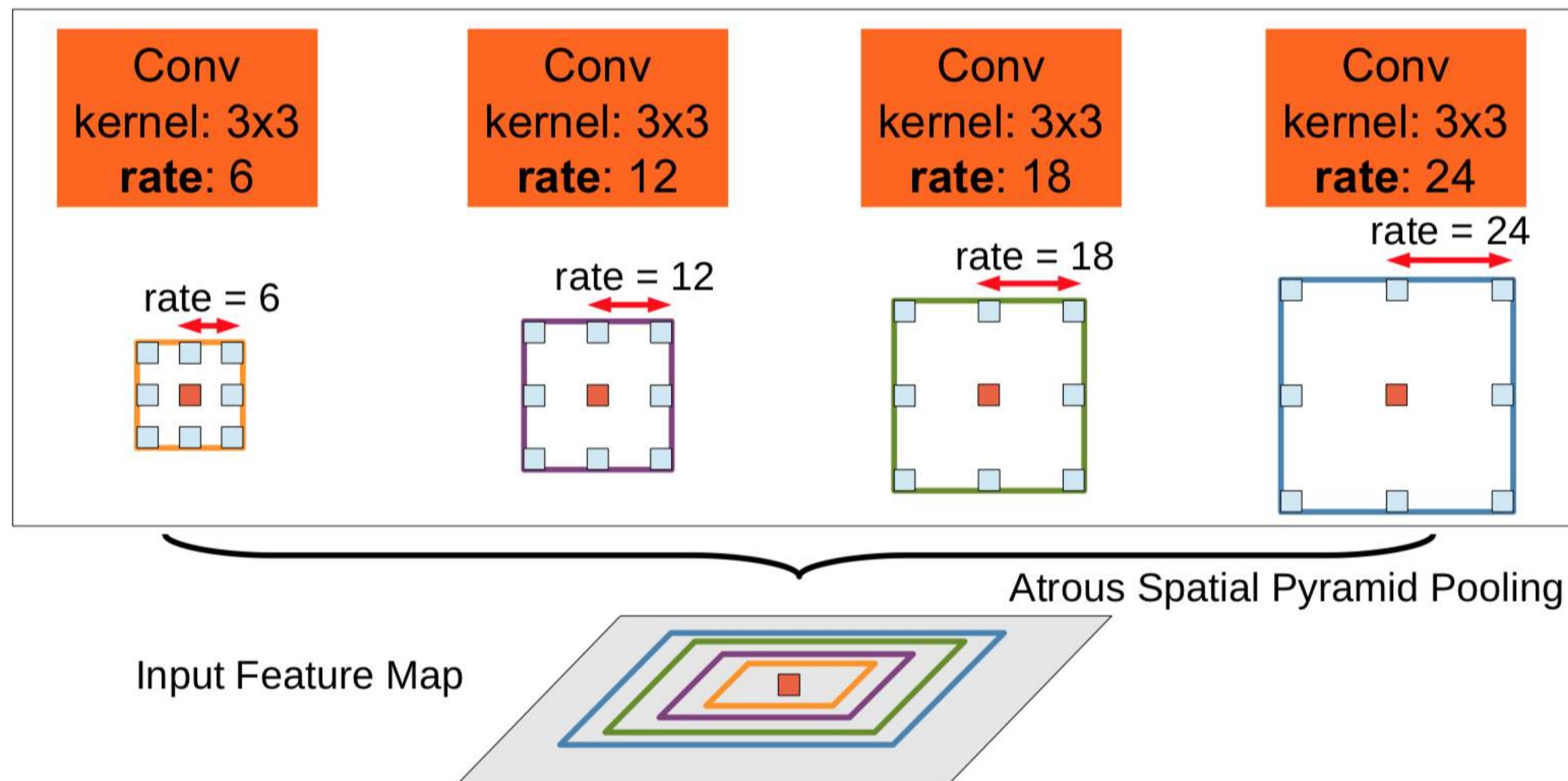
Обычная свёртка



Dilated свёртка

- Вместо того, чтобы улучшать «восстановители» исходной размерности можно уменьшать «снижение размерности»
- Получаем большую разрешающую способность без пулинга

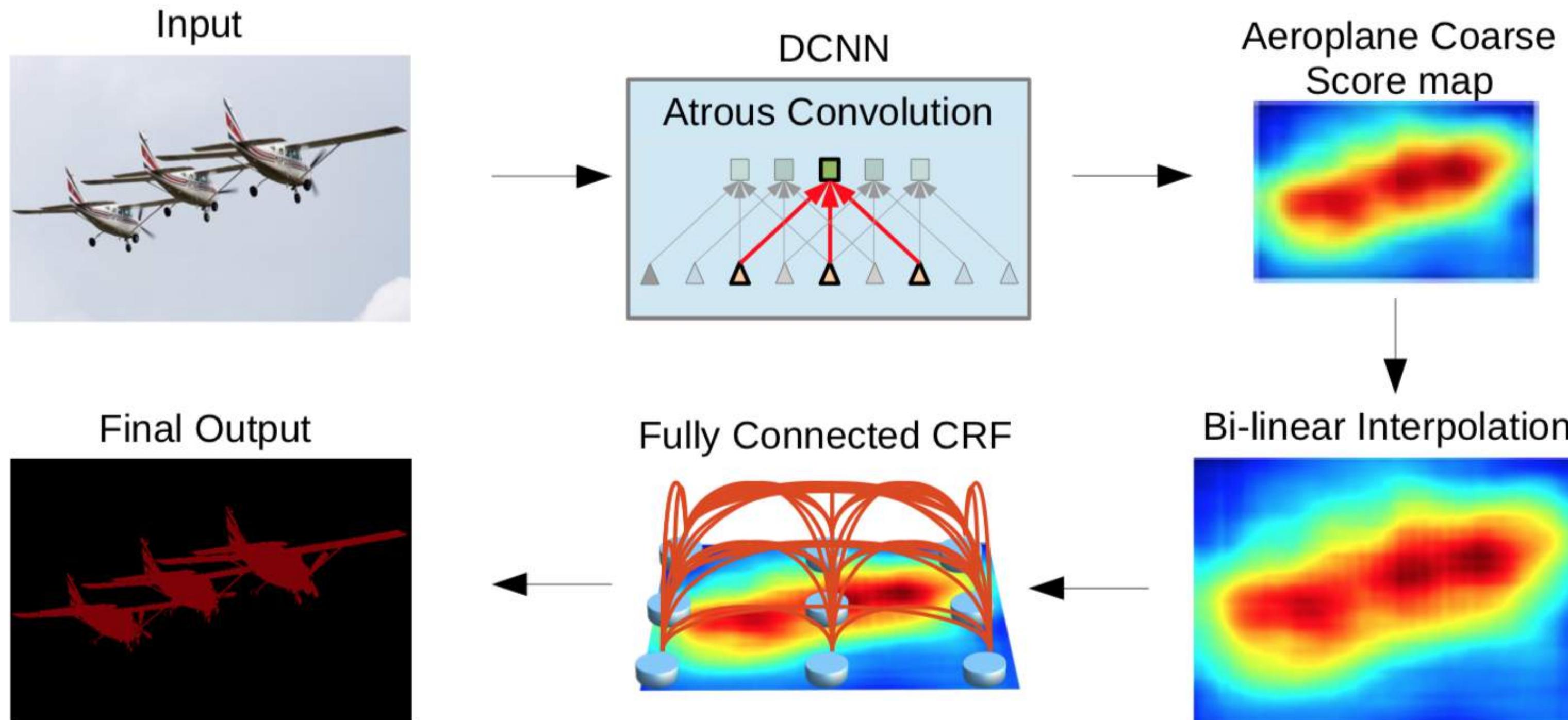
# DeepLab



- **Atrous spatial pyramid pooling**
  - блок для извлечения признаков с dilated свёртками с разными параметрами
- Улучшалась в версиях v2 / v3 / v3+

[arXiv:1412.7062 Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs](https://arxiv.org/abs/1412.7062)

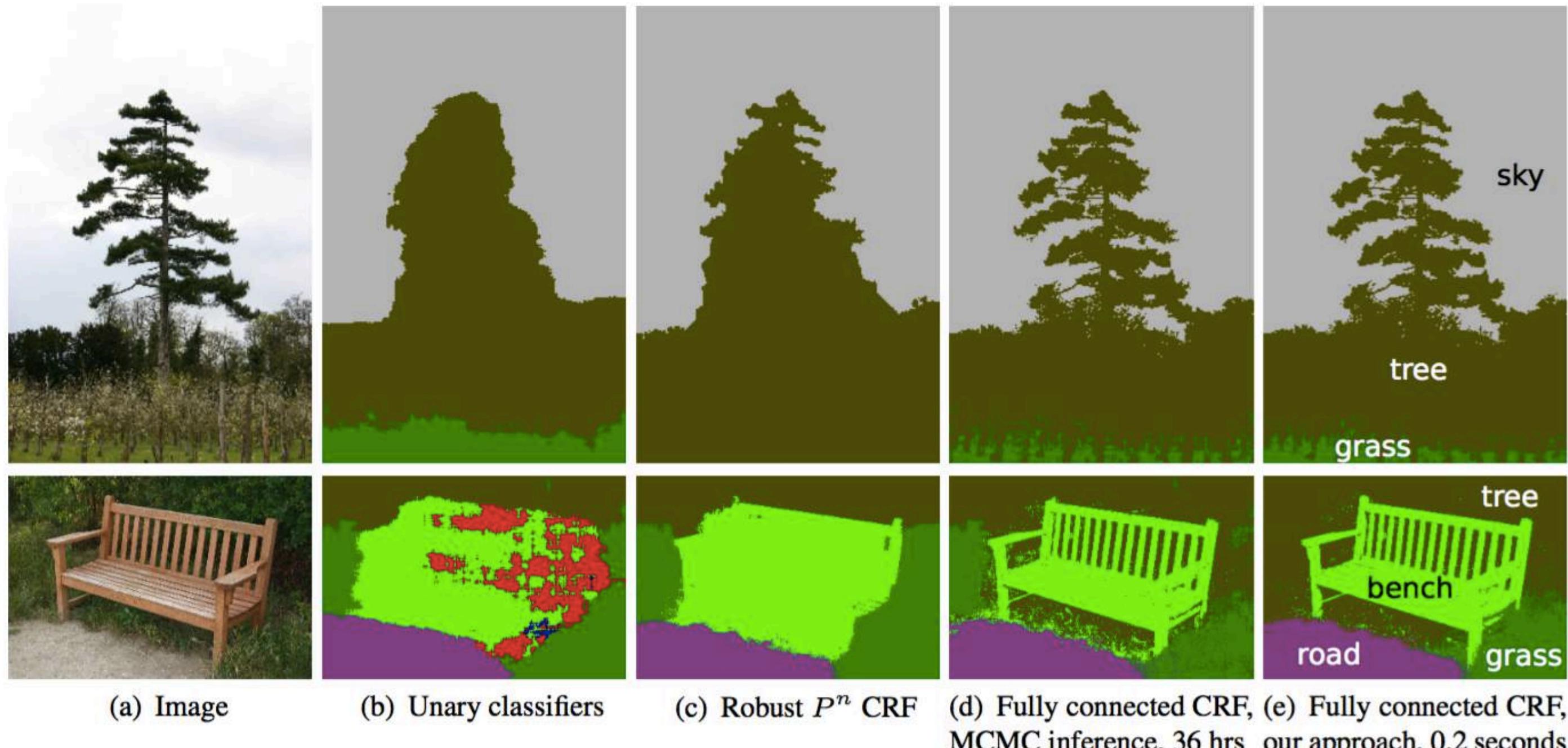
# DeepLab



- Постпроцессинг с conditional random field (CRF) для уточнения масок

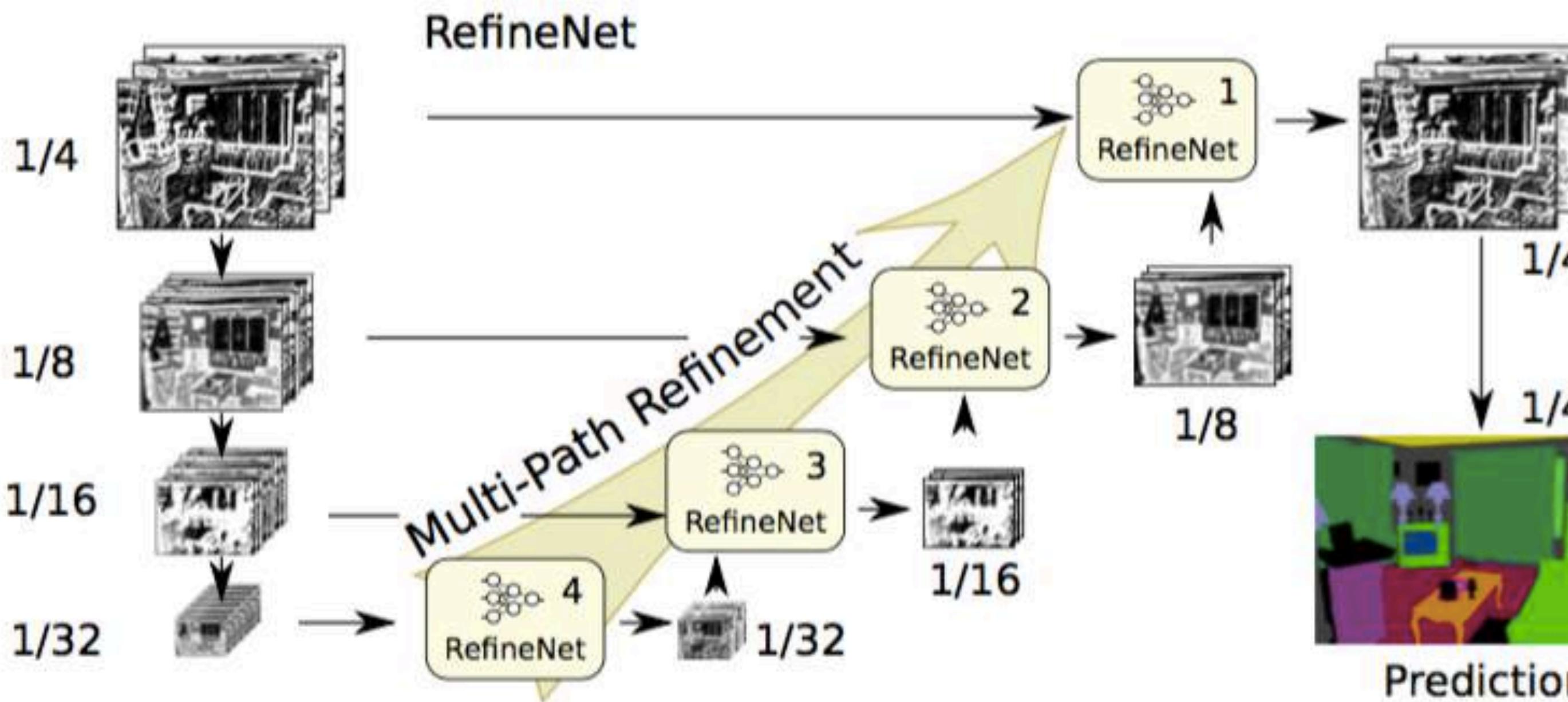
[arXiv:1412.7062 Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs](https://arxiv.org/abs/1412.7062)

# Conditional Random Field



- CRF – графическая модель, сглаживающая маски
- Улучшает качество на 1-2%

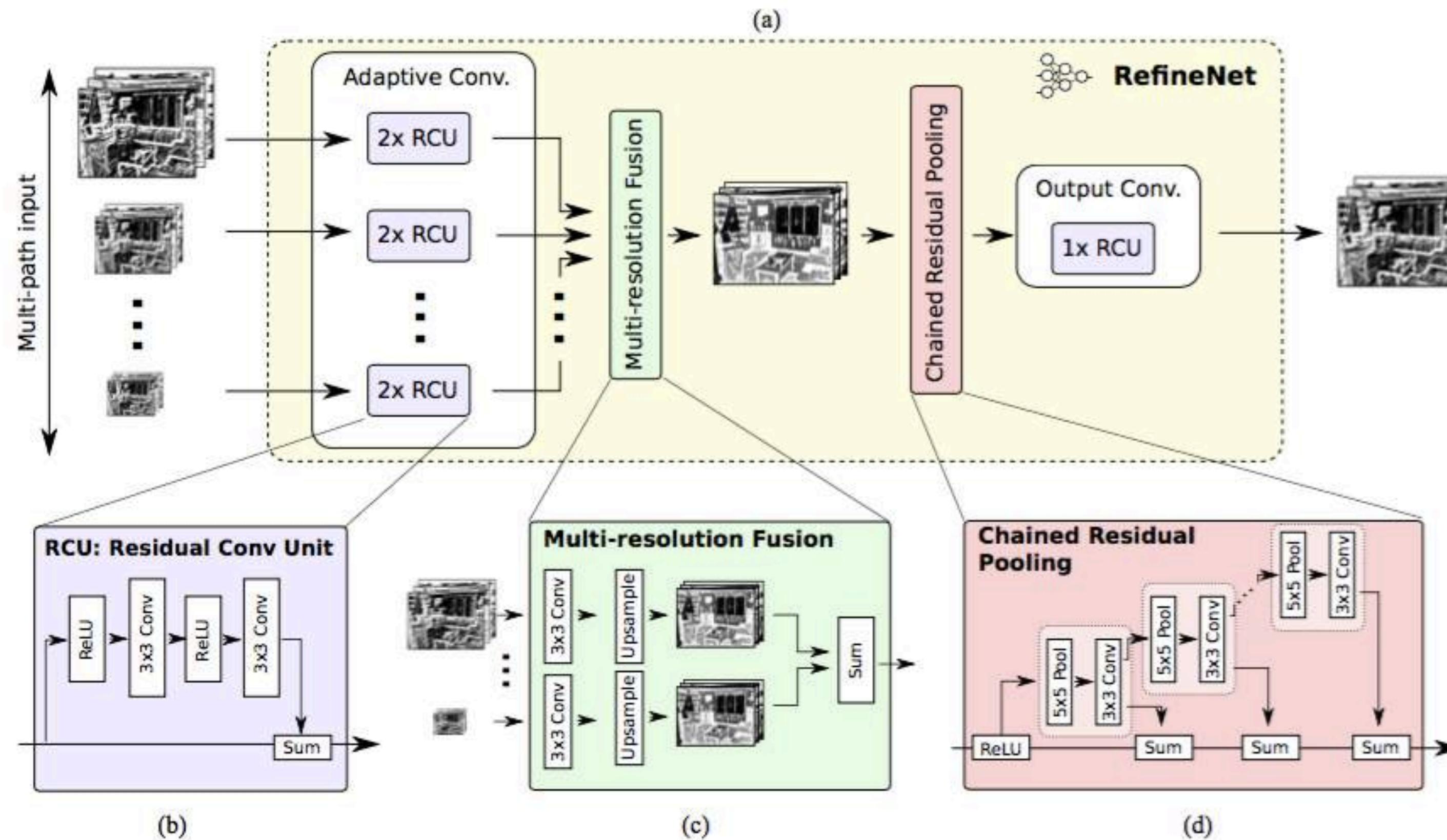
# RefineNet



- Архитектура со сложными «улучшайзерами»

[arXiv:1611.06612 RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation](https://arxiv.org/abs/1611.06612)

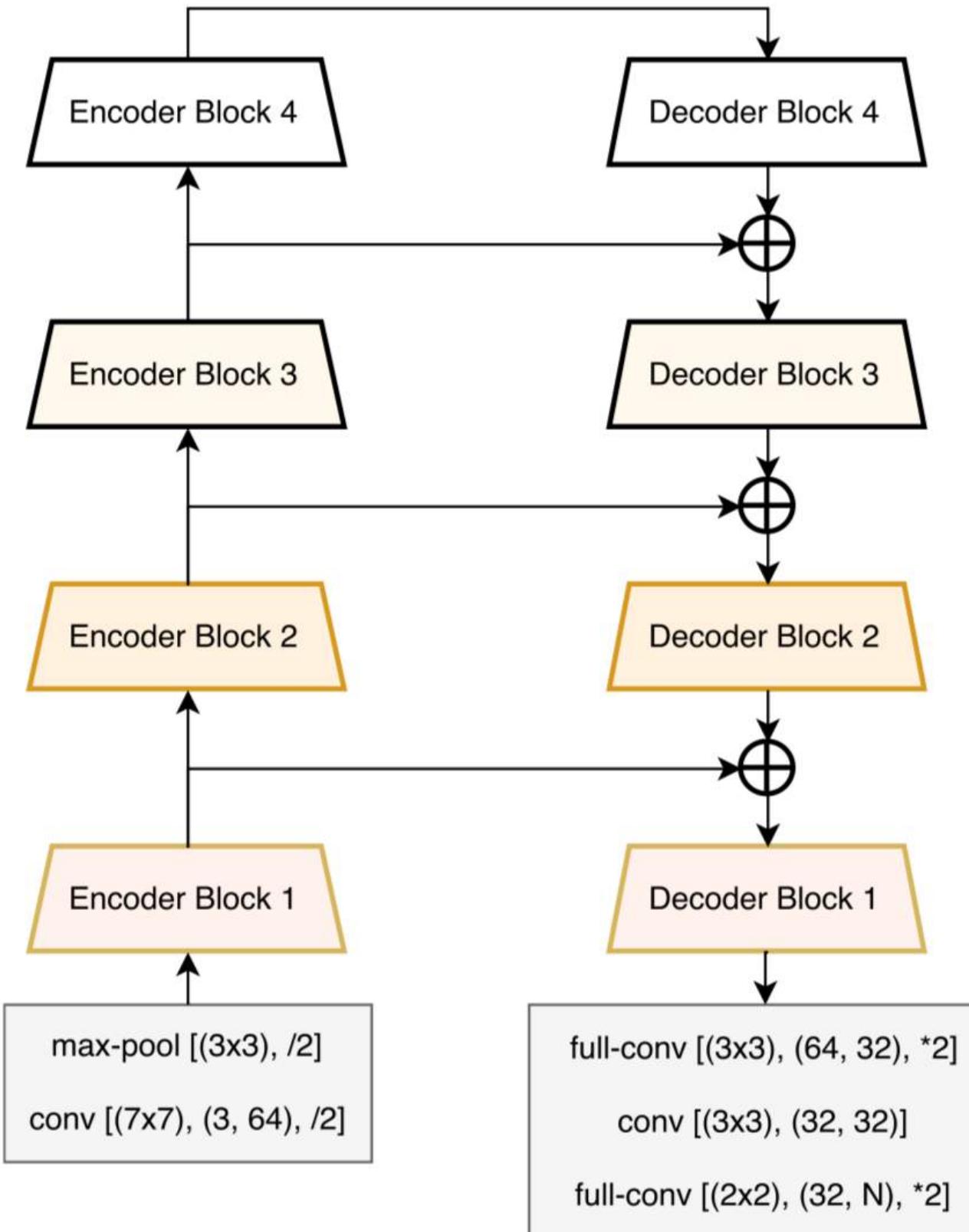
# RefineNet



- Каждый «улучшатель»:
  - Блок из ResNet
  - Смешивание блоков с разных разрешений
  - Агрегатор контекста

[arXiv:1611.06612 RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation](https://arxiv.org/abs/1611.06612)

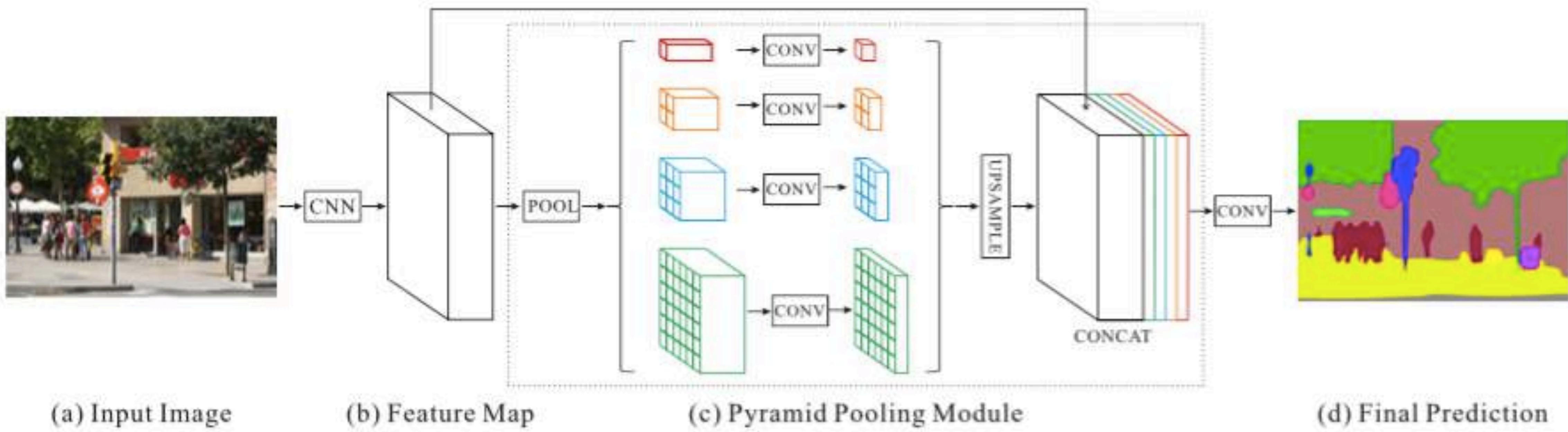
# LinkNet



- Лёгкая Encoder-Decoder архитектура
- Свёртки  $1 \times 1$  для понижения размерности (перед и после жирных свёрток)
- Fractionally strided свёртки для увеличения размерности

[arXiv:1707.03718 LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation](https://arxiv.org/abs/1707.03718)

# PSPNet



- Дополнительные функции потерь на блоках ResNet
- Учёт глобального контекста в Spatial Pyramid Pooling

[arXiv:1612.01105 Pyramid Scene Parsing Network](https://arxiv.org/abs/1612.01105)

# Finally

- Минимизация работы на больших размерностях изображения
- Использование признаков с разных масштабов сети (trade-off между детальностью и глобальным контекстом)
- Deeplab v3 / PSPNet – ≈SoTA, LinkNet – быстрый

# План

- Детекция объектов
- Семантическая сегментация
- **Сегментация отдельных объектов**
- Компьютерное зрение для видео
- Датасеты/реализации/...

# Instance segmentation

Classification



CAT

Single object

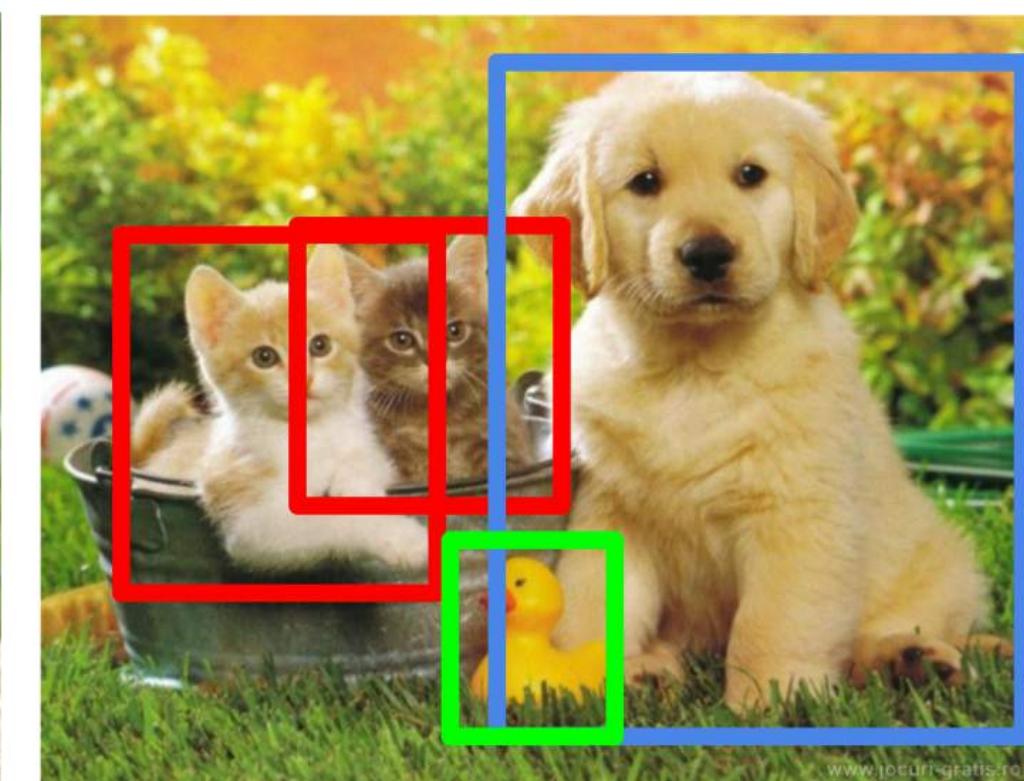
Classification  
+ Localization



CAT

Single object

Object Detection



CAT, DOG, DUCK

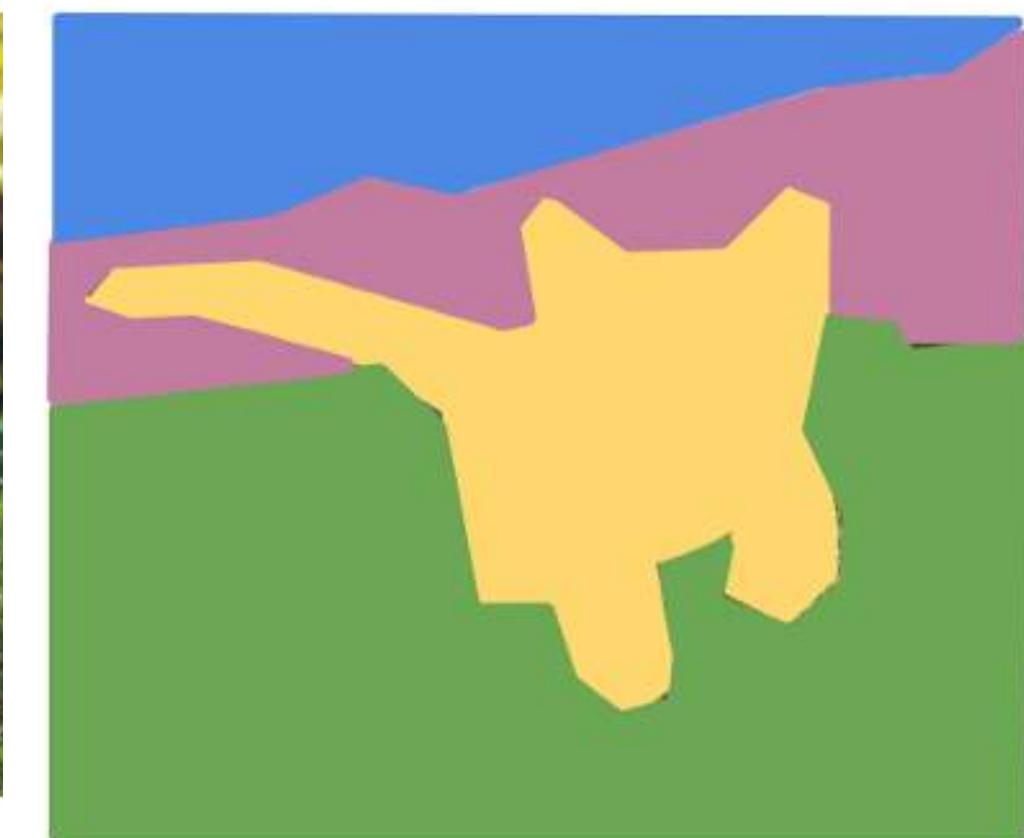
Multiple objects

Instance  
Segmentation



CAT, DOG, DUCK

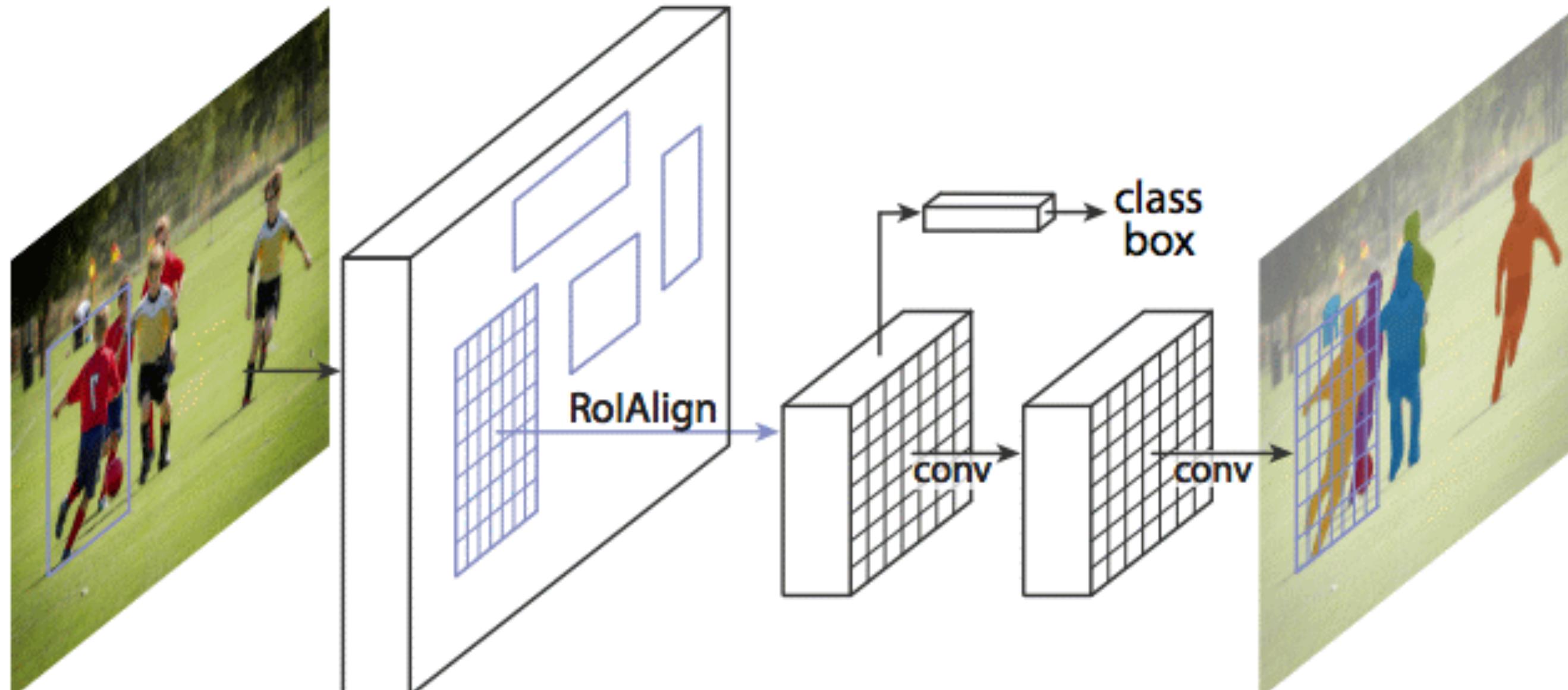
Semantic  
Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

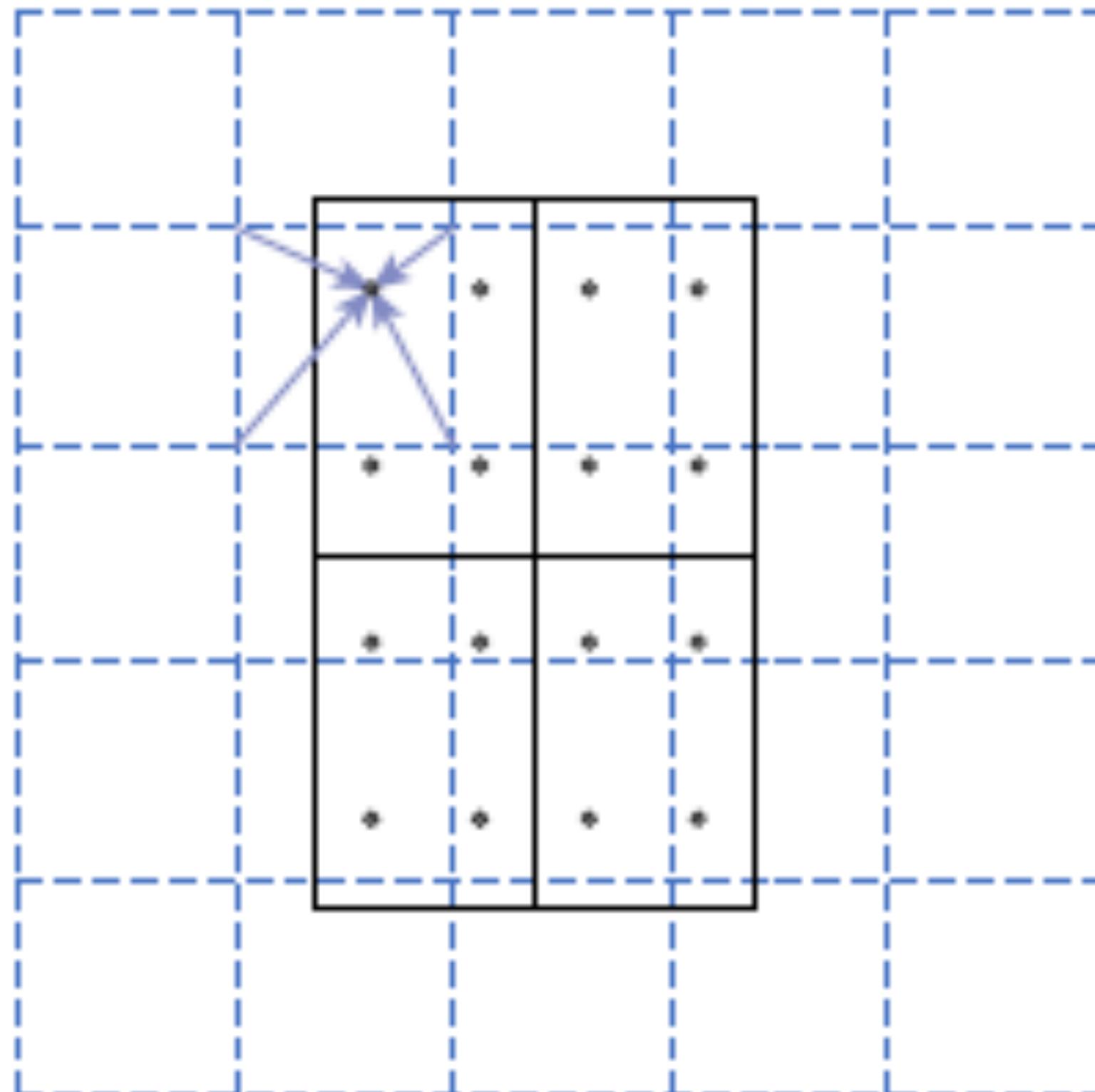
# Mask R-CNN



- Пользуемся подходом Faster R-CNN:
  - извлечение признаков на всём изображении
  - поиск зон-кандидатов
  - предсказание bbox и классов
- Добавляем ветку для предсказания маски (для каждого класса)

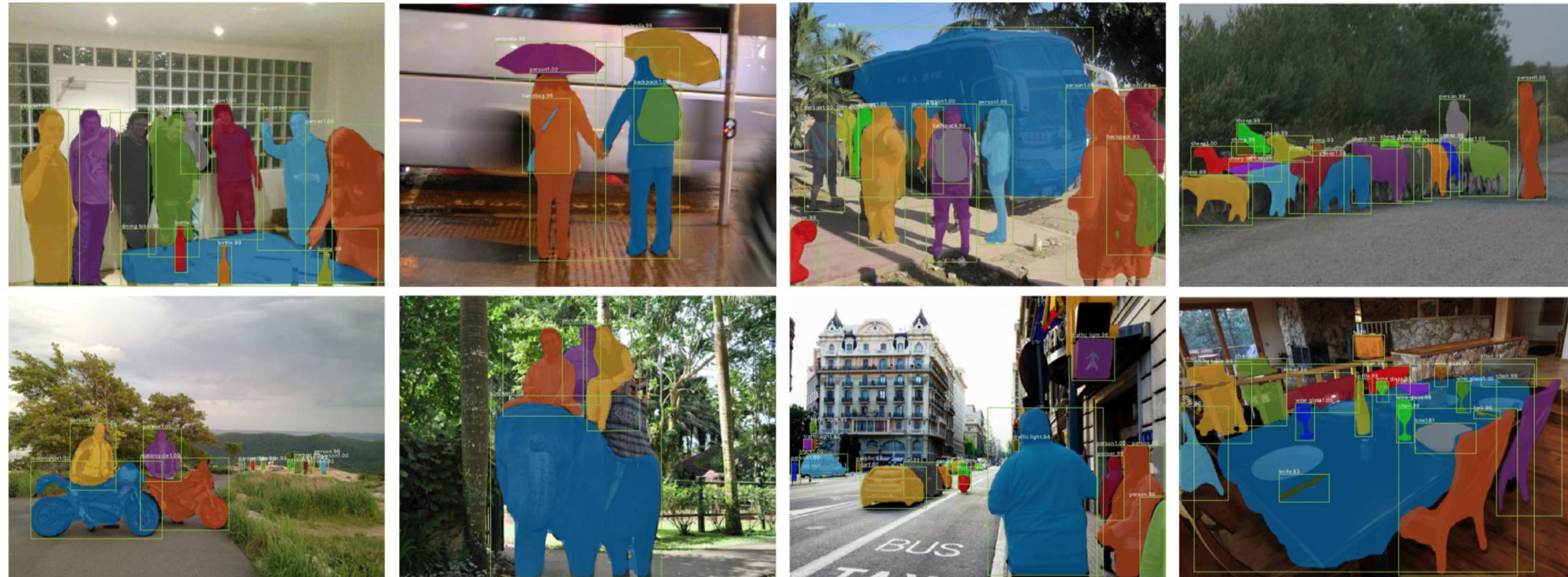
[arXiv:1703.06870 Mask R-CNN](https://arxiv.org/abs/1703.06870)

# RoI Align



- В Faster R-CNN использовали RoI Pooling для проекции признаков в нужную размерность
- Для bbox это нормально, но в сегментации теряются детали
- Вместо округления значения координат билинейная интерполяция по четырём ближайшим целочисленным точкам

# Mask R-CNN



[arXiv:1703.06870 Mask R-CNN](https://arxiv.org/abs/1703.06870)

# Human pose estimation



- В Mask R-CNN в качестве классов возьмём опорные точки скелета
- Можно делать регрессию на координаты опорных точек скелета

# План

- Детекция объектов
- Семантическая сегментация
- Сегментация отдельных объектов
- **Компьютерное зрение для видео**
- Датасеты/реализации/...

# CV для видео

Видео = последовательность кадров, но:

- Вычислительно сложнее (по 30 фреймов на секунду видеоряда)
- Длинный контекст (некоторые кадры могут не содержать объектов)
- Большой простор для архитектур и подходов
- Сложнее с датасетами

# Наивный подход

Покадровое применения известных методов

- + Готовые обученные архитектуры и модели
- Не используется информация о соседних кадрах
- Предсказания «прыгают» и «моргают» от кадра к кадру (например, bbox-ы)

# Наивный подход

В некоторых задачах предсказания нужны по видео  
целиком

- Усреднение предсказаний покадровой модели  
(можно сложную логику)
- Усреднение признаков и обучать модель второго  
уровня — можно сделать end-to-end модель

# Обогащение покадровой модели

Вспомогательная информация из соседних кадров  
(например, выраженная через движение объектов)

- Стекать несколько соседних кадров и подавать сети целиком
- Optical flow

# Optical flow

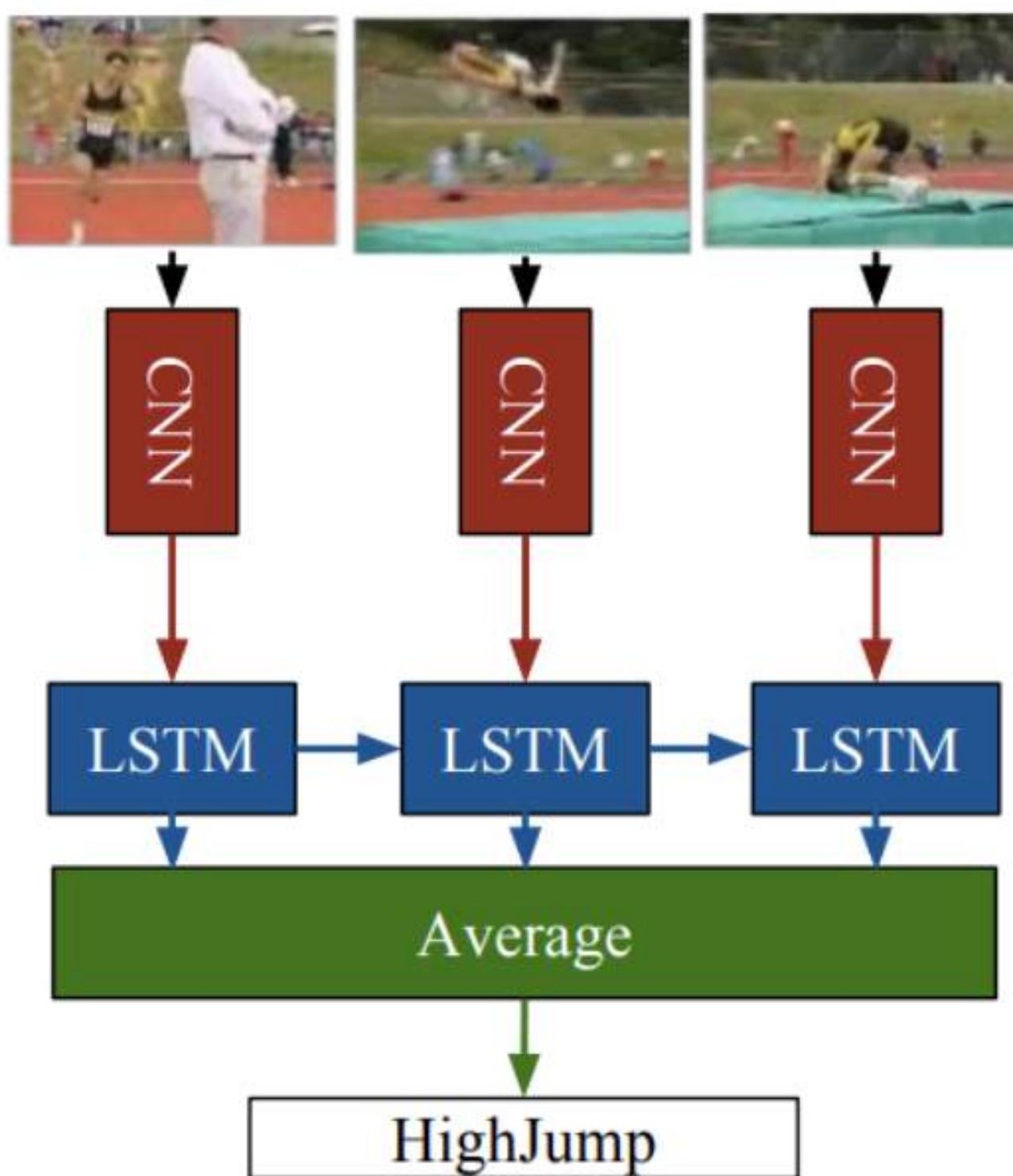


- Сопоставление точек двух изображений — определение направления движения точек
- Передадим динамику из видео в покадровую модель
- Есть стандартные подходы в openCV, есть DL подходы

# Рекуррентные сети

## Activity Recognition

Sequences in the Input



- Хочется, чтобы модель второго уровня использовала информацию о последовательности кадров
- Идейно подходят рекуррентные сети

# План

- Детекция объектов
- Семантическая сегментация
- Сегментация отдельных объектов
- Компьютерное зрение для видео
- **Датасеты/реализации/...**

# Реализации

- Реализовать вручную по статье:
  - Сложно исправлять ошибки
  - Бывают нюансы, неописанные в статье
  - Занимает много времени
- + Легче модифицировать и встраивать

# Реализации

- Найти готовую:
  - + Ниже порог входа (легко запустить и проверить базовое качество)
  - Выше порог входа на модификацию и встраивание (не всегда!)
- Ко всем сколько-нибудь известным статьям написана не одна реализация на каждом из фреймворков (иногда с запозданием!)

# Реализации

- Искать в Google/Github/...: <method> <framework>
- Иногда реализации от самих авторов статьи (далеко не всегда аккуратные и эффективные)
- Подборки реализаций (например, страница [awesome-semantic-segmentation](#))

# Awesome Semantic Segmentation

## Networks by architecture

### Semantic segmentation

- U-Net [<https://arxiv.org/pdf/1505.04597.pdf>] [2015]
  - <https://github.com/zhxuhao/unet> [Keras] ⚡ Stars 1k
  - <https://github.com/jocicmarko/ultrasound-nerve-segmentation> [Keras] ⚡ Stars 805
  - <https://github.com/EdwardTyantov/ultrasound-nerve-segmentation> [Keras] ⚡ Stars 157
  - [https://github.com/ZFTurbo/ZF\\_UNET\\_224\\_Pretrained\\_Model](https://github.com/ZFTurbo/ZF_UNET_224_Pretrained_Model) [Keras] ⚡ Stars 163
  - <https://github.com/yihui-he/u-net> [Keras] ⚡ Stars 288
  - [https://github.com/jakeret/tf\\_unet](https://github.com/jakeret/tf_unet) [Tensorflow] ⚡ Stars 1k
  - <https://github.com/divamgupta/image-segmentation-keras> [Keras] ⚡ Stars 513
  - <https://github.com/ZijunDeng/pytorch-semantic-segmentation> [PyTorch] ⚡ Stars 877
  - <https://github.com/akirasosa/mobile-semantic-segmentation> [Keras] ⚡ Stars 407
  - <https://github.com/orobix/retina-unet> [Keras] ⚡ Stars 697
  - <https://github.com/qureai/ultrasound-nerve-segmentation-using-torchnet> [Torch] ⚡ Stars 38
  - <https://github.com/ternaus/TernausNet> [PyTorch] ⚡ Stars 625
  - [https://github.com/qubvel/segmentation\\_models](https://github.com/qubvel/segmentation_models) [Keras] ⚡ Stars 625
  - [https://github.com/LeeJunHyun/Image\\_Segmentation#u-net](https://github.com/LeeJunHyun/Image_Segmentation#u-net) [PyTorch] ⚡ Stars 195
  - <https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/> [Caffe + Matlab]
- SegNet [<https://arxiv.org/pdf/1511.00561.pdf>] [2016]
  - <https://github.com/alexgkendall/caffe-segnet> [Caffe]
  - <https://github.com/developmentseed/caffe/tree/segnet-multi-gpu> [Caffe]

# Датасеты



Microsoft Common Objects in Context  
(MS COCO)

- 80 классов
- 200k изображений с лейблами
- Детекция объектов
- Семантическая сегментация (в том числе по объектам)

# Датасеты



a.



b.



c.

The PASCAL Visual Object Classes Challenge  
2012 (VOC2012)

- 20 классов
- Конкурс по сравнению методов
- Детекция объектов – 11k картинок с 27k объектами
- Семантическая сегментация (в том числе по объектам) – 3k картинок с 7k объектами

# Датасеты

- Open Images – детекция объектов и связи между объектами
- KITTI – семантическая сегментации дорог с камеры автомобиля
- Cityscapes – семантическая сегментация (и объектная) городских сцен с камеры автомобиля
- Visual genome – детекция объектов и связи между объектами
- ImageNet – не только задача классификации, но и детекция и локализация объектов