

# Сегментация и поиск изображений

Александр Дьяконов

16 марта 2020 года

## План: задачи с изображениями

**Классификация – что изображено**

**Локализация – где изображено**

**Детектирование – что и где**

**Сегментация – матрица меток сегментов  
семантическая сегментация / сегментация объектов**

**Преобразование изображений**

- удаление шума
- стилизация

**Восстановление объектов (ex: 3D-модели)**

## Семантическая сегментация

### Расстановка меток классов для пикселей



segmented →

- 1: Person
- 2: Purse
- 3: Plants/Grass
- 4: Sidewalk
- 5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	1	1	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	1	1	3	3	3	3	3	5	5	5	5	5	5
5	5	3	3	3	3	3	3	3	3	1	1	3	3	3	3	3	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	1	1	1	1	4	4	4	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	1	1	1	1	4	4	4	4	5	5	5	5	5
4	4	4	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4

<https://www.jeremyjordan.me/semantic-segmentation/>

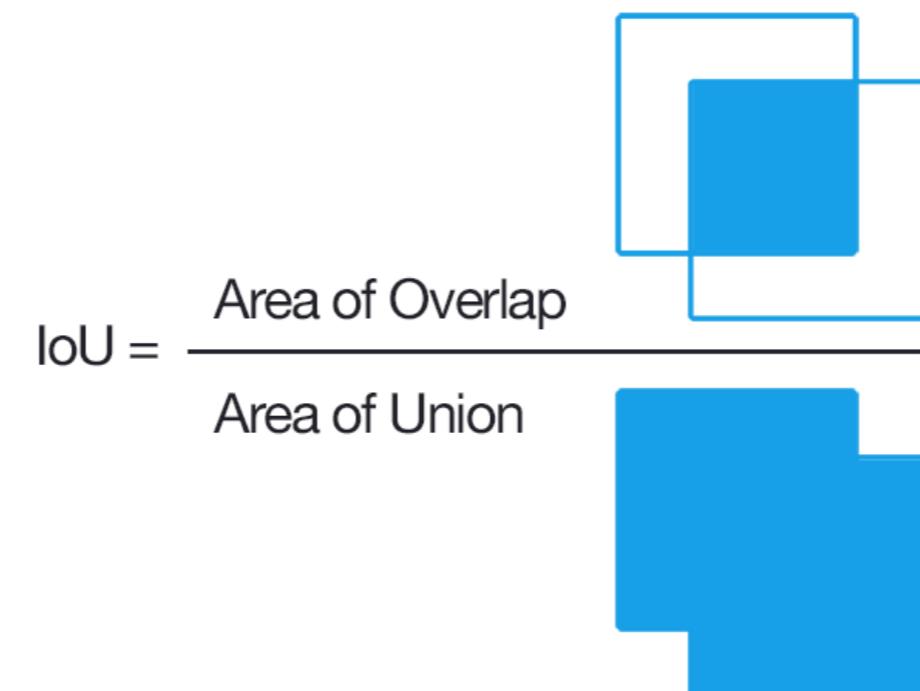
## Семантическая сегментация

**часто: segmentation accuracy**

(для класса – обычная точность по пикселям)

**Меры качества  
как для множеств**

**коэффициент Жаккара / IoU**

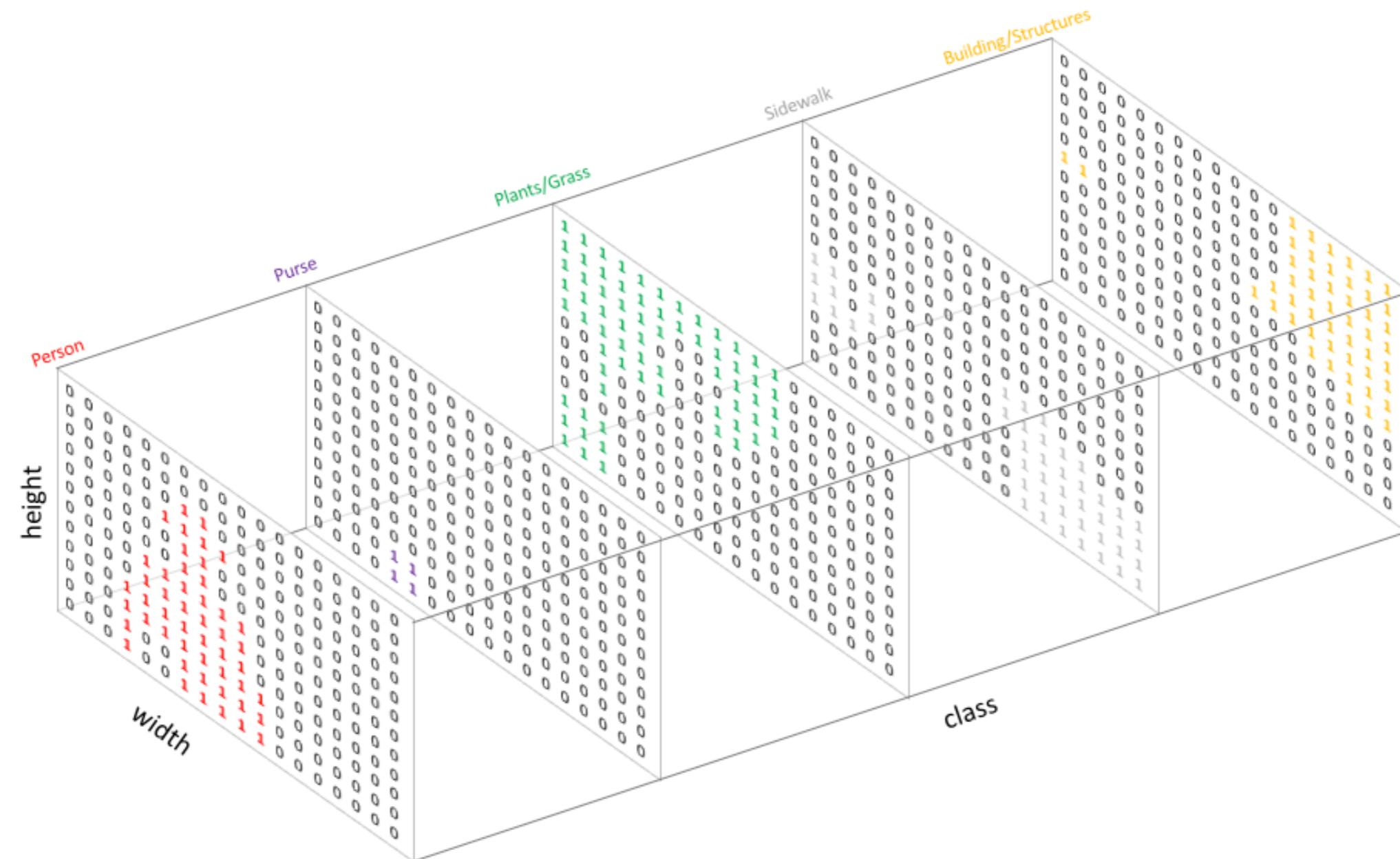


## Семантическая сегментация: приложения

- беспилотное вождение
- медицинские изображения
- изображения со спутников
- для других задач (например ИИ)
- извлечение изображения отдельных объектов (киноиндустрия)

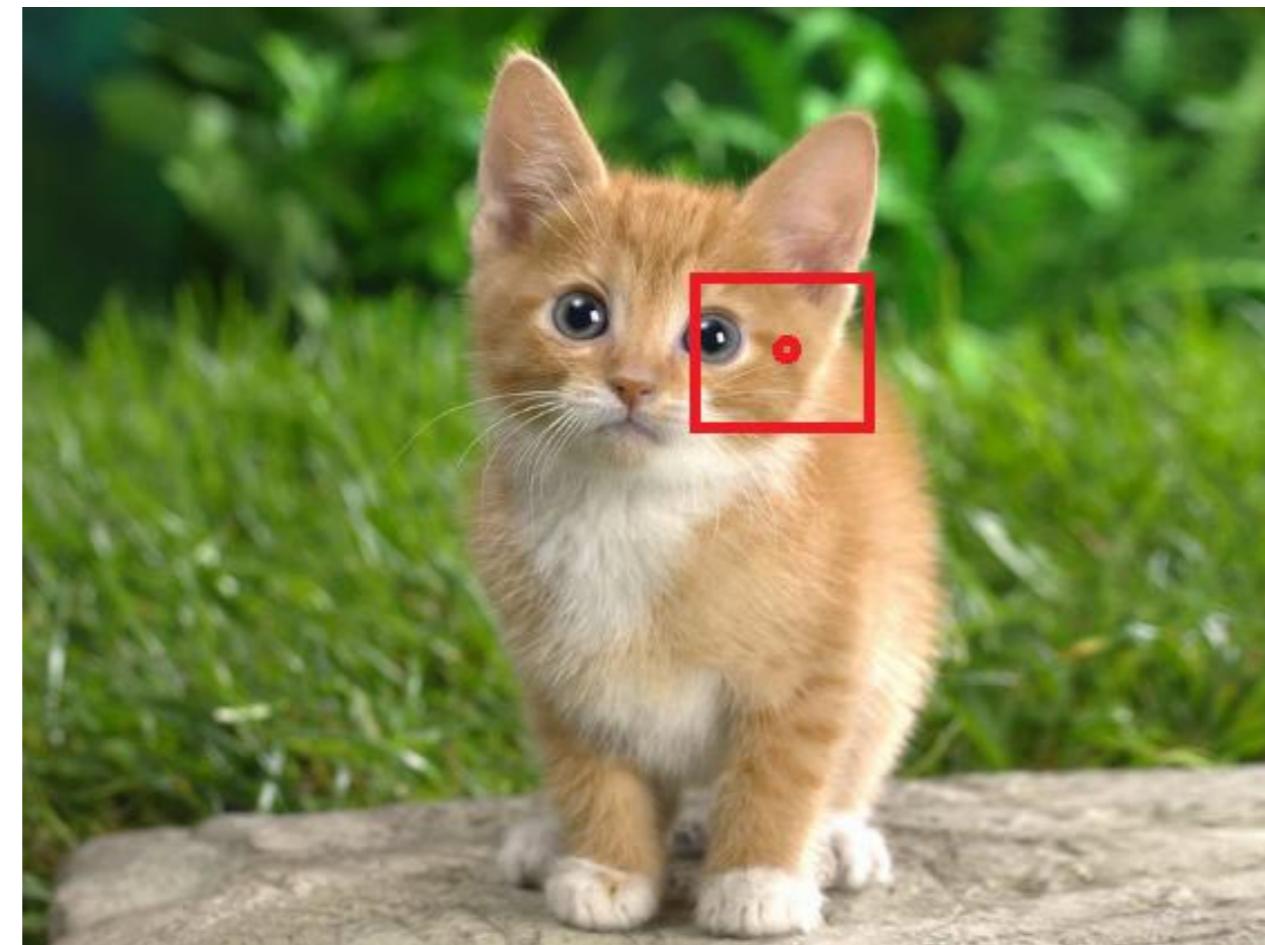
## Семантическая сегментация

### ОНЕ-кодирование целевого вектора



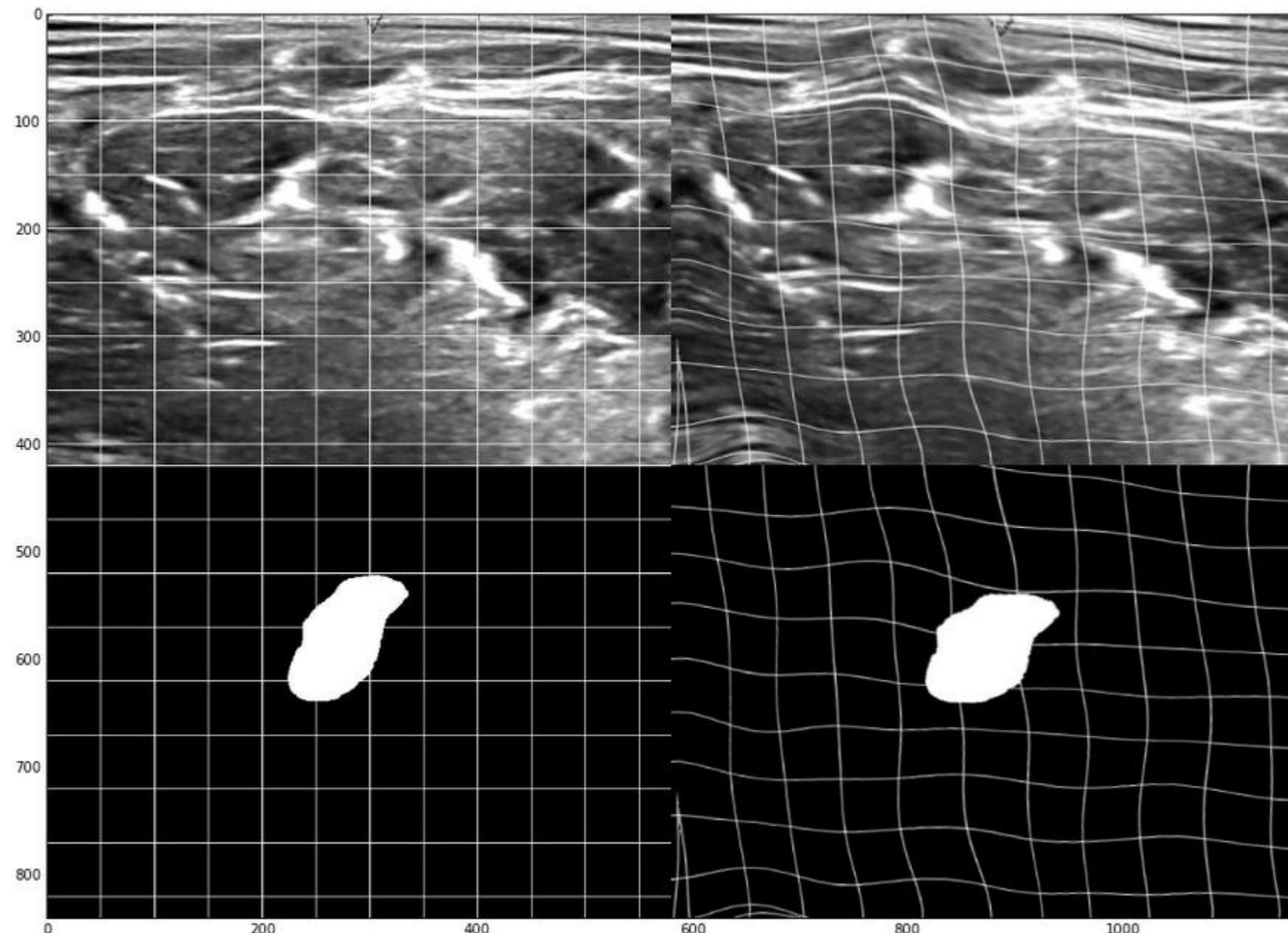
## Семантическая сегментация

**Надо расставить метки классов пикселям... Примитивная идея:**



**Метка по классификации окна с центром в этом пикселе. Очень долго...**

## Elastic Transform



<https://www.kaggle.com/bguberfain/elastic-transform-for-data-augmentation>

## Классические методы сегментации

**кластеризация по цвету**

+ объединения соседних схожих регионов

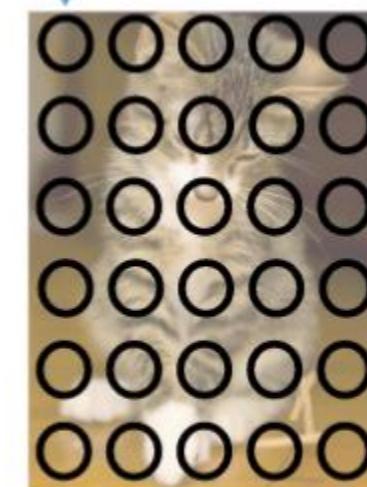
**условные случайные поля (CRF)**

**объекты непрерывны  $\Rightarrow$  соседние пиксели должны иметь одинаковые метки**



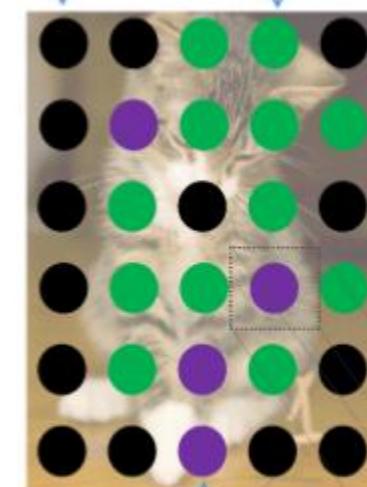
(a)

$$X_1 \in \{\text{bg, cat, dog, person}\}$$



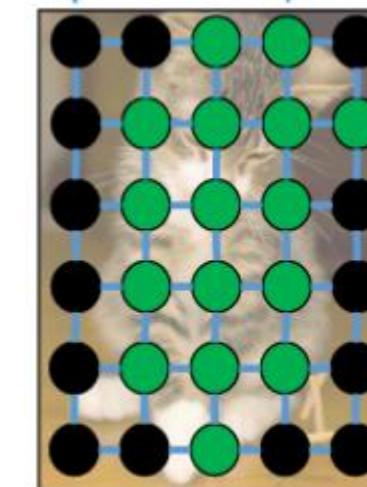
(b)

$$X_1 = \text{bg} \quad X_4 = \text{cat}$$



(c)

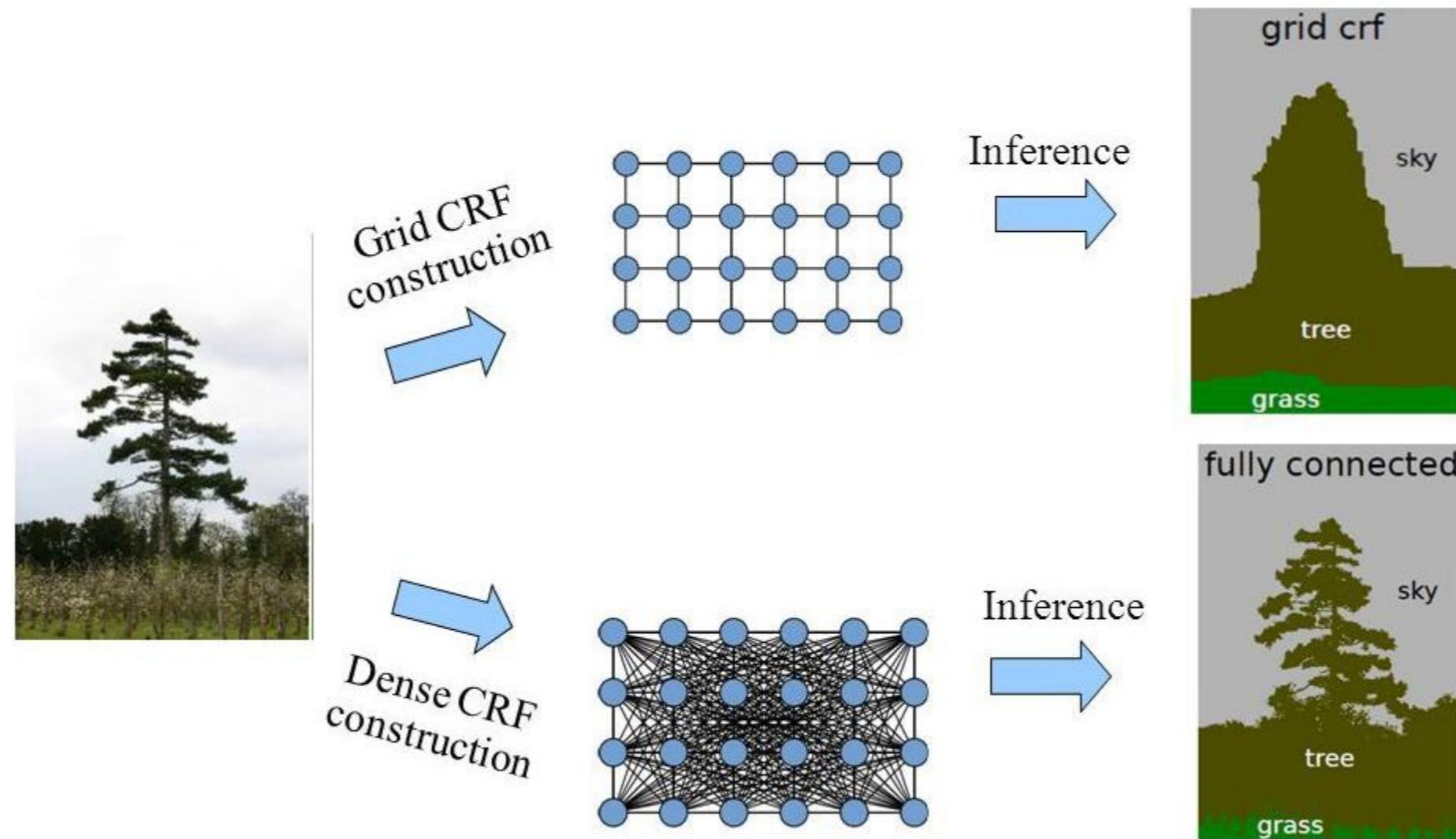
$$X_1 = \text{bg} \quad X_4 = \text{cat}$$



(d)

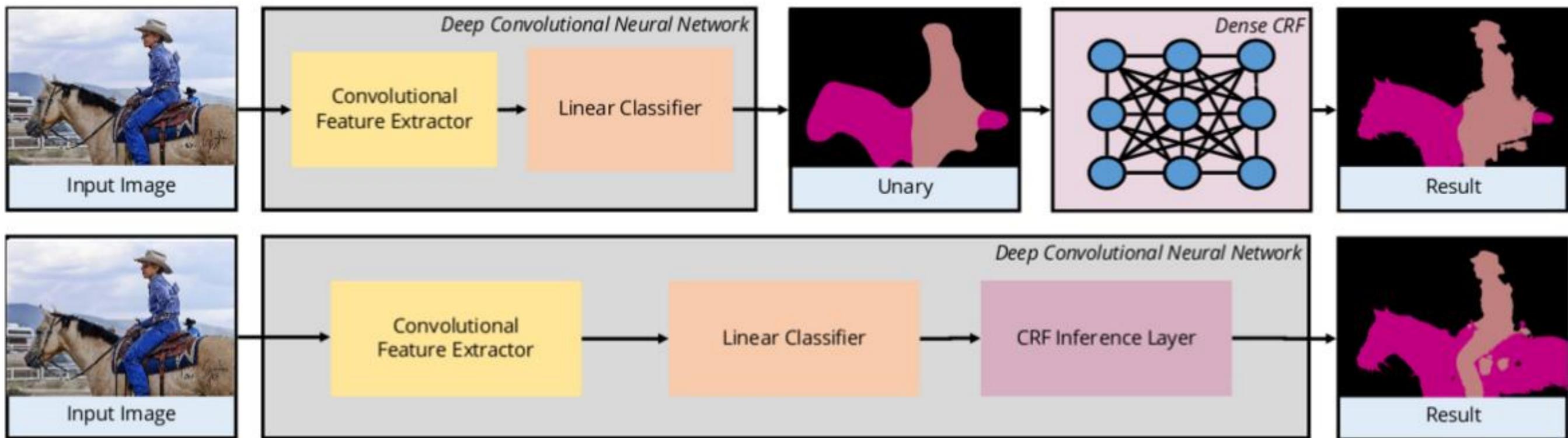
<https://neurohive.io/ru/osnovy-data-science/semantic-segmentation/>

## Классические методы сегментации



**в CRF можно подавать результат работы НС!**

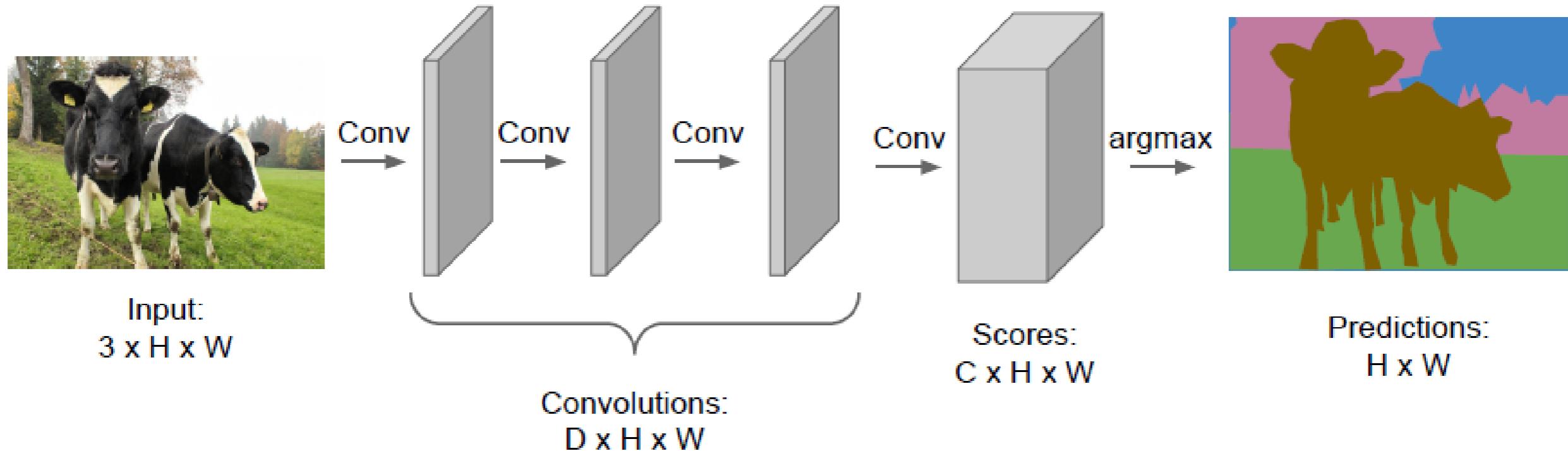
## Гибридный метод: DL + CRF



<https://neurohive.io/ru/osnovy-data-science/semantic-segmentation/>

## Полностью свёрточная сеть – Fully Convolutional Network (FCN)

Идея: полностью свёрточная структура сети –FCN



Проблемы: высокое разрешение исходного изображения

- надо его понизить (downsampling)
  - pooling
  - strided convolution
- потом надо восстановить (upsampling)

<http://cs231n.stanford.edu>

# FCN: восстановление изображения

## Как восстанавливать изображение?

**Nearest Neighbor**

1	2
3	4



Input: 2 x 2

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

Output: 4 x 4

**“Bed of Nails”**

1	2
3	4



Input: 2 x 2

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

Output: 4 x 4

**Max Pooling**

Remember which element was max!

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



Input: 4 x 4

Output: 2 x 2

**Max Unpooling**  
Use positions from  
pooling layer

1	2
3	4



Input: 2 x 2

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

Output: 4 x 4

## Обратная свёртка (upconvolution / conv-transpose)

**Напомним...**

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} * \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \underbrace{\begin{pmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 \\ 0 & 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} \end{pmatrix}}_H \cdot \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix}$$

## Обратная свёртка (upconvolution / conv-transpose)

Можно...

$$\begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix} *^{\text{т}} \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = H^{\text{т}} \cdot \begin{pmatrix} z_{11} \\ z_{21} \\ z_{12} \\ z_{22} \end{pmatrix}$$

**Обратная свёртка увеличивает пространственное разрешение...**

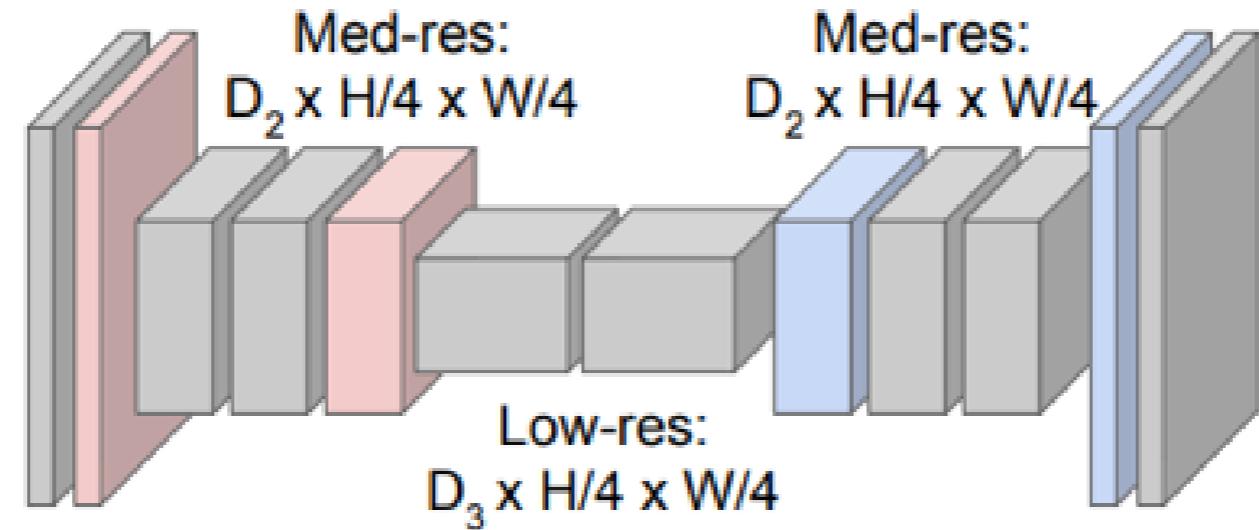
## Семантическая сегментация

В итоге что-то такое...



Input:  
 $3 \times H \times W$

High-res:  
 $D_1 \times H/2 \times W/2$

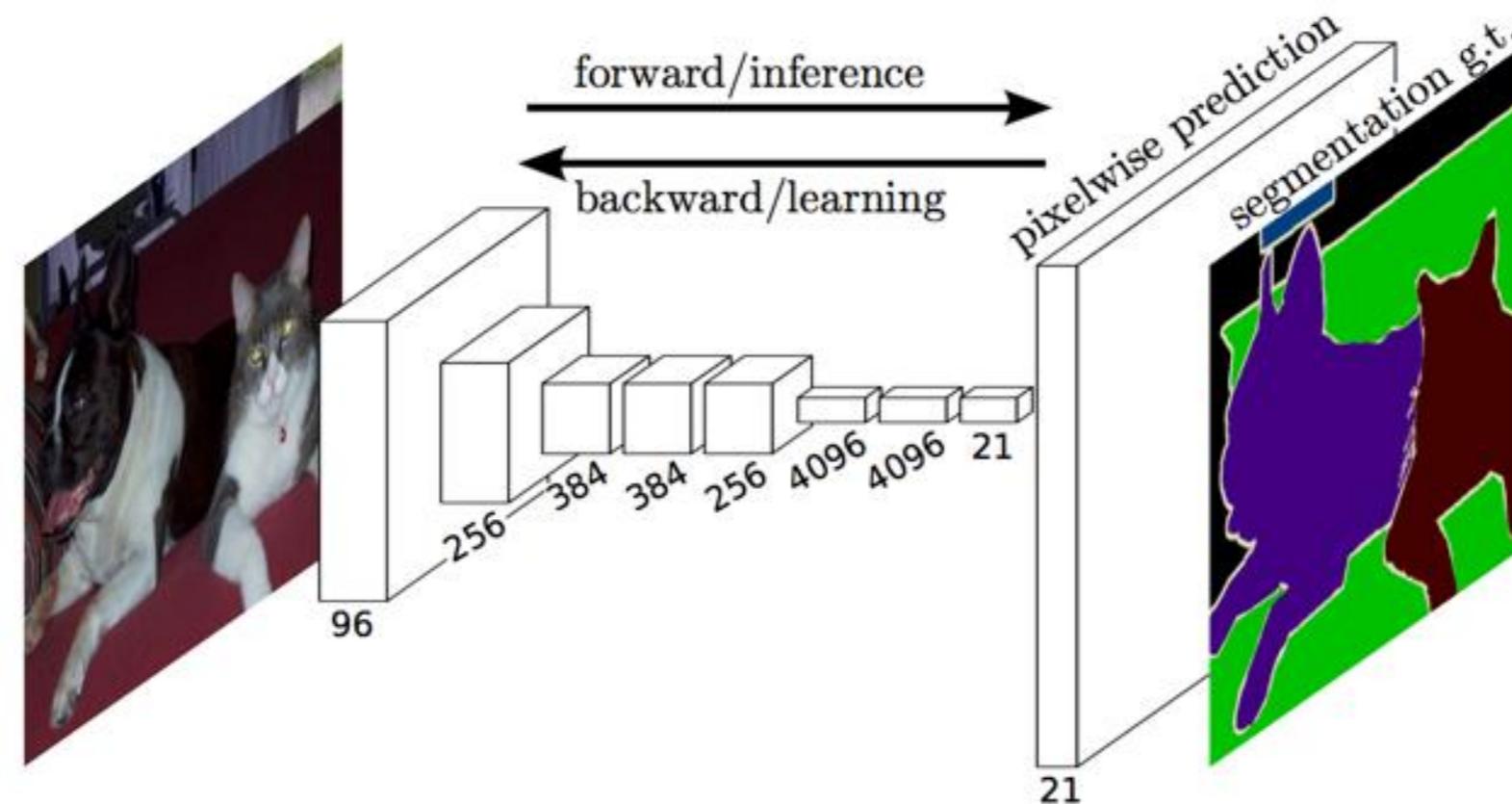


Predictions:  
 $H \times W$

уменьшаются размеры, но увеличивается число каналов!

## Fully Convolutional Networks (FCN)

### Первая успешная попытка..



«Fully Convolutional Networks for Semantic Segmentation» [Long et al., 2015  
[https://people.eecs.berkeley.edu/~jonlong/long\\_shelhamer\\_fcn.pdf](https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf)]

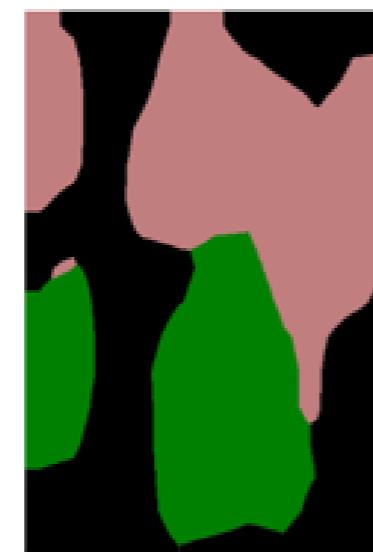
## Fully Convolutional Networks (FCN)

### Проблема

Ground truth target



Predicted segmentation



### Сейчас сделаем так...

Ground truth target



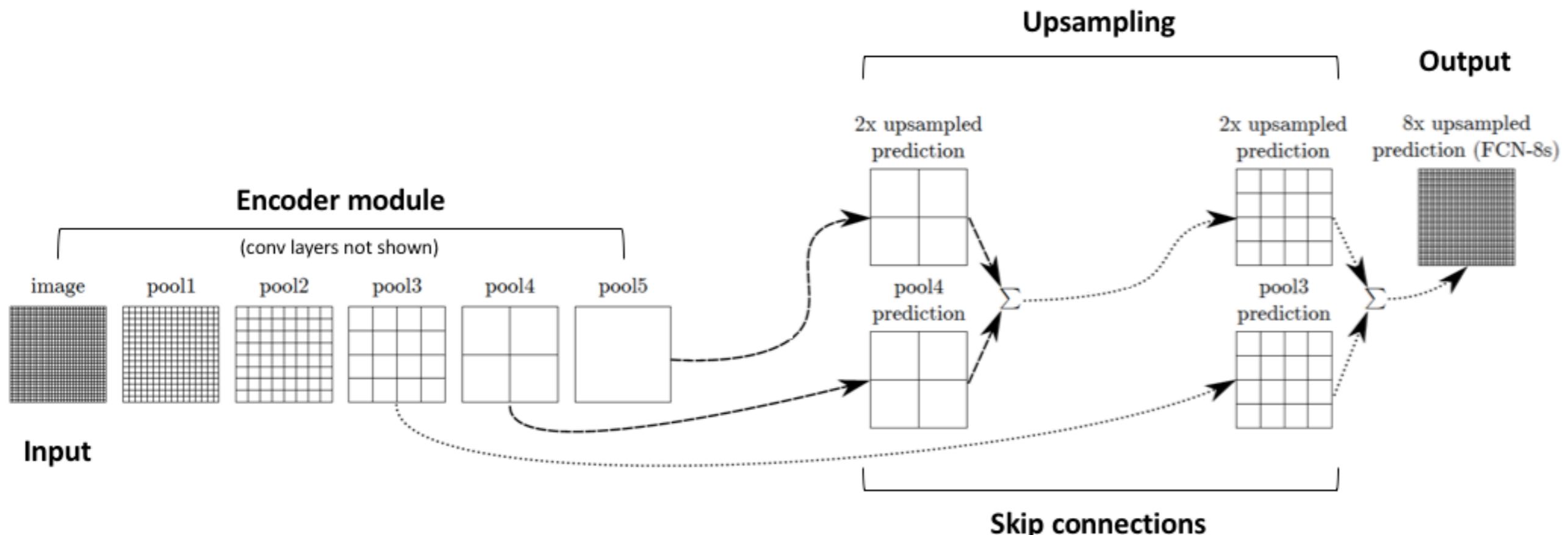
Predicted segmentation



**Понимаем «что» (на последних слоях),  
но забываем «где» (было на первых)**

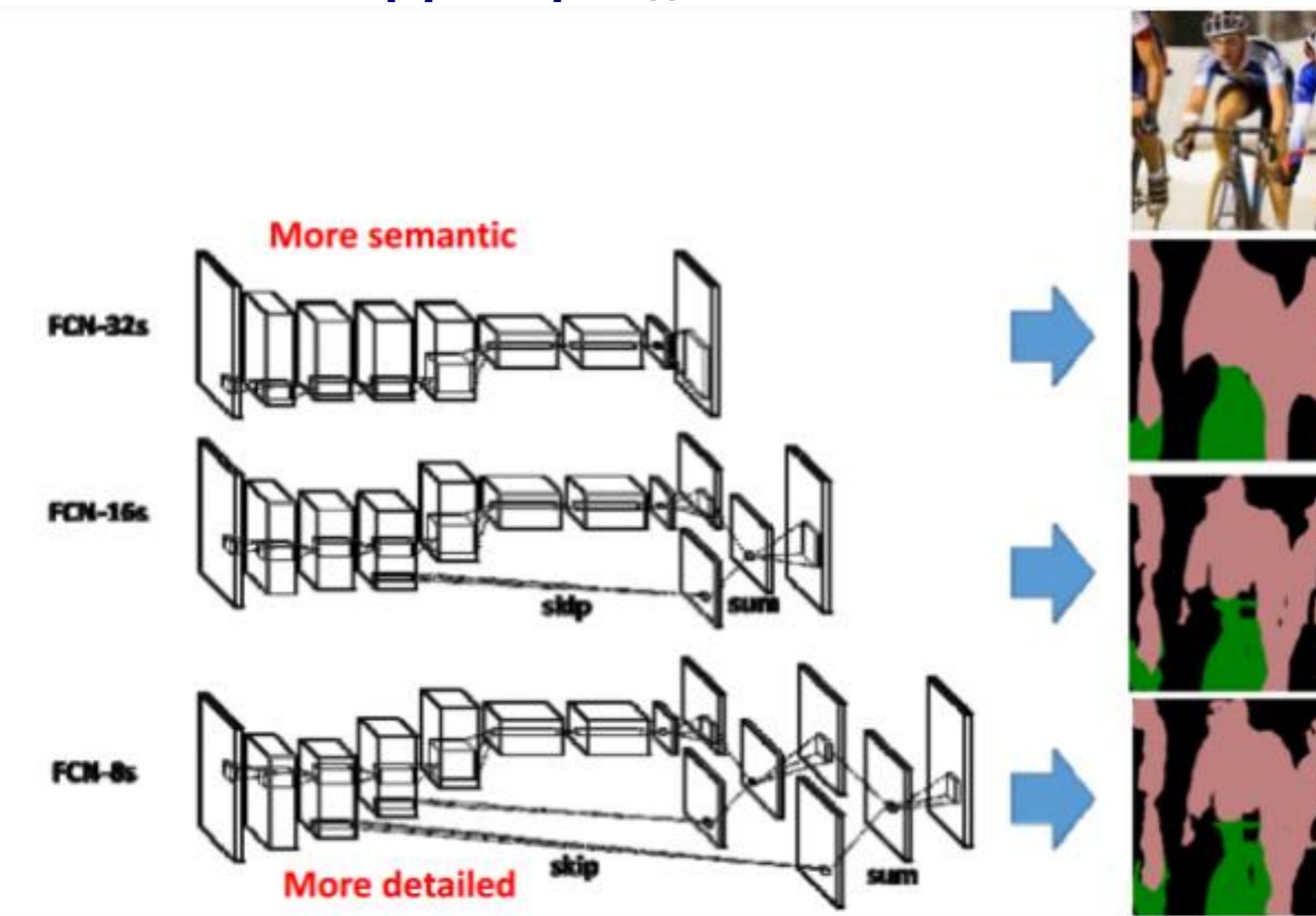
## Fully Convolutional Networks (FCN)

### Решение – прокидывание связей (skip connections)

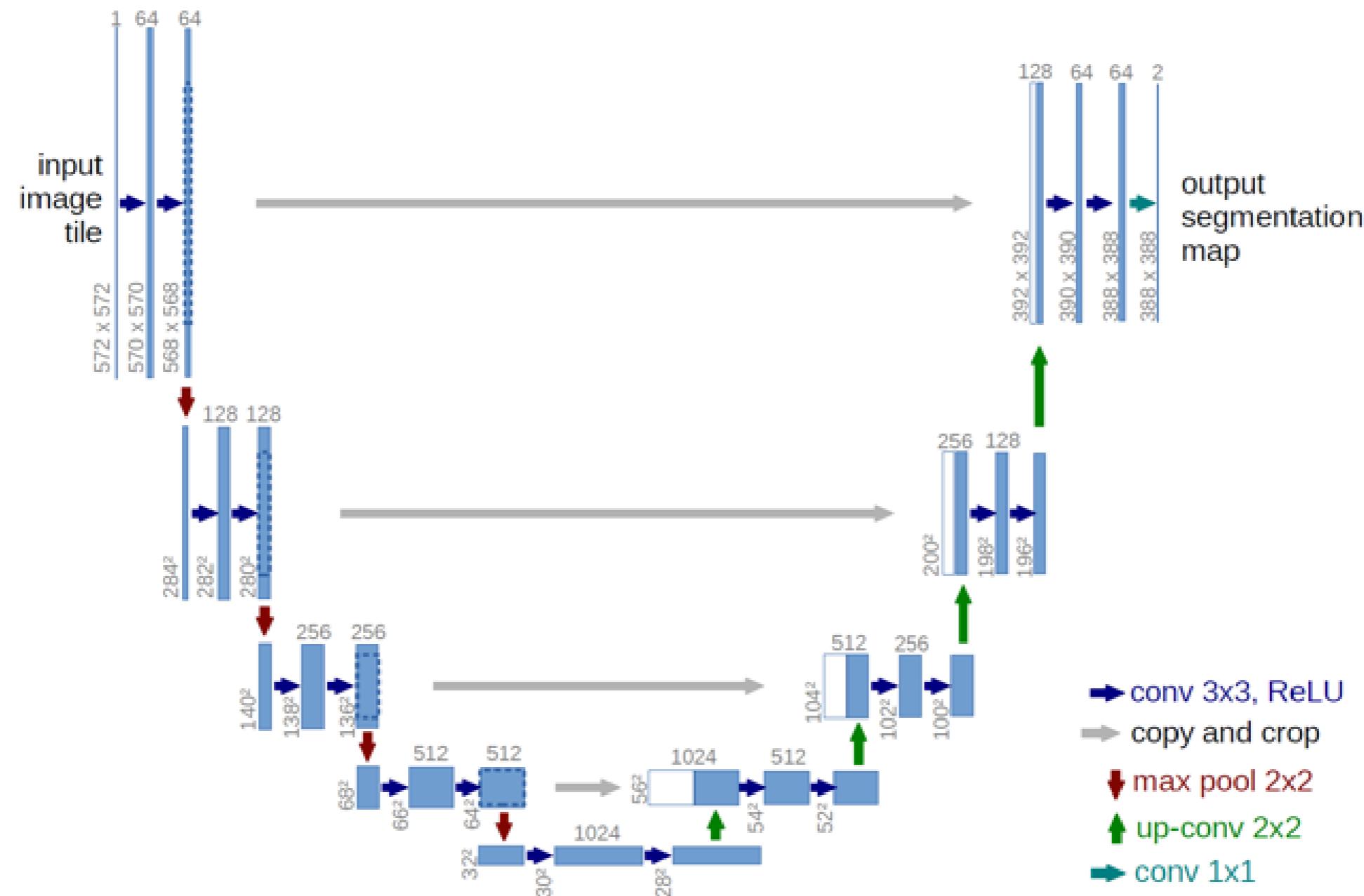


- шахматные эффекты
- низкое разрешение на краях

## Эффект прокидывания связей



## U-Net (самая популярная)



«U-Net: Convolutional Networks for Biomedical Image Segmentation» [Ronneberger O. и др., 2015 <https://arxiv.org/abs/1505.04597>】

## U-Net (самая популярная сейчас)

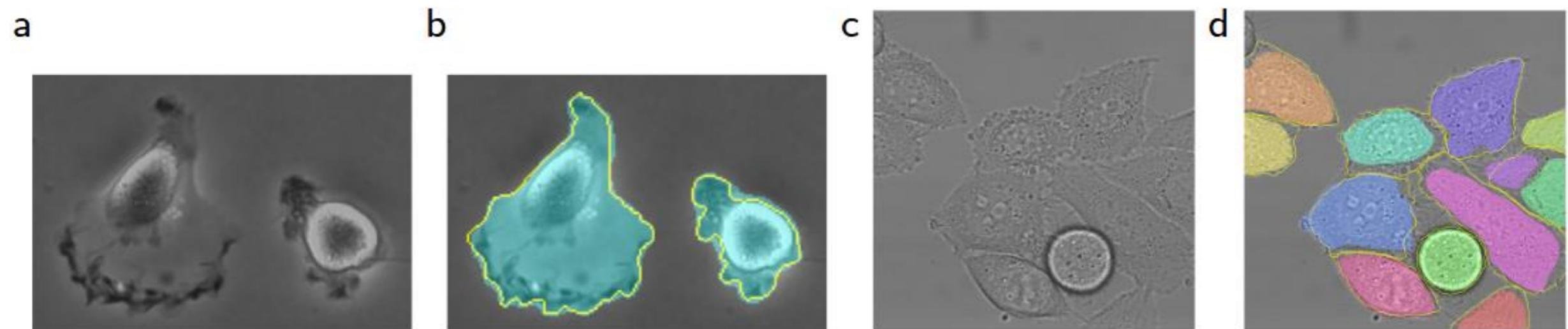
Левая часть

**3×3-свёртки, ReLU, 2×2-пулинг (шаг=2)**

Правая часть

**2×2 обратная свёртка (половинит число каналов), конкатенация с обрезанными признаками из левой части (там больше H×W), 3×3-свёртки, ReLU**

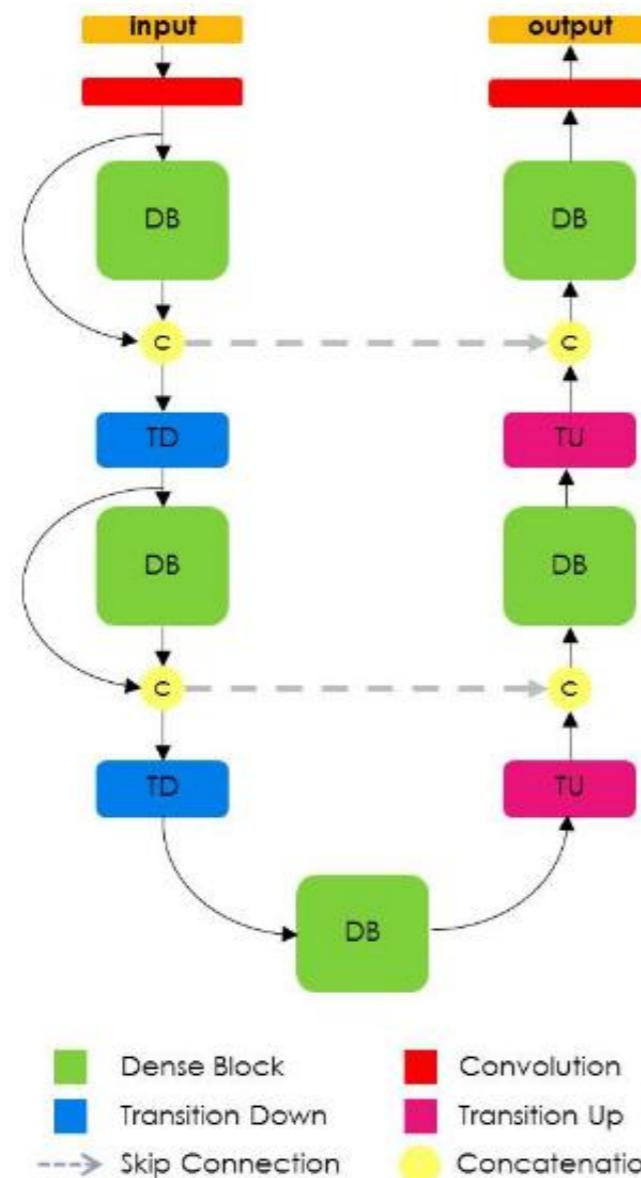
**всего 23 свёрточных слоя, реализация – Caffe**



**Польза аугментации в некоторых задачах!**

**Это была фишка статьи – данных можно сделать много!**

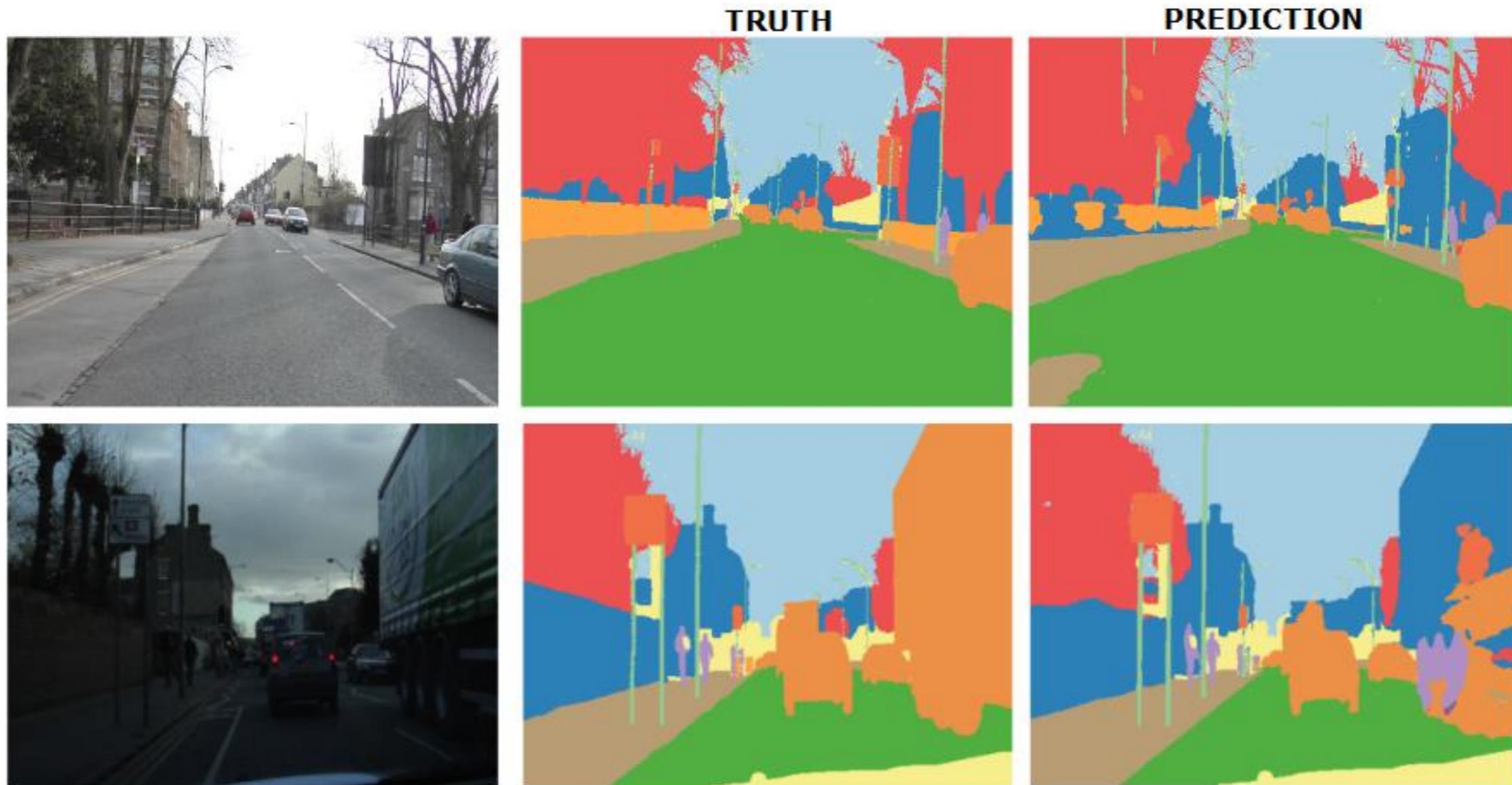
## «Тирамису» = DenseNet + U-Net



**HeUniform + RMSprop + weight  
Dropout + BN**

Input, $m = 3$
$3 \times 3$ Convolution, $m = 48$
DB (4 layers) + TD, $m = 112$
DB (5 layers) + TD, $m = 192$
DB (7 layers) + TD, $m = 304$
DB (10 layers) + TD, $m = 464$
DB (12 layers) + TD, $m = 656$
DB (15 layers), $m = 896$
TU + DB (12 layers), $m = 1088$
TU + DB (10 layers), $m = 816$
TU + DB (7 layers), $m = 578$
TU + DB (5 layers), $m = 384$
TU + DB (4 layers), $m = 256$
$1 \times 1$ Convolution, $m = c$
Softmax

«The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation» [Jégou S. и др., 2017 <https://arxiv.org/abs/1611.09326>]

**«Тирамису» = DenseNet + U-Net**

– требует больших мощностей

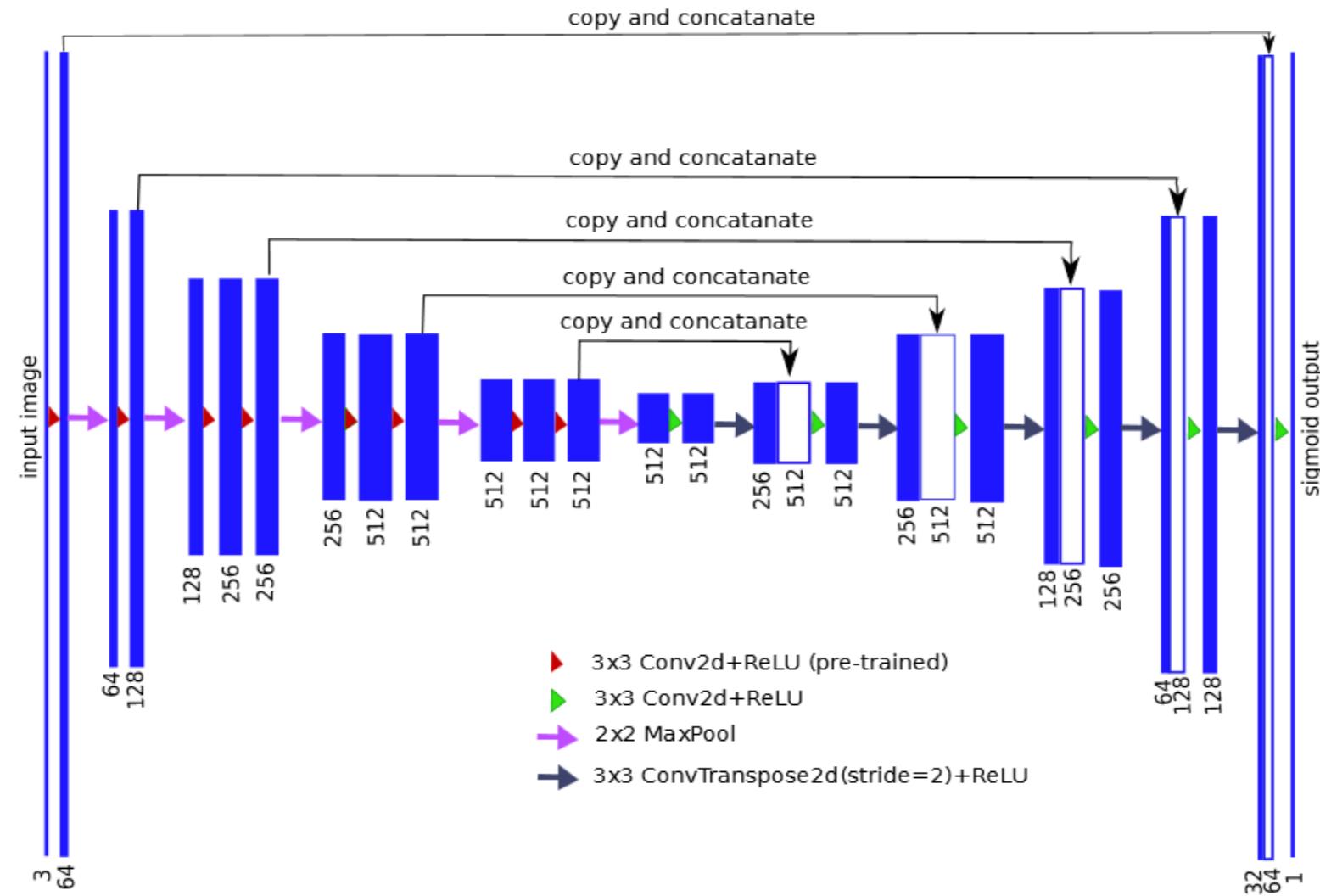
**TernausNet**

Fig. 1. Encoder-decoder neural network architecture also known as U-Net where VGG11 neural network without fully connected layers as its encoder. Each blue rectangular block represents a multi-channel features map passing through a series of transformations. The height of the rod shows a relative map size (in pixels), while their widths are proportional to the number of channels (the number is explicitly subscribed to the corresponding rod). The number of channels increases stage by stage on the left part while decrease stage by stage on the right decoding part. The arrows on top show transfer of information from each encoding layer and concatenating it to a corresponding decoding layer.

**использование обученной VGG11 в качестве кодировщика** <https://arxiv.org/abs/1801.05746>

## PSP-Net = Pyramid Scene Parsing Network

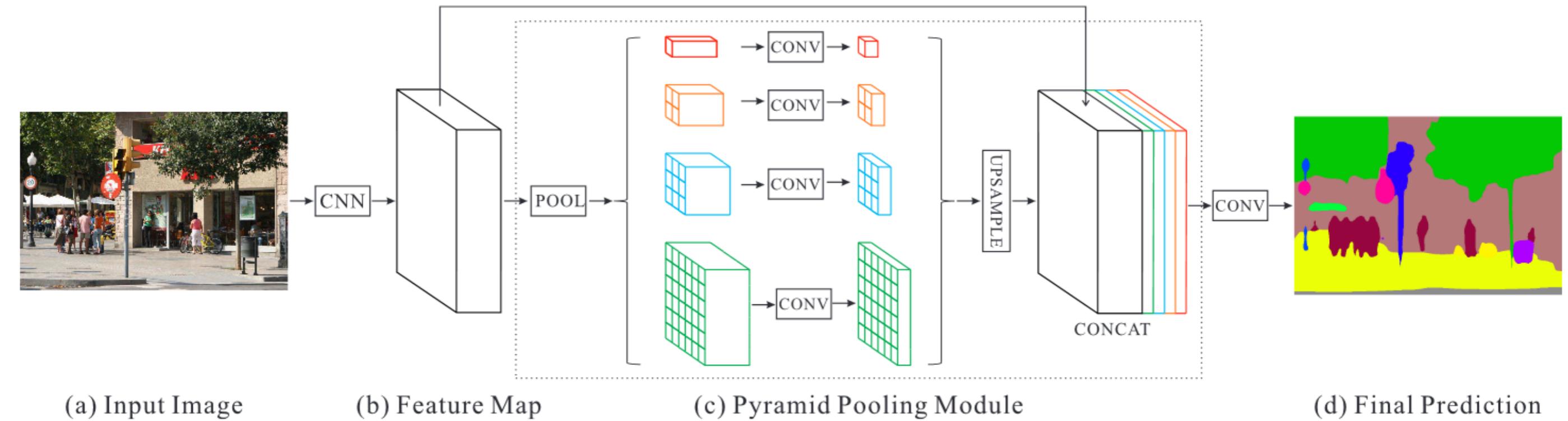


Figure 3. Overview of our proposed PSPNet. Given an input image (a), we first use CNN to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d).

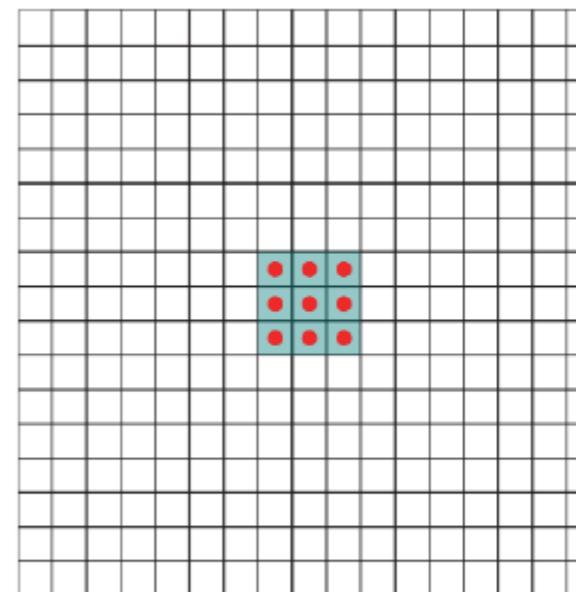
## Расширенные свёртки (Dilated convolutions / Atrous Convolutions)

Discrete Convolution Operation:

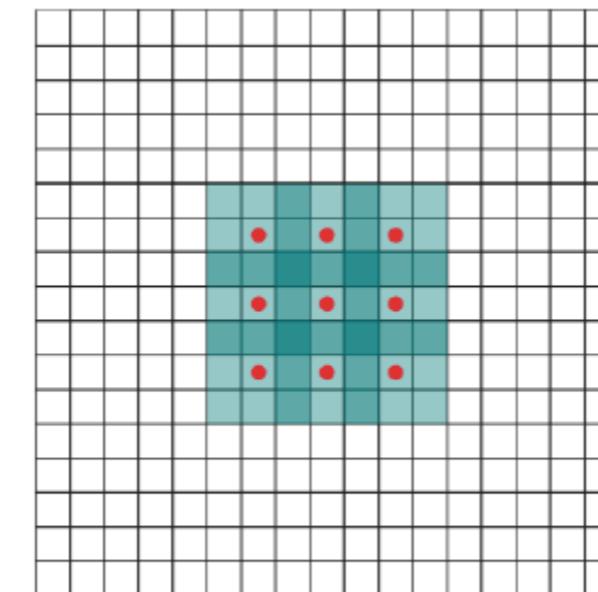
$$(F * k)(p) = \sum_{s+t=p} F(s) k(t)$$

Dilated Convolution Operation:

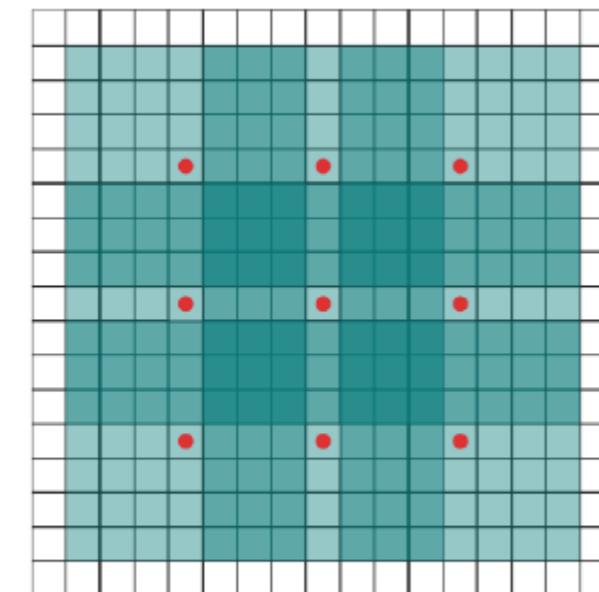
$$(F *_l k)(p) = \sum_{s+lt=p} F(s) k(t)$$



(a)



(b)



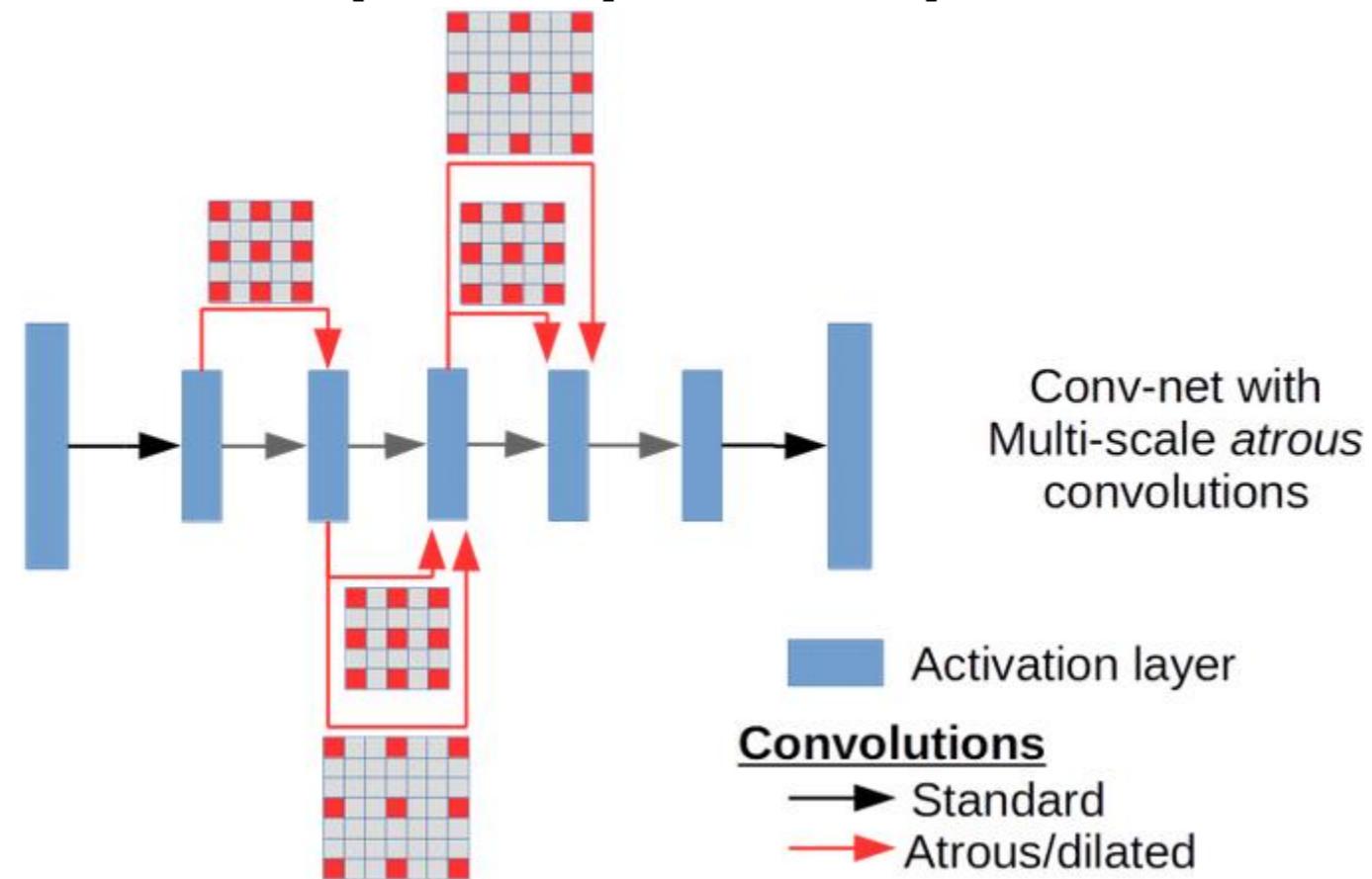
(c)

**увеличение перцептивного поля (perception field)**

<https://arxiv.org/abs/1511.07122>

## Расширенные свёртки (Dilated convolutions / Atrous Convolutions)

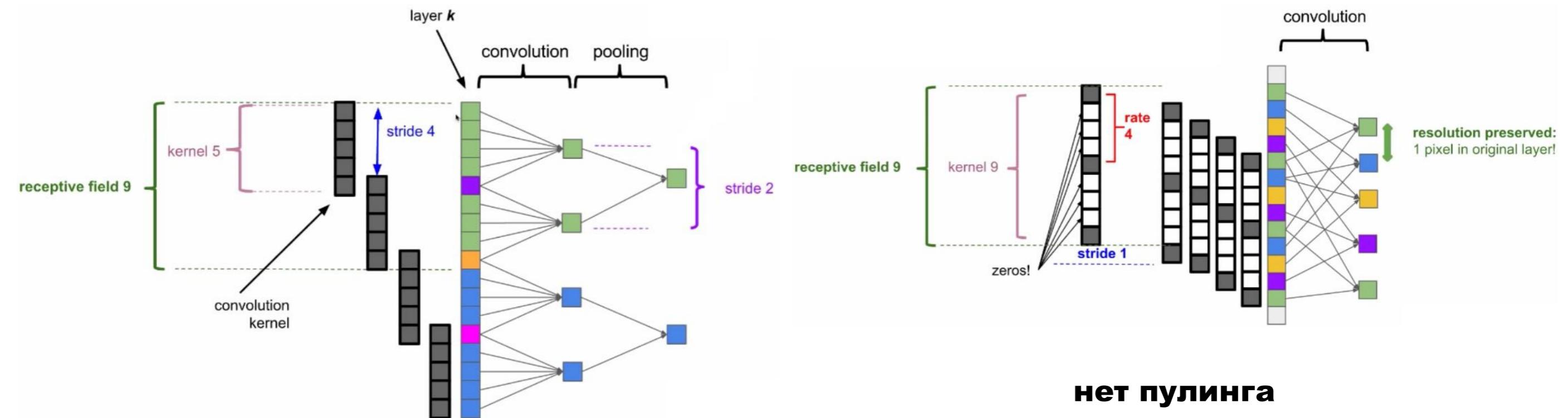
можно комбинировать признаки с разных масштабов!



также в задачах, например увеличения разрешения

<https://dzone.com/articles/atrous-convolutions-and-u-net-architectures-for-de>

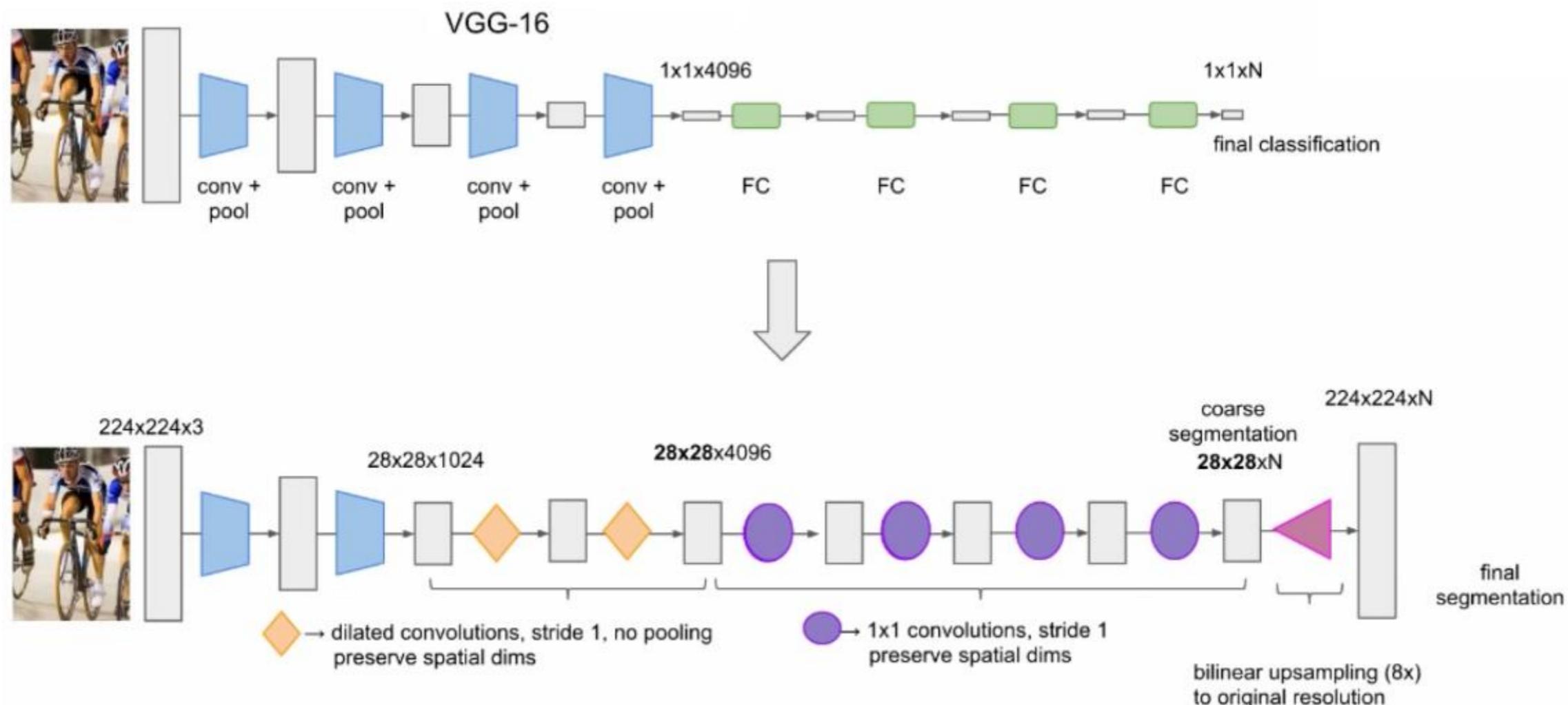
## Расширенные свёртки (Dilated convolutions / Atrous Convolutions)



**получаем большую, но разреженную свёртку  
вычислительная сложность  $\sim$  число ненулей  
при одинаковом размере модели можем сделать любое рецептивное поле**

deepsystems.io

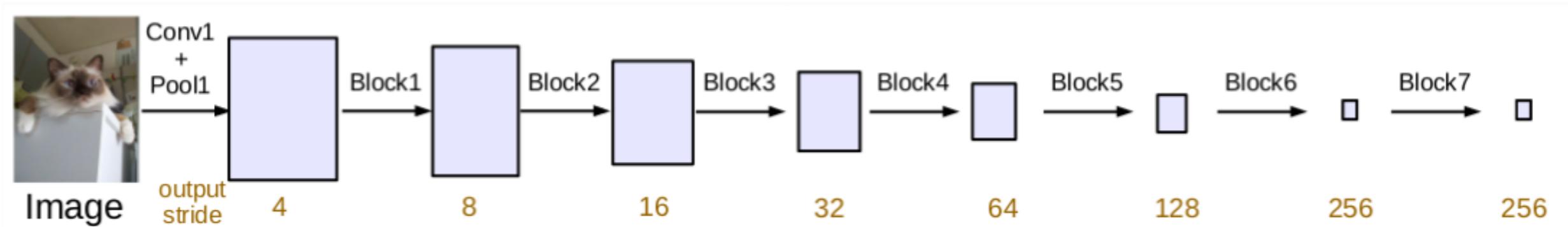
## DeepLabv1/2: использование расширенных свёрток



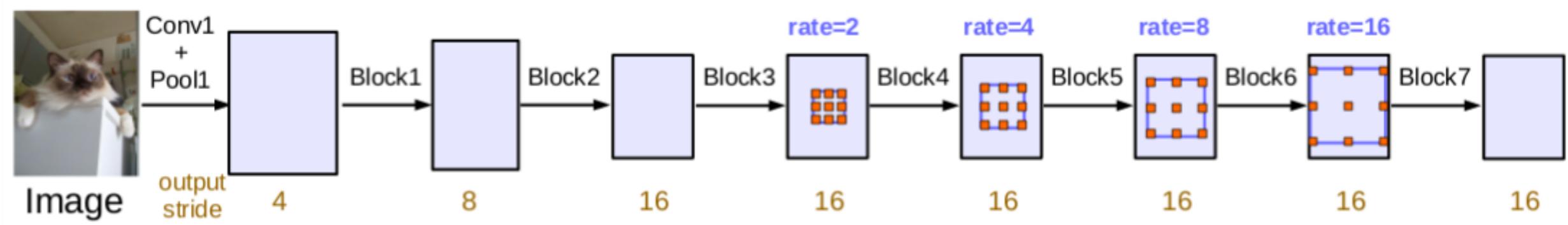
**нет пулинга – последние свёртка + пулинг заменили на расширенные свёртки (stride=1)  
постпроцессинг – CRF**

## DeepLabv3: использование расширенных свёрток

**можно комбинировать признаки с разных масштабов!**



(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with  $rate > 1$  is applied after block3 when  $output\_stride = 16$ .

Figure 3. Cascaded modules without and with atrous convolution.

Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam «Rethinking Atrous Convolution for Semantic Image Segmentation» // <https://arxiv.org/pdf/1706.05587.pdf>

## DeepLabv3: использование расширенных свёрток

### «Параллельный режим»

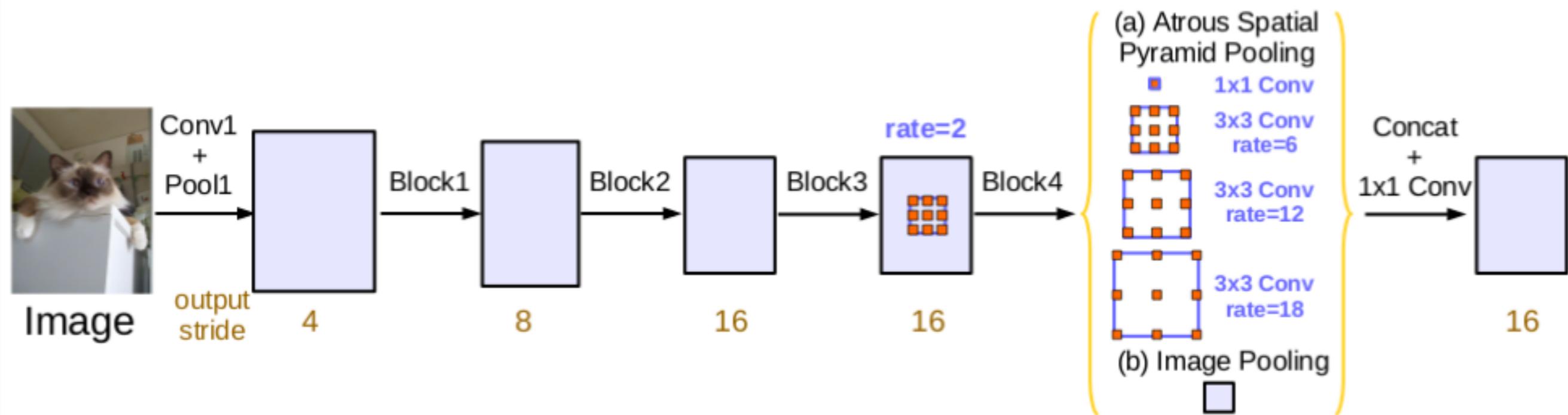
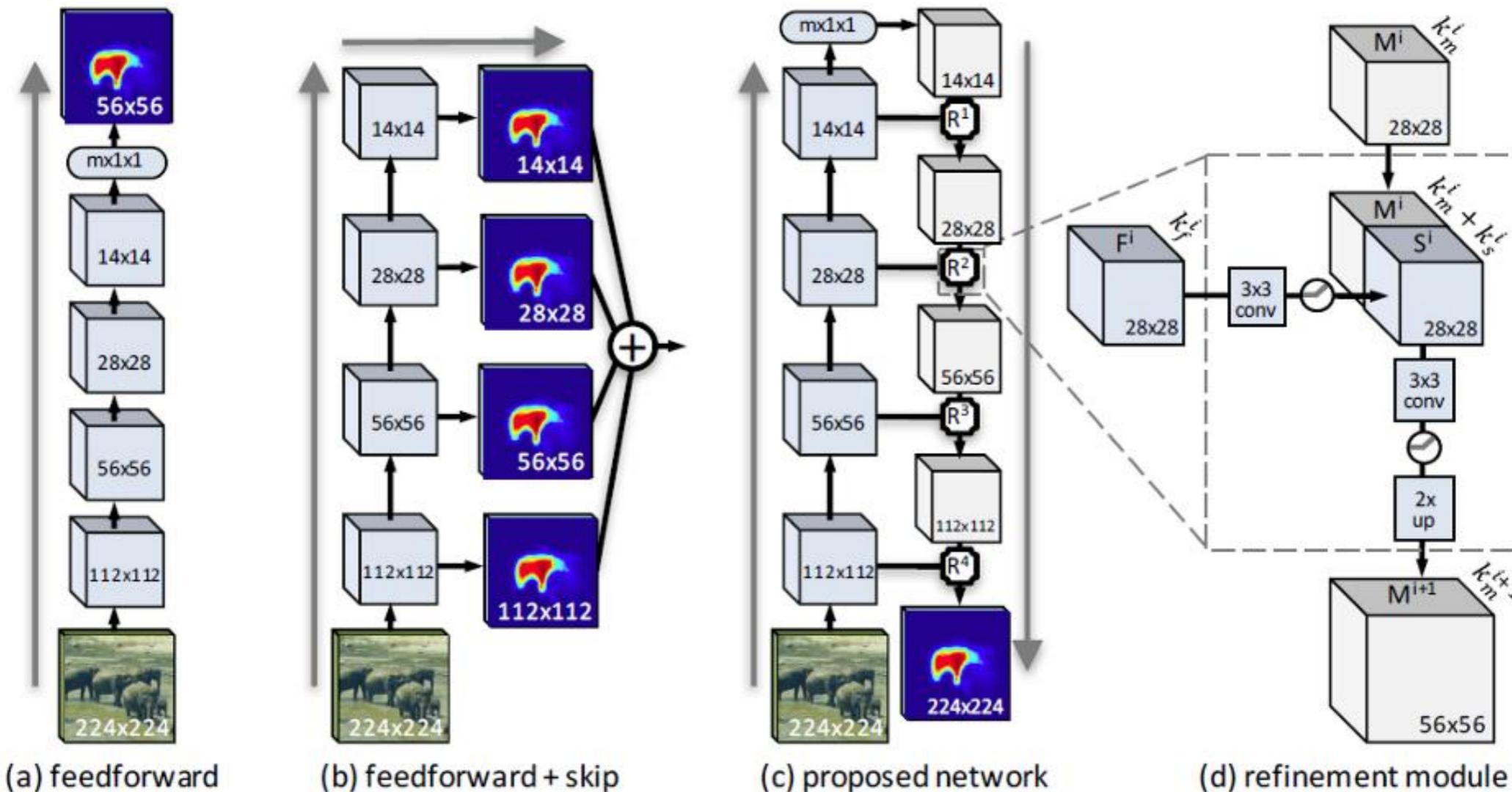


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

<https://arxiv.org/pdf/1706.05587.pdf>

## SharpMask: Top to Down Refinement encoder-decoder



Learning to refine object segments, ECCV 2016

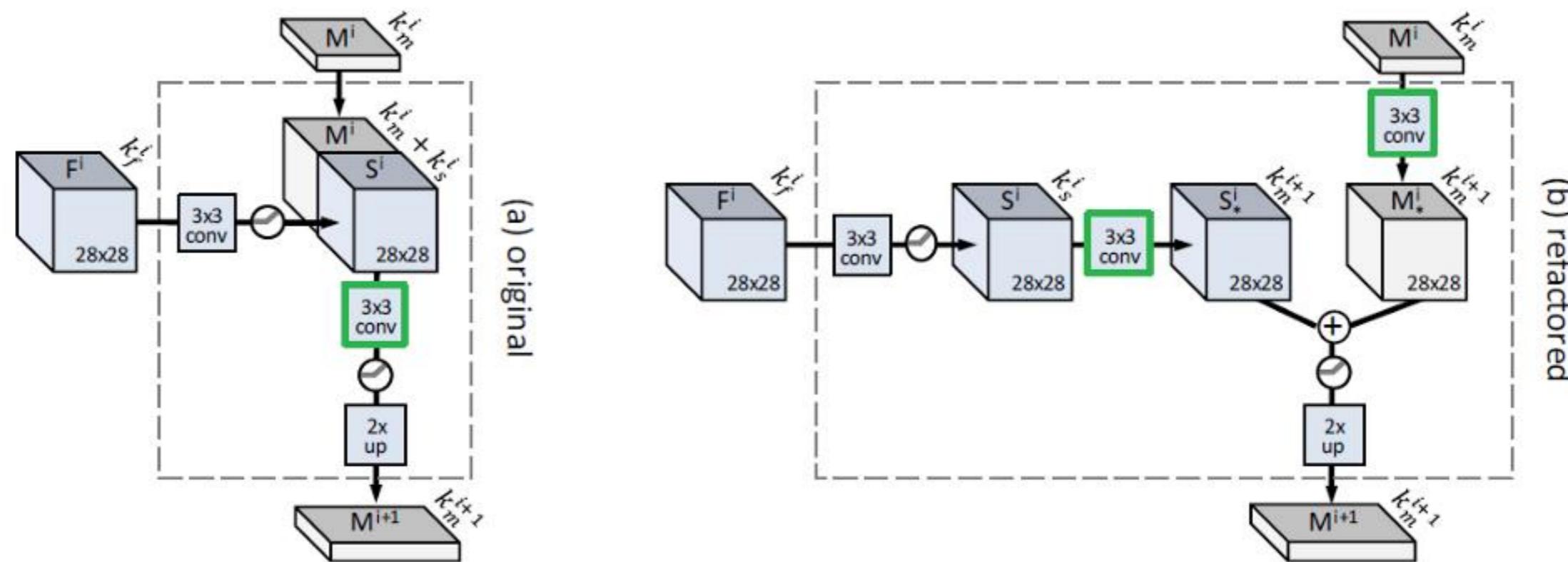
<https://towardsdatascience.com/review-sharpmask-instance-segmentation-6509f7401a61>

## SharpMask: Top to Down Refinement

На проходе вверх:

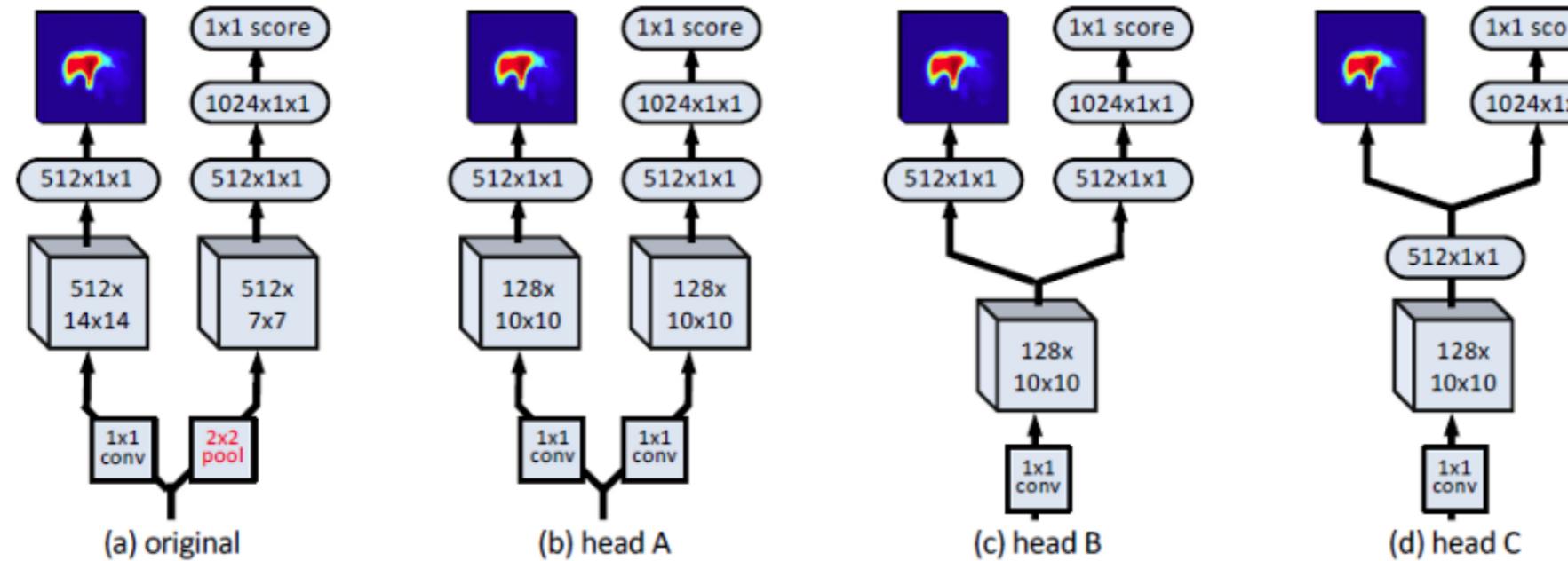
**ImageNet-Pretrained 50-layer ResNet**

На проходе вниз (top-down): **3×3 свёртки + 2× upsampled (билинейная интерполяция)**  
**Конкатенация, перед которой также 3×3 свёртки (+ сокращение числа каналов)**



справа – более эффективная реализация

## SharpMask: Head Architecture



Пробовали по-разному

Обучение

есть тонкости

сначала – проход вверх, чтобы получать грубую маску

потом он замораживается...

Работа – можно уточнять (refined) только «многообещающие» области

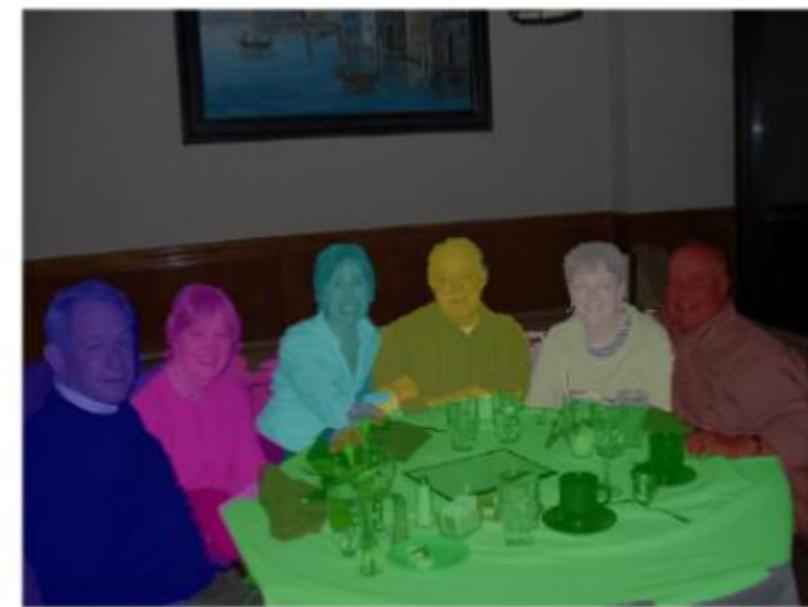
## SharpMask: примеры работы



## Сегментация объектов (Instance segmentation)



Semantic Segmentation



Instance Segmentation

**в семантической сегментации каждому пикслю – класс  
в сегментации объектов надо различать группы пикселей формально одного класса,  
но принадлежащие разным объектам**

<https://neurohive.io/ru/osnovy-data-science/semantic-segmention/>

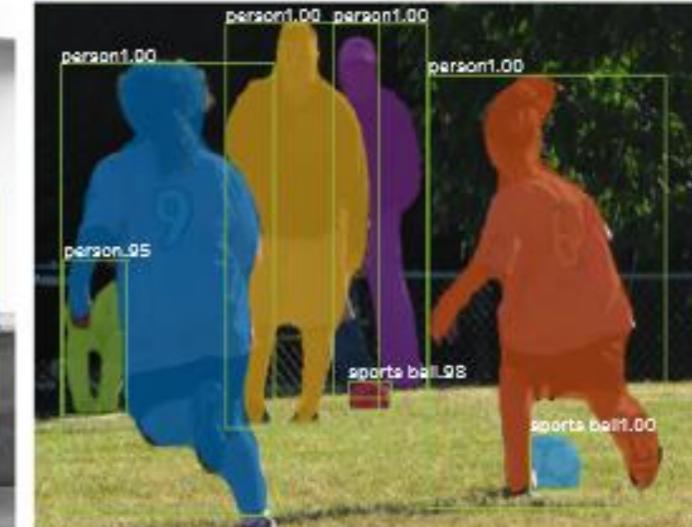
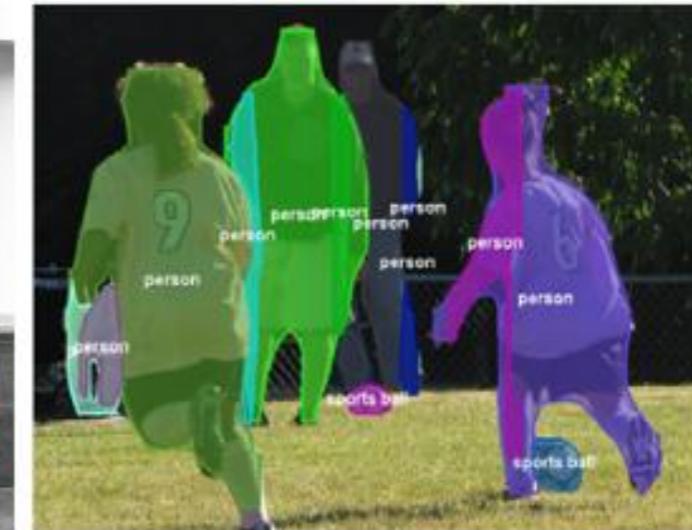
## Сегментация объектов (Instance segmentation)

не просто найти рамку, а выделить объект!

FCIS

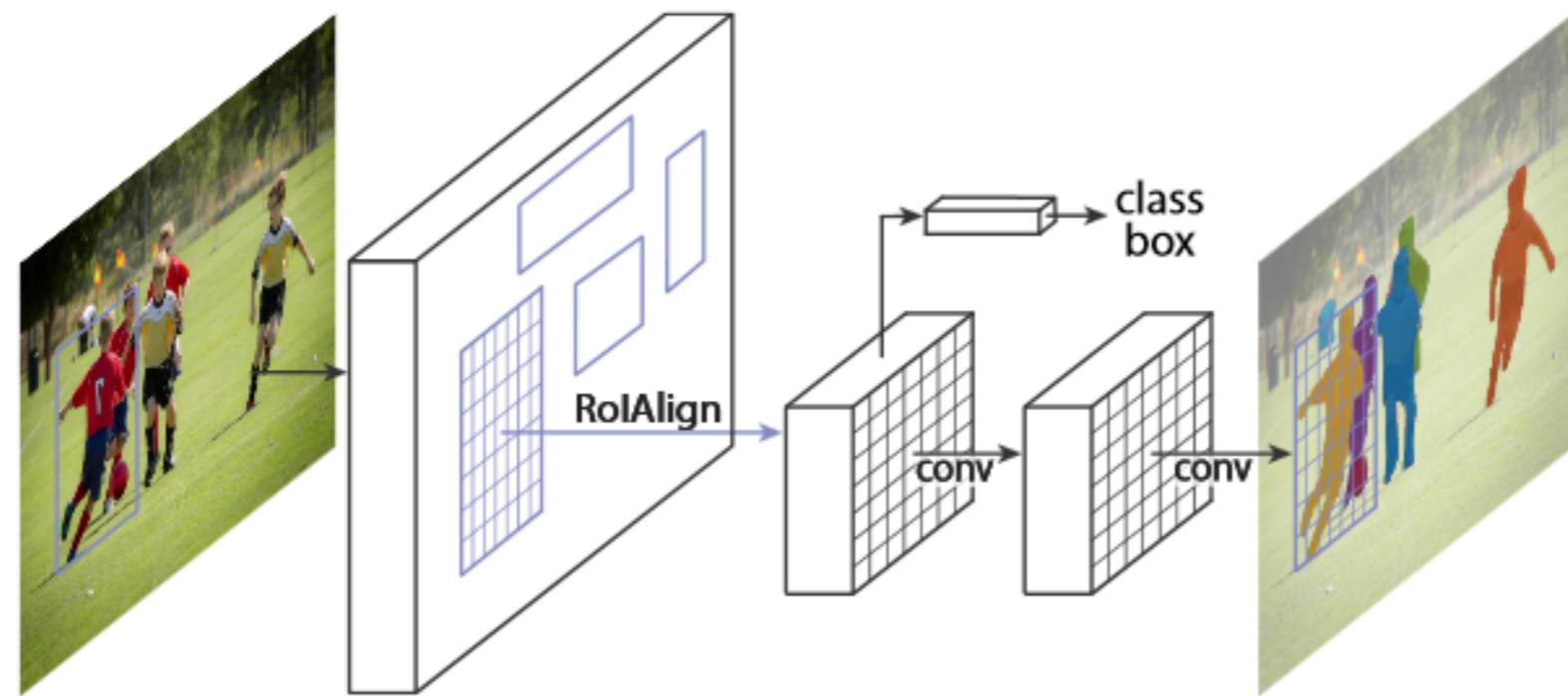


Mask R-CNN



Mask R-CNN <https://arxiv.org/pdf/1703.06870.pdf>

## Mask R-CNN



**Faster R-CNN + определение сегментационной маски (маленькая FCN)  
+ борьба за более точное определение границ**

**Хороши также для определения позы  
специальные маски для детектирования опорных точек (локоть, плечо и т.п.)  
– один пиксель =1, остальные =0**

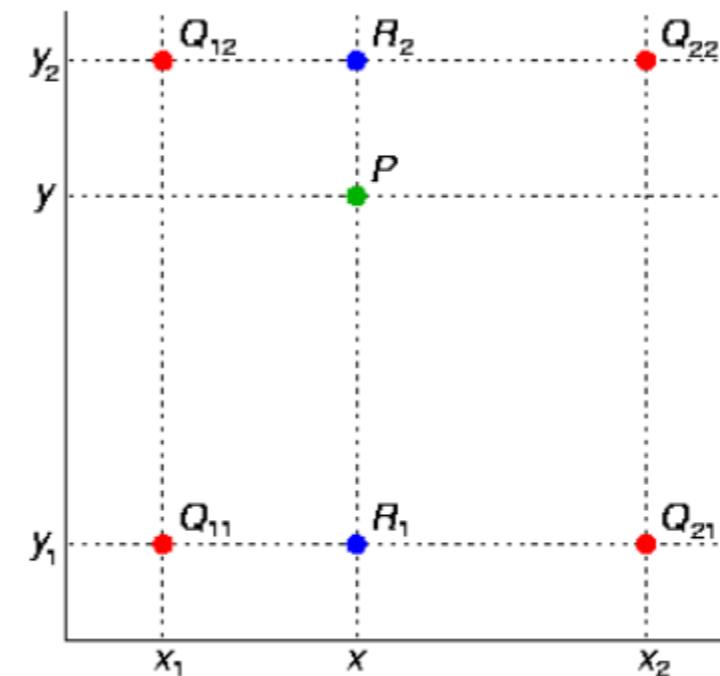
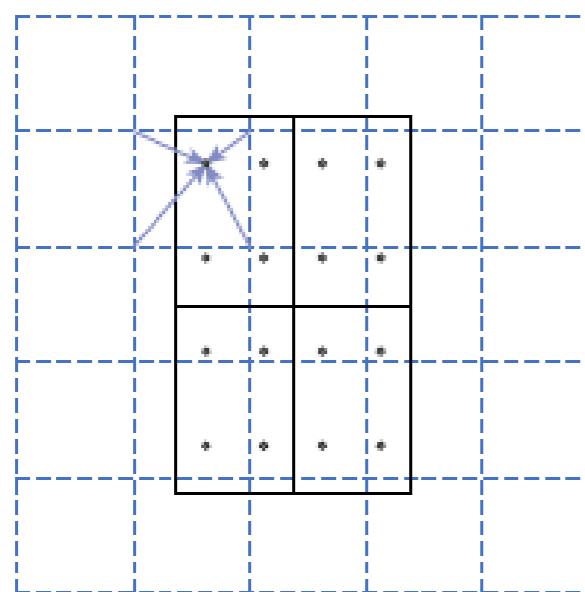
## Mask R-CNN

**маска вида 1/0 – принадлежит или нет объекту (class-agnostic)  
для опорных точек – ОНЕ таких точек**

**ошибка = сумма трёх (класс, регрессия региона, маска)**

## RoIPool → RoIAlign

**т.к. на карте признаков уменьшенного не получается точно разместить регион:  
билинейная интерполяция**



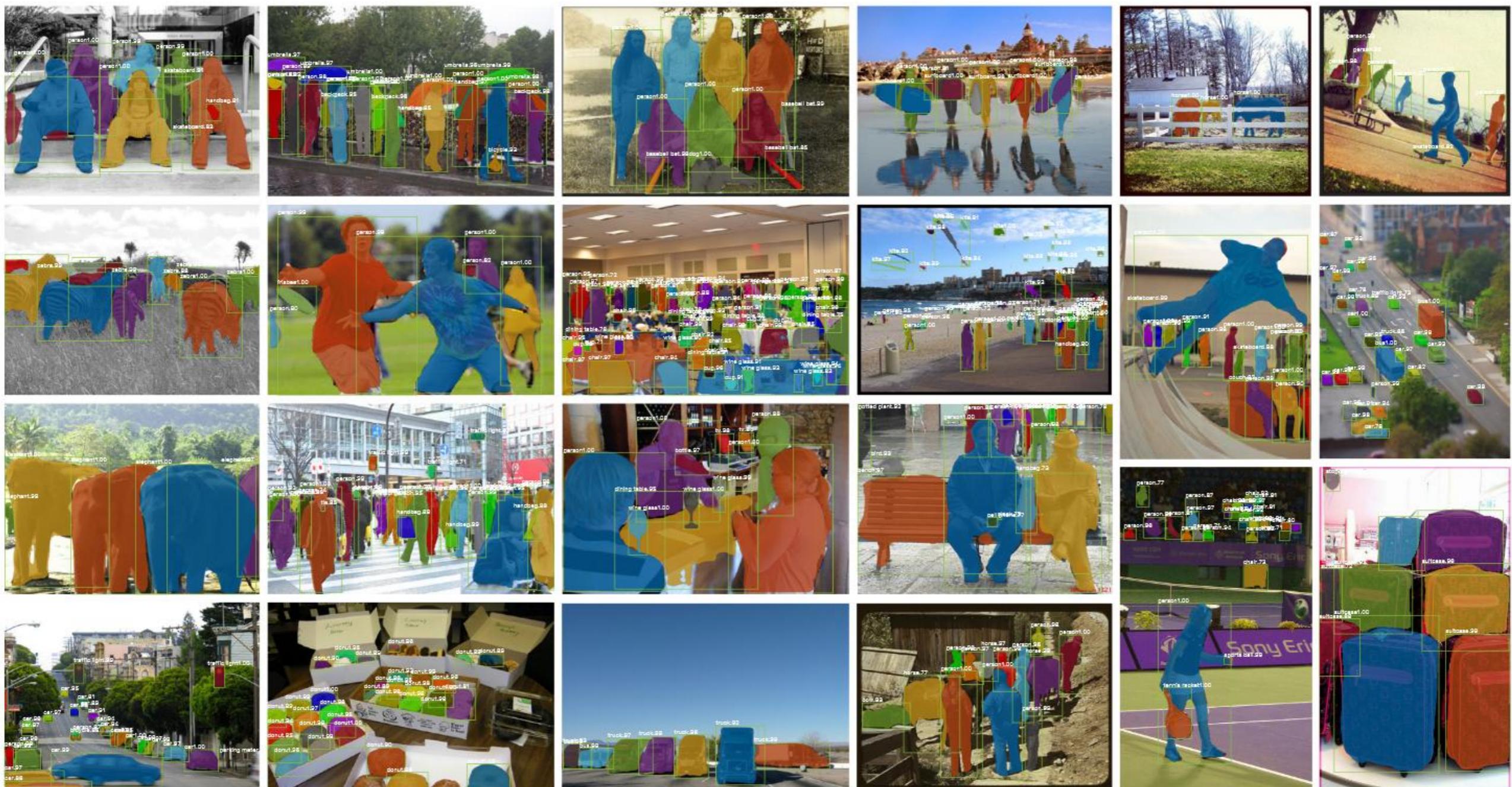
$$R_1 = (x, y_1)$$

$$R_2 = (x, y_2)$$

$$f(R_1) \approx \frac{(x_2 - x)}{(x_2 - x_1)} f(Q_{11}) + \frac{(x - x_1)}{(x_2 - x_1)} f(Q_{21})$$

$$f(R_2) \approx \frac{(x_2 - x)}{(x_2 - x_1)} f(Q_{12}) + \frac{(x - x_1)}{(x_2 - x_1)} f(Q_{22})$$

$$f(P) \approx \frac{(y_2 - y)}{(y_2 - y_1)} f(R_1) + \frac{(y - y_1)}{(y_2 - y_1)} f(R_2)$$

Figure 5. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

## Panoptic Feature Pyramid Networks

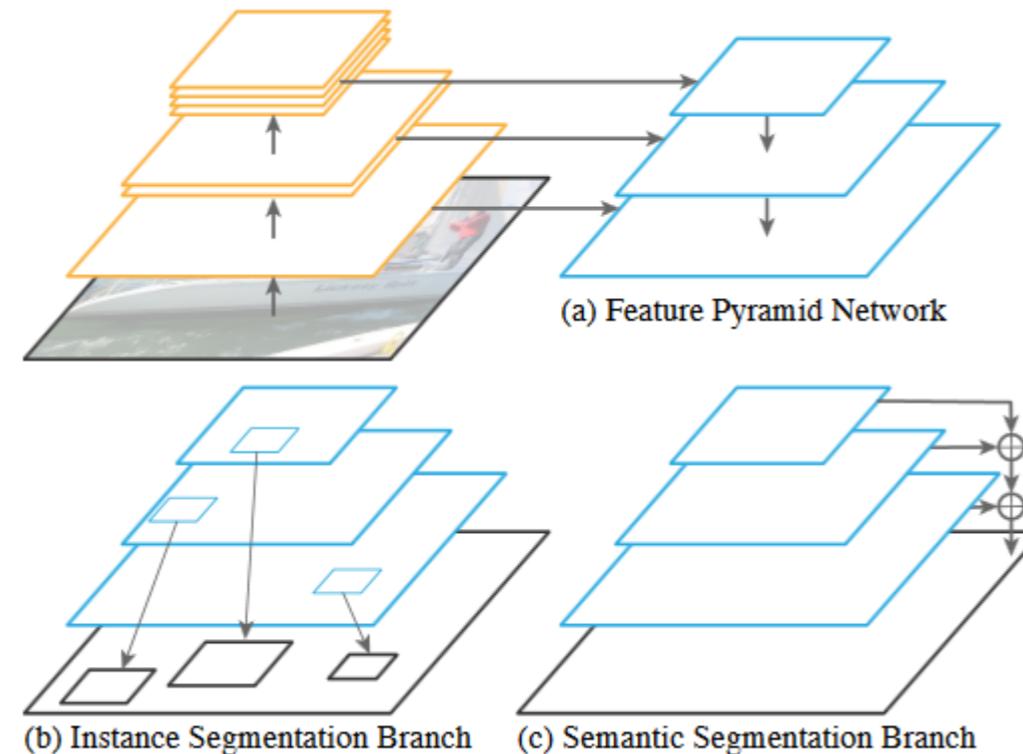


Figure 1: **Panoptic FPN**: (a) We start with an FPN backbone [36], widely used in object detection, for extracting rich multi-scale features. (b) As in Mask R-CNN [24], we use a region-based branch on top of FPN for instance segmentation. (c) In parallel, we add a lightweight dense-prediction branch on top of the same FPN features for semantic segmentation. This simple extension of Mask R-CNN with FPN is a fast and accurate baseline for both tasks.

## Instance + Semantic Segmentation

Alexander Kirillov, Ross Girshick, Kaiming He, Piotr Dollár «Panoptic Feature Pyramid Networks» // <https://arxiv.org/pdf/1901.02446.pdf>

## Panoptic Feature Pyramid Networks

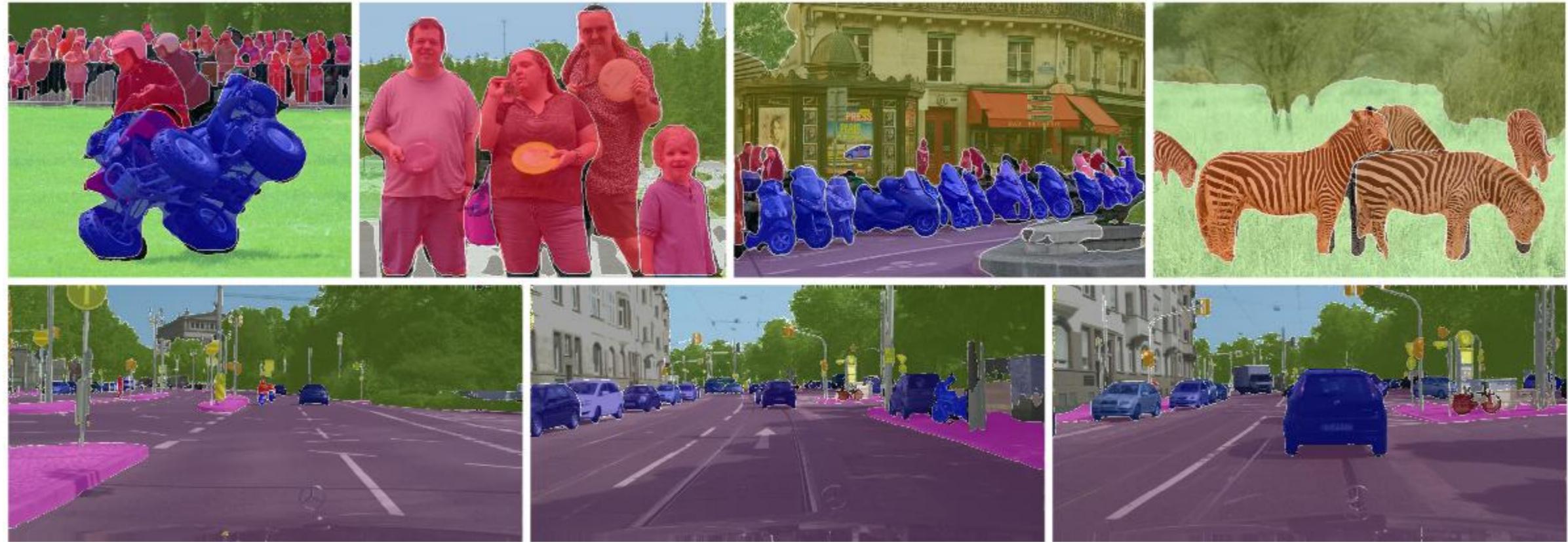
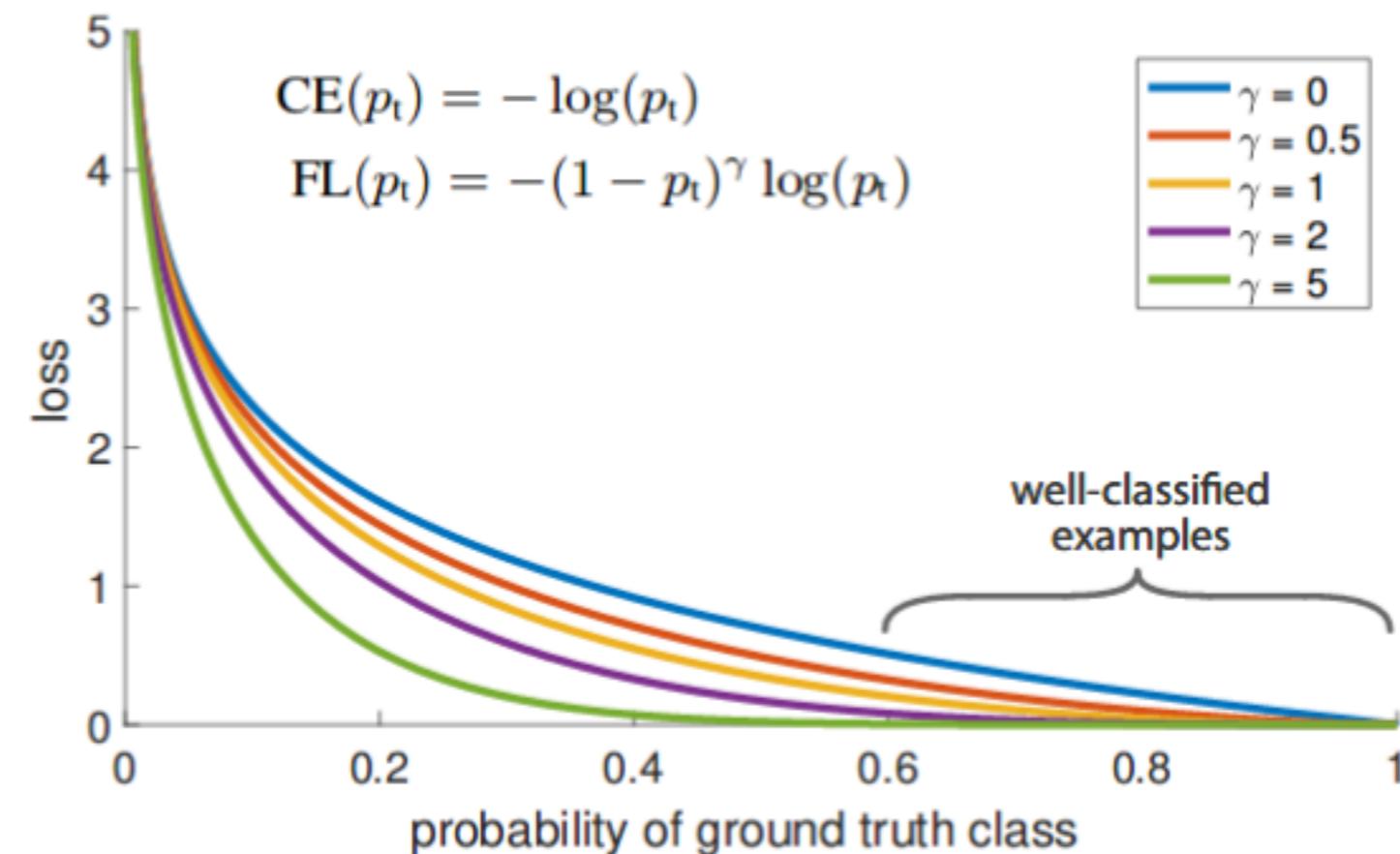


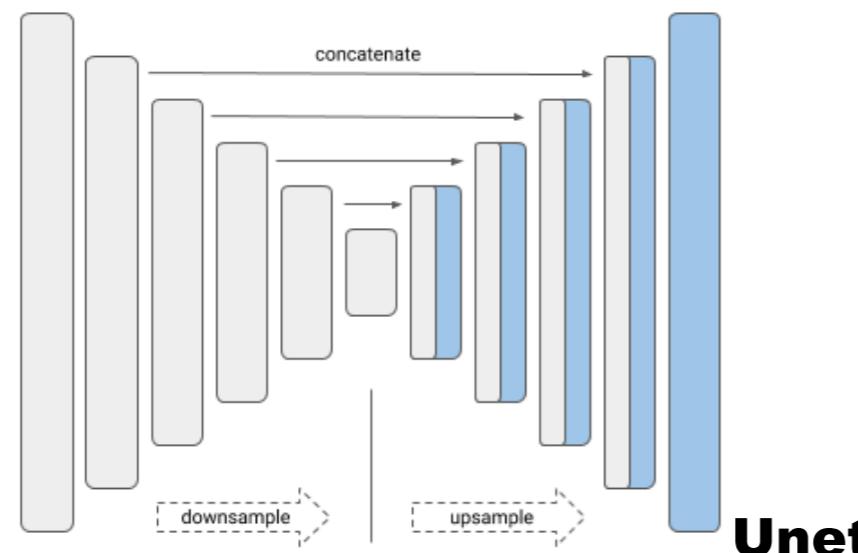
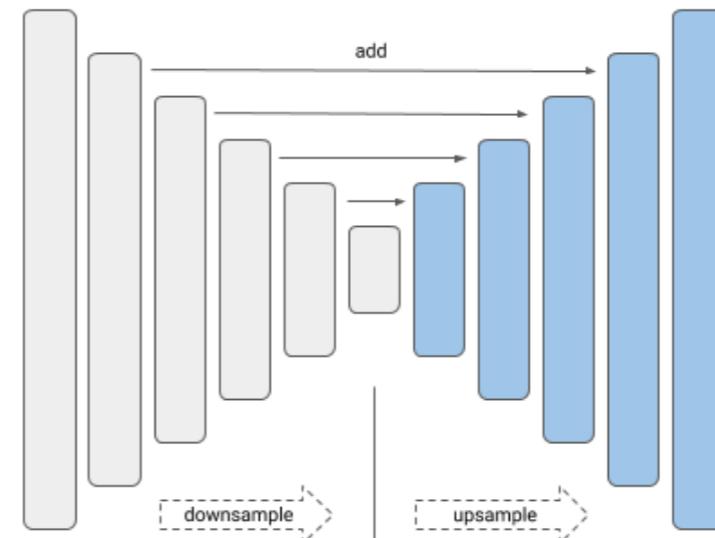
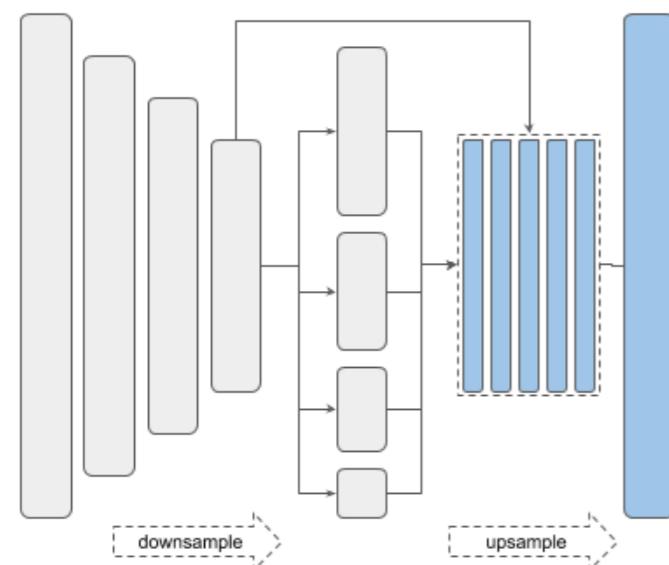
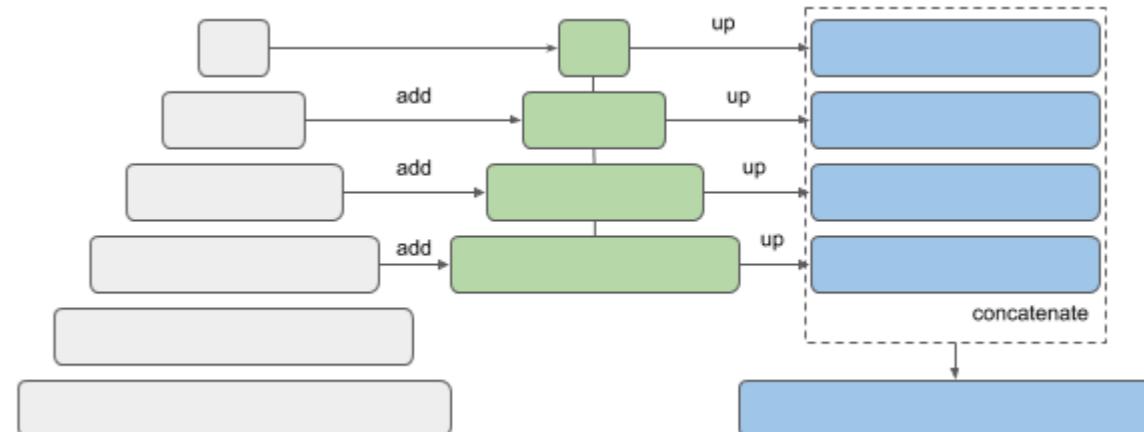
Figure 2: Panoptic FPN results on COCO (top) and Cityscapes (bottom) using a single ResNet-101-FPN network.

**RetinaNet****использование Focal Loss**

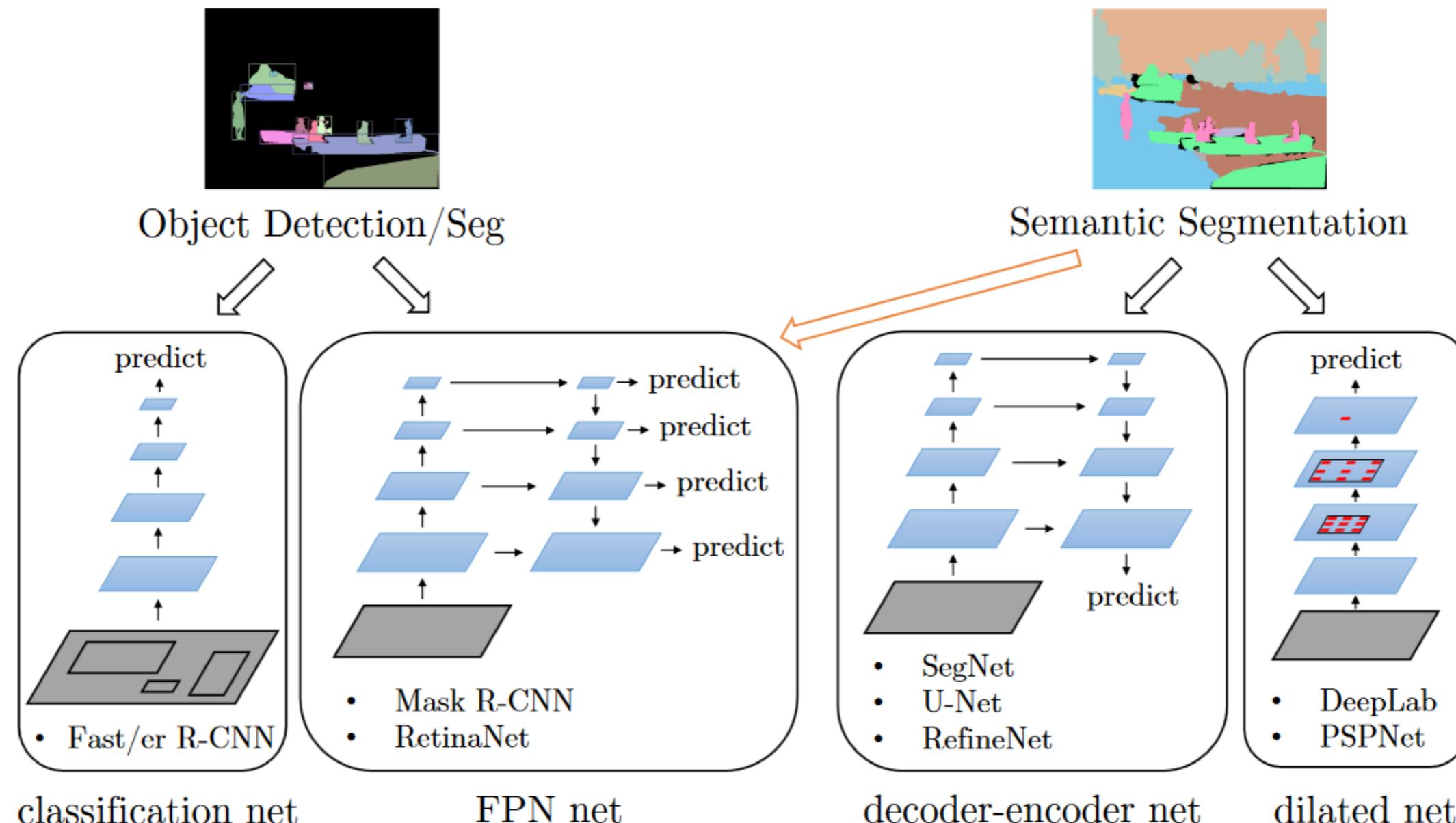
**V-Net****использование функции Дайса**

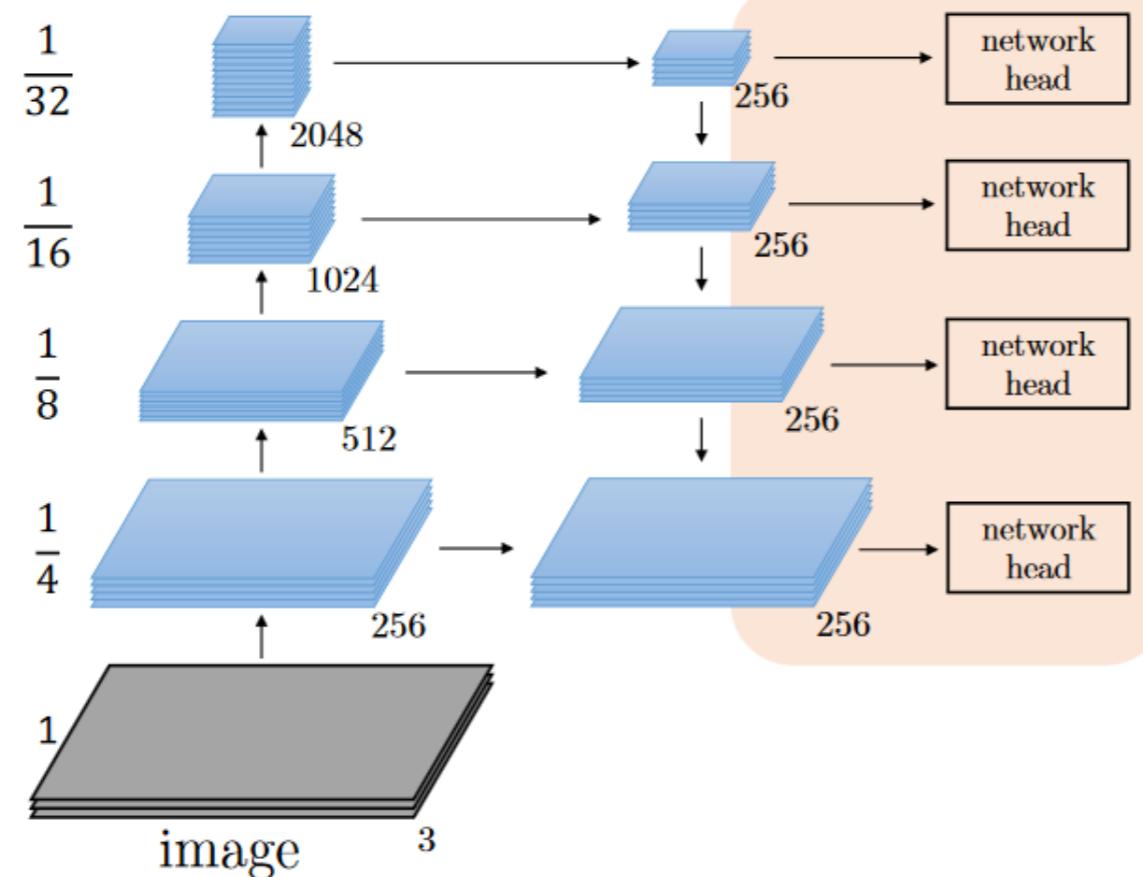
$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

## Модели для сегментации

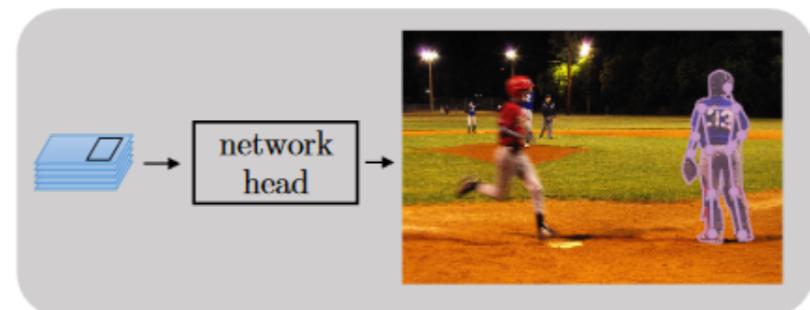
**Unet****Linknet****PSPNet****FPN**

[https://github.com/qubvel/segmentation\\_models](https://github.com/qubvel/segmentation_models)

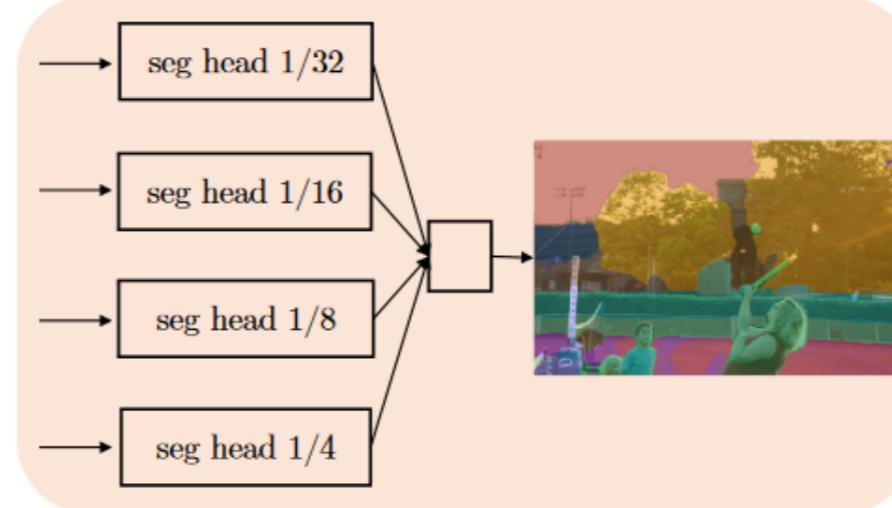
**UNet + FPN**

**UNet + FPN**Feature Pyramid  
Network (FPN) [3]

Mask R-CNN[4]

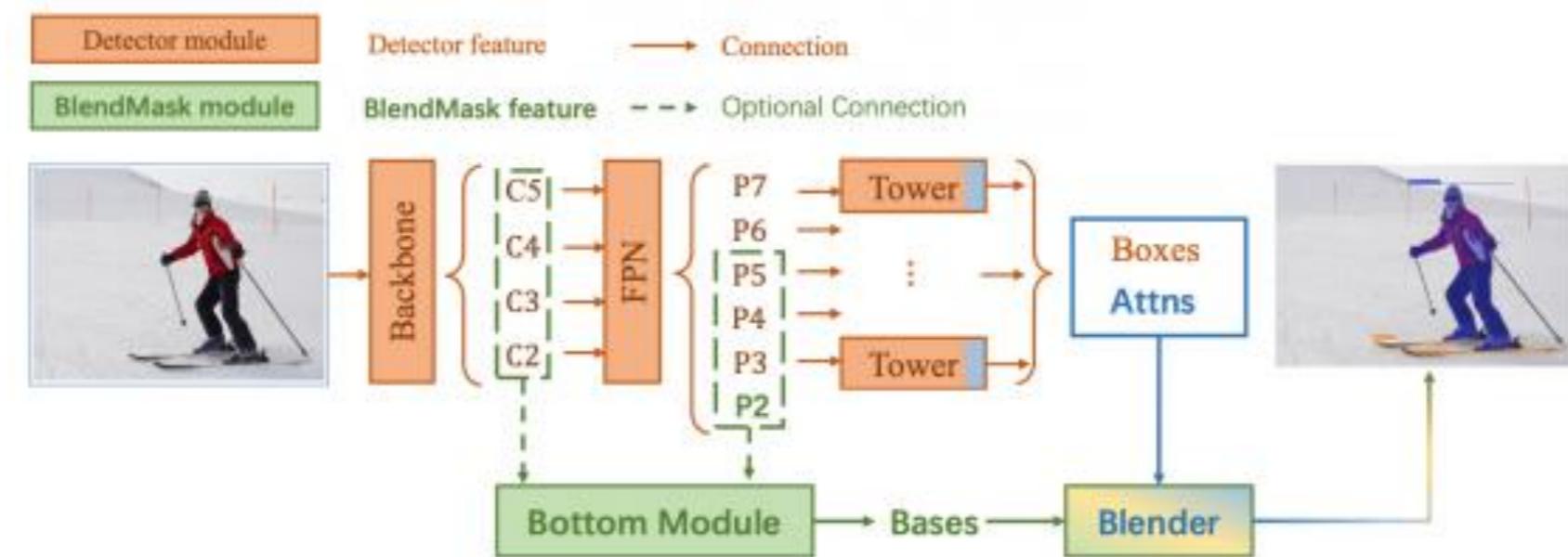


our work



<http://presentations.cocodataset.org/COCO17-Stuff-FAIR.pdf>

# BlendMask: SOTA 2020



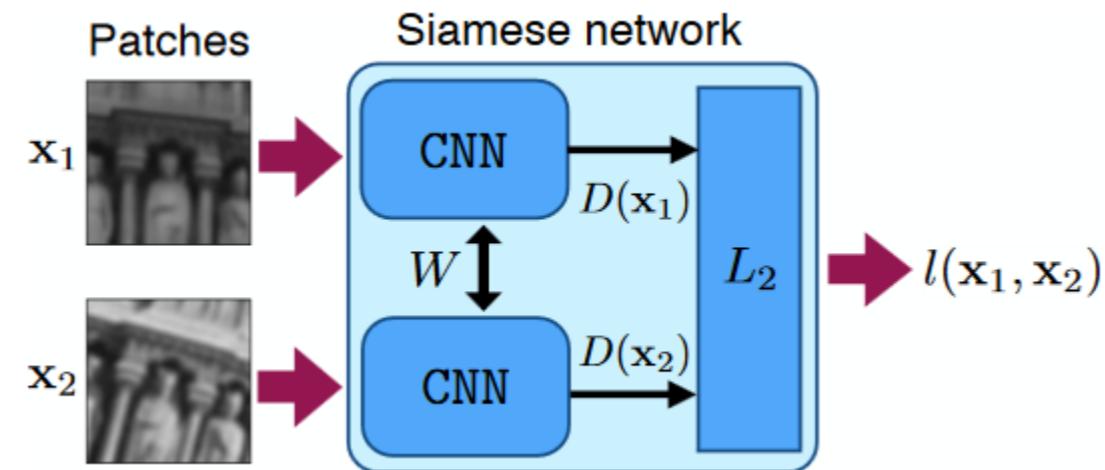
**«blender»**  
на 20% быстрее Mask R-CNN

## Поиск изображений

**Найти похожие изображения**

**Идея: изображение → вектор, поиск по векторам**

## Сиамские сети

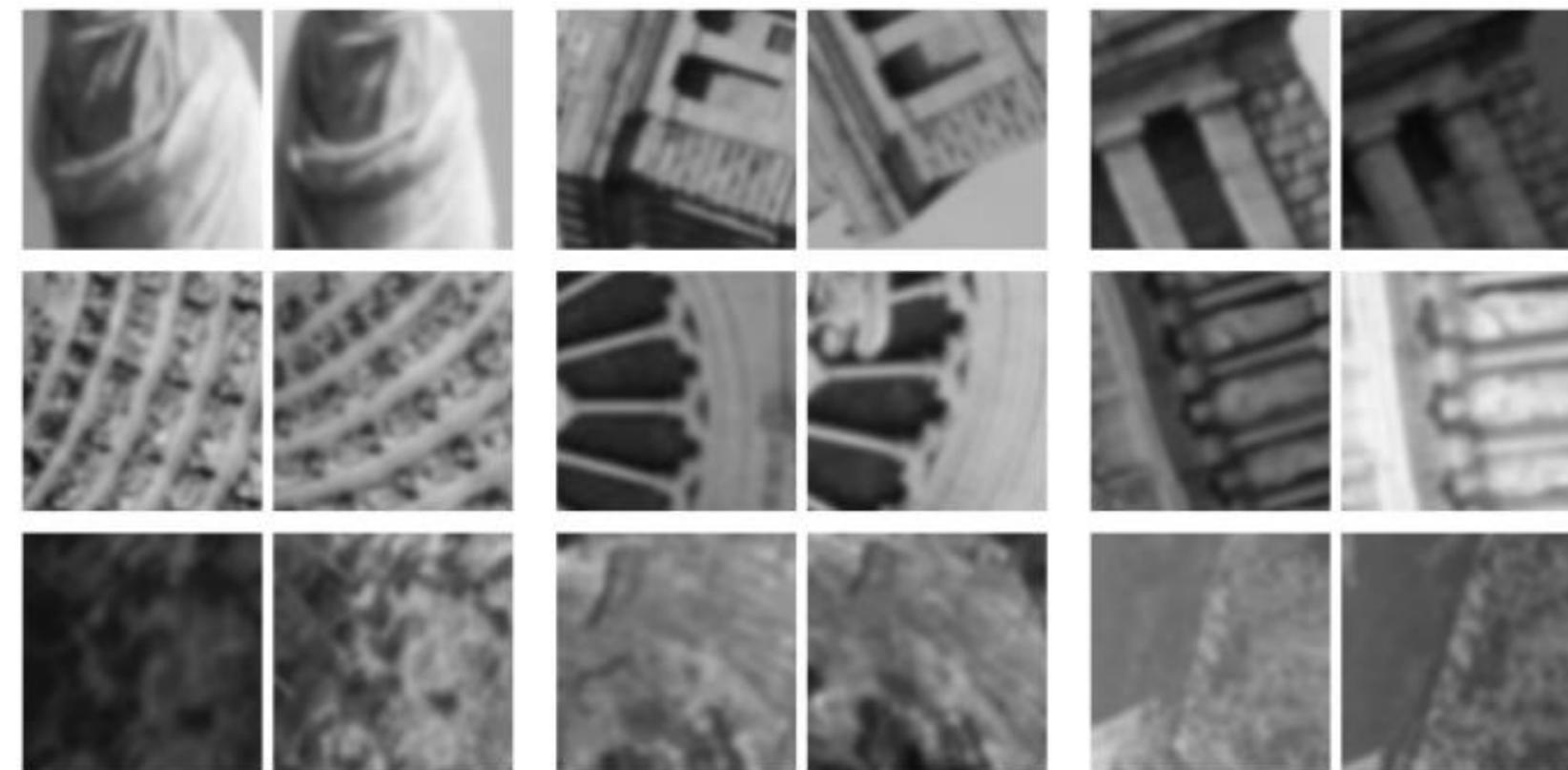


Simo-Serra et al. «Discriminative Learning of Deep Convolutional Feature Point Descriptors»,  
2015 // <http://icwww.epfl.ch/~trulls/pdf/iccv-2015-deepdesc.pdf>

**Учим descriptors: L2-расстояние было на них адекватно  
Раньше descriptors строились вручную  
Hinge Embedding Loss**

$$l(x_1, x_2) = \begin{cases} \|D(x_1) - D(x_2)\|_2, & p_1 = p_2 \\ \max(0, C - \|D(x_1) - D(x_2)\|_2), & p_1 \neq p_2 \end{cases}$$

## Сиамские сети



**Пары похожих изображений для обучения**

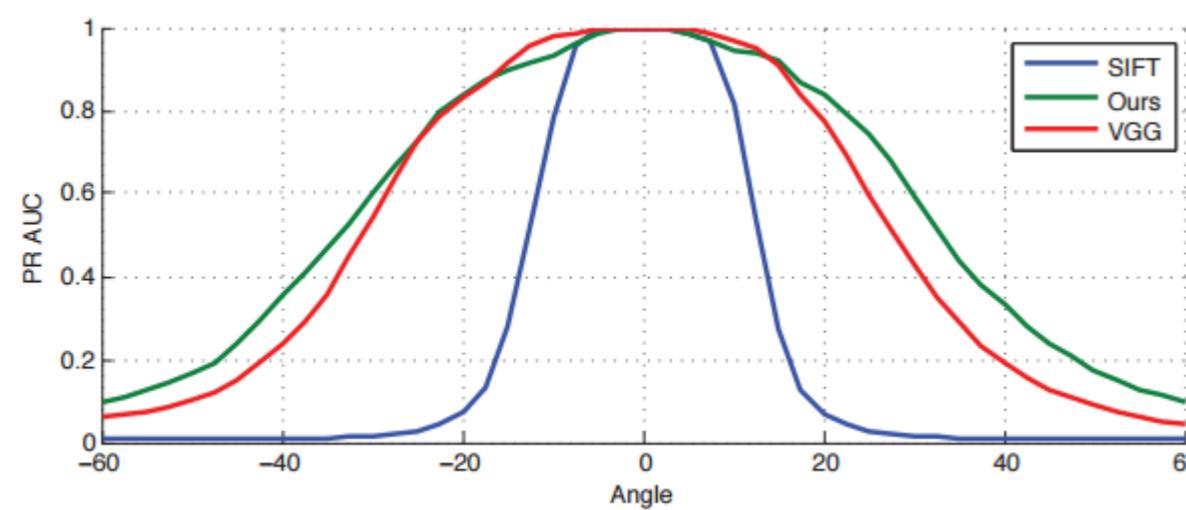
**Multi-View Stereo Dataset (MVS)**

> 1.5M чб 64×64 с разных ракурсов

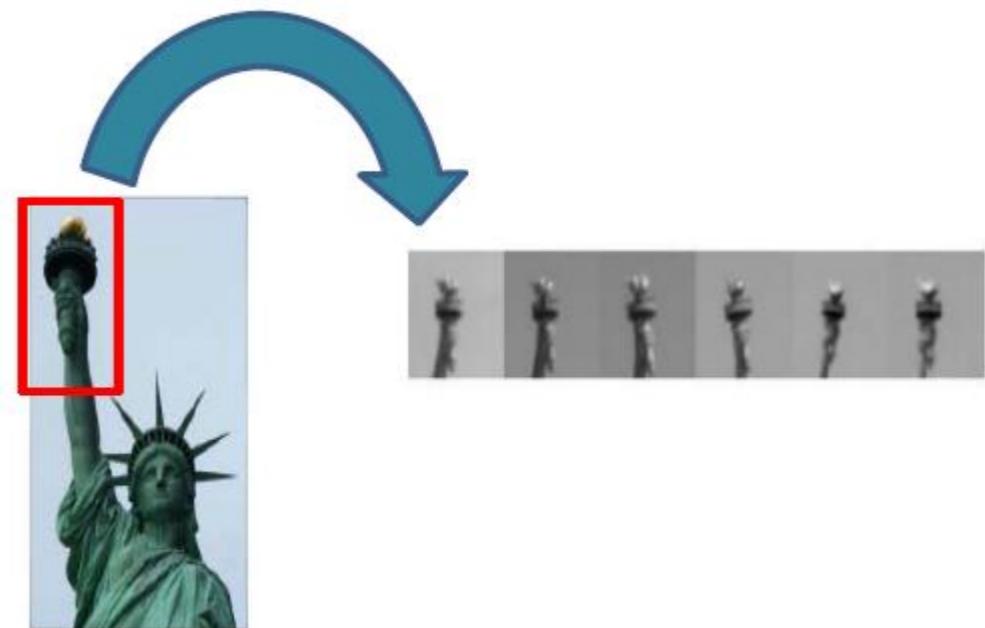
3D-реконструкций разных архитектурных сооружений

## Сиамские сети

Layer	1	2	3
Input size	$64 \times 64$	$29 \times 29$	$8 \times 8$
Filter size	$7 \times 7$	$6 \times 6$	$5 \times 5$
Output channels	32	64	128
Pooling & Norm.tion	$2 \times 2$	$3 \times 3$	$4 \times 4$
Nonlinearity	Tanh	Tanh	Tanh
Stride	2	3	4

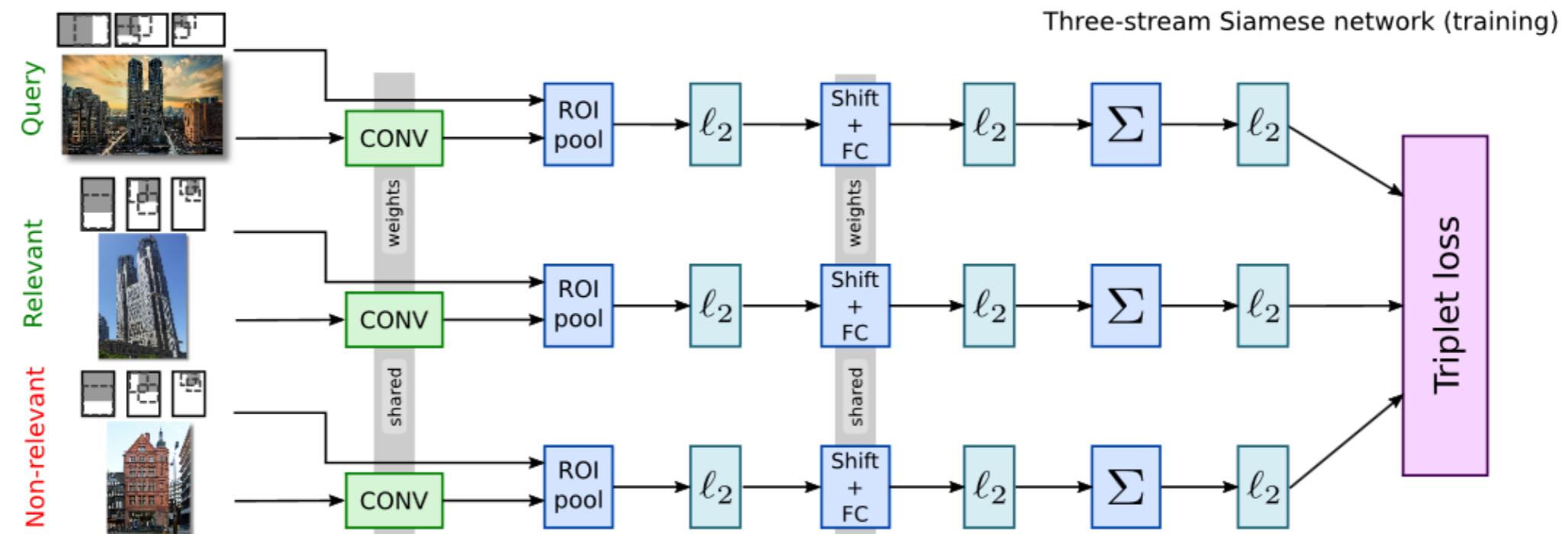


**устойчивость к повороту**



**Tanh > ReLU**  
**L2-pooling > max-pooling**  
**Нормировки!!!**  
**Subtractive normalization**  
 (вычитаем взвешенное среднее по  $5 \times 5$ -ядру)

## Triplet Loss в Сиамских сетях



**Все три сети одинаковые!!! Нас интересуют признаки на выходе**

**Свёрточные слои и натренированной VGG16 → локальные признаки, не зависящие от размеров**

**MAX-pooling в разных регионах**

**PCA (сдвиг + FC-слой)**

**L2-нормализация и сумма**

**(поэтому нет зависимости числа признаков от числа регионов)**

## Triplet Loss в Сиамских сетях

**В итоге**

**скалярное произведение пр-ов ~ many-to-many region matching**

**Все операции дифференцируемые!**

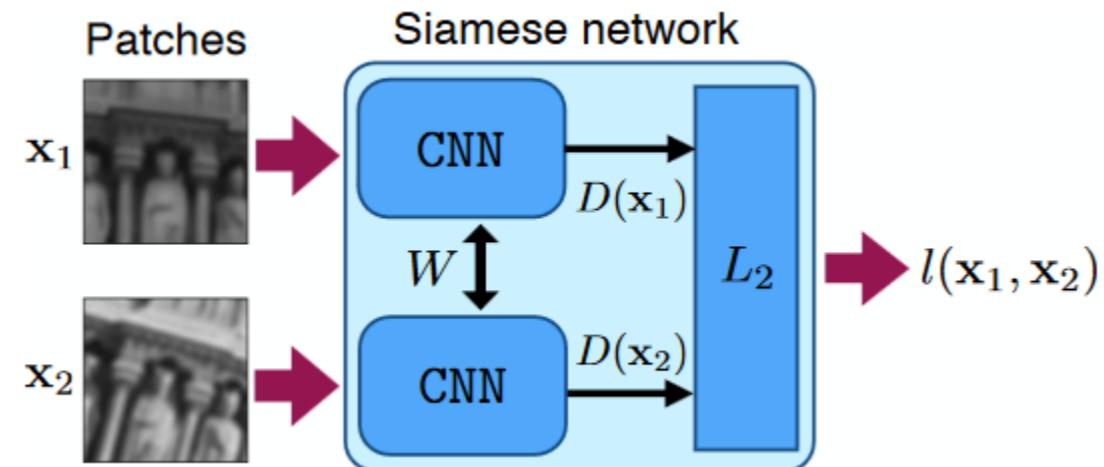
### Обучение – Triplet Loss

**запрос(q) – релевантный (+) – нерелевантный (–) ответ**

$$L(I_q, I^+, I^-) = \frac{1}{2} \max(0, m + \|q - d^+\|^2 - \|q - d^-\|^2)$$

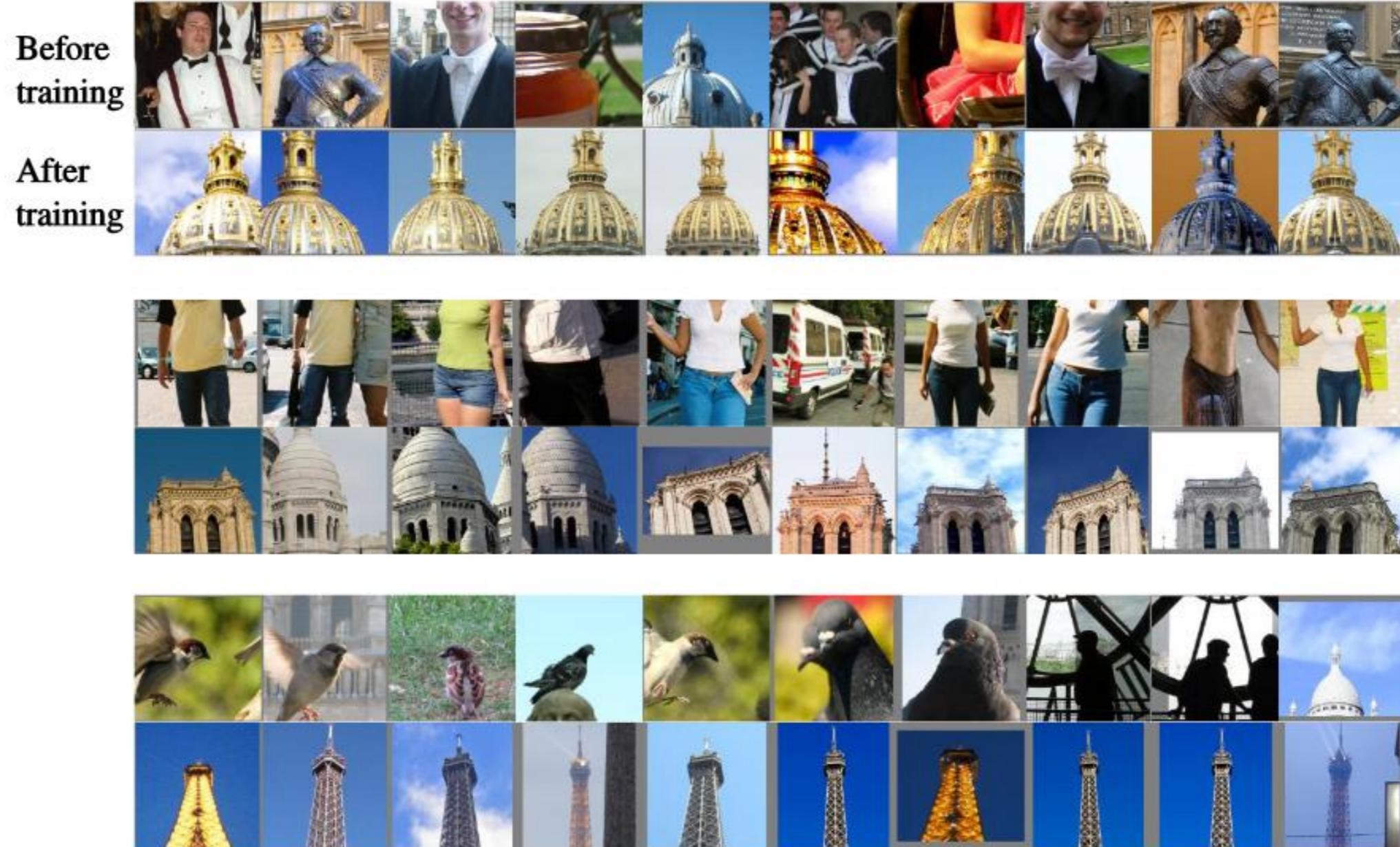
**Gordo et al. «Deep Image Retrieval», 2016 <https://arxiv.org/pdf/1604.01325.pdf>**

## Сиамские сети: обучение не факт, что в этой статье такое

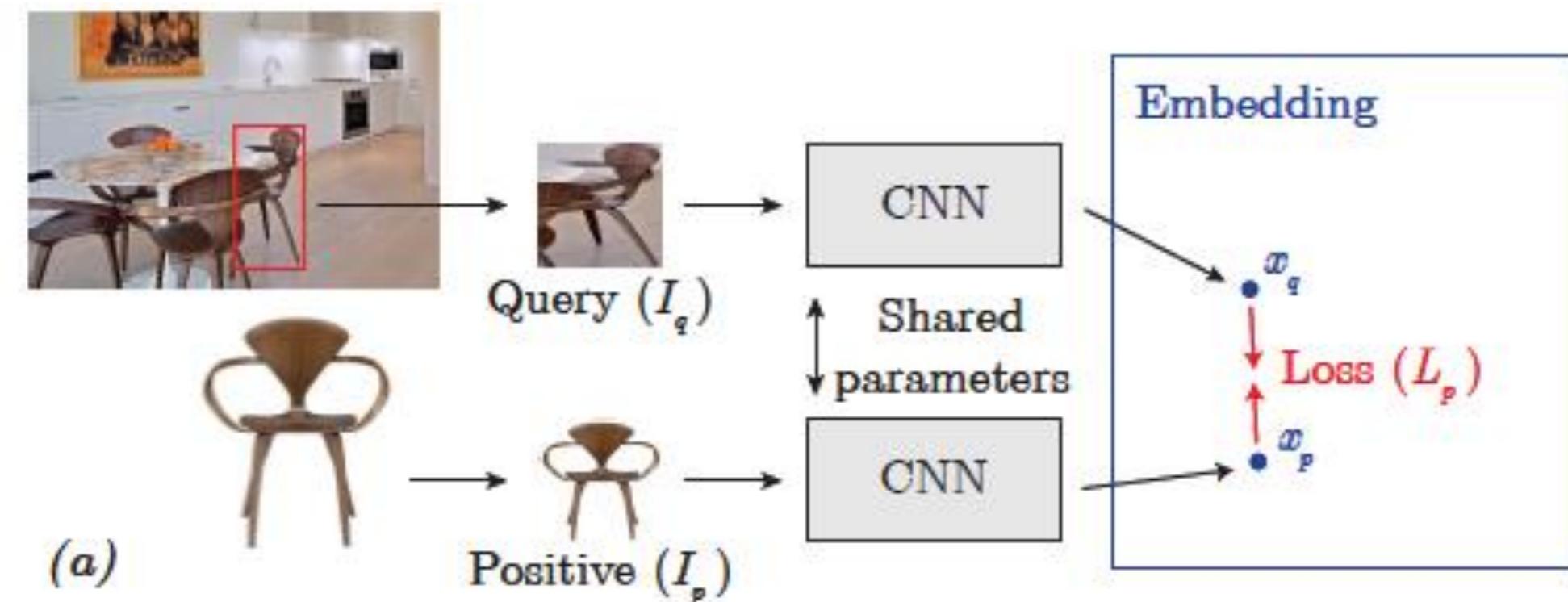


**корректировка в двух ветвях независимо  
затем – усреднение весов  
~ RNN**

## Поиск изображений



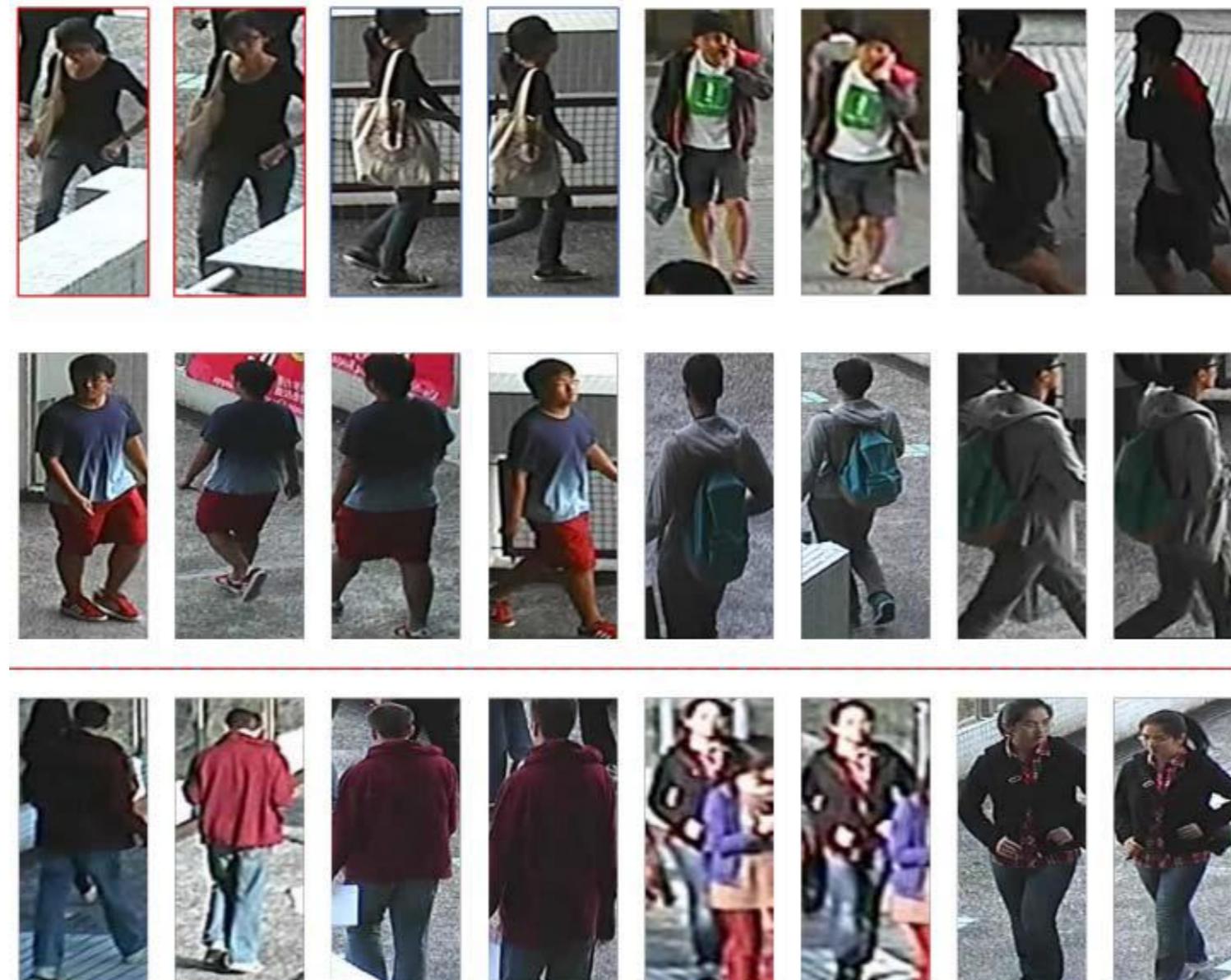
## Сиамские сети



Bell, S. and Bala, K., 2015. Learning visual similarity for product design with convolutional neural networks. ACM Transactions on Graphics (TOG), 34(4), p.98.

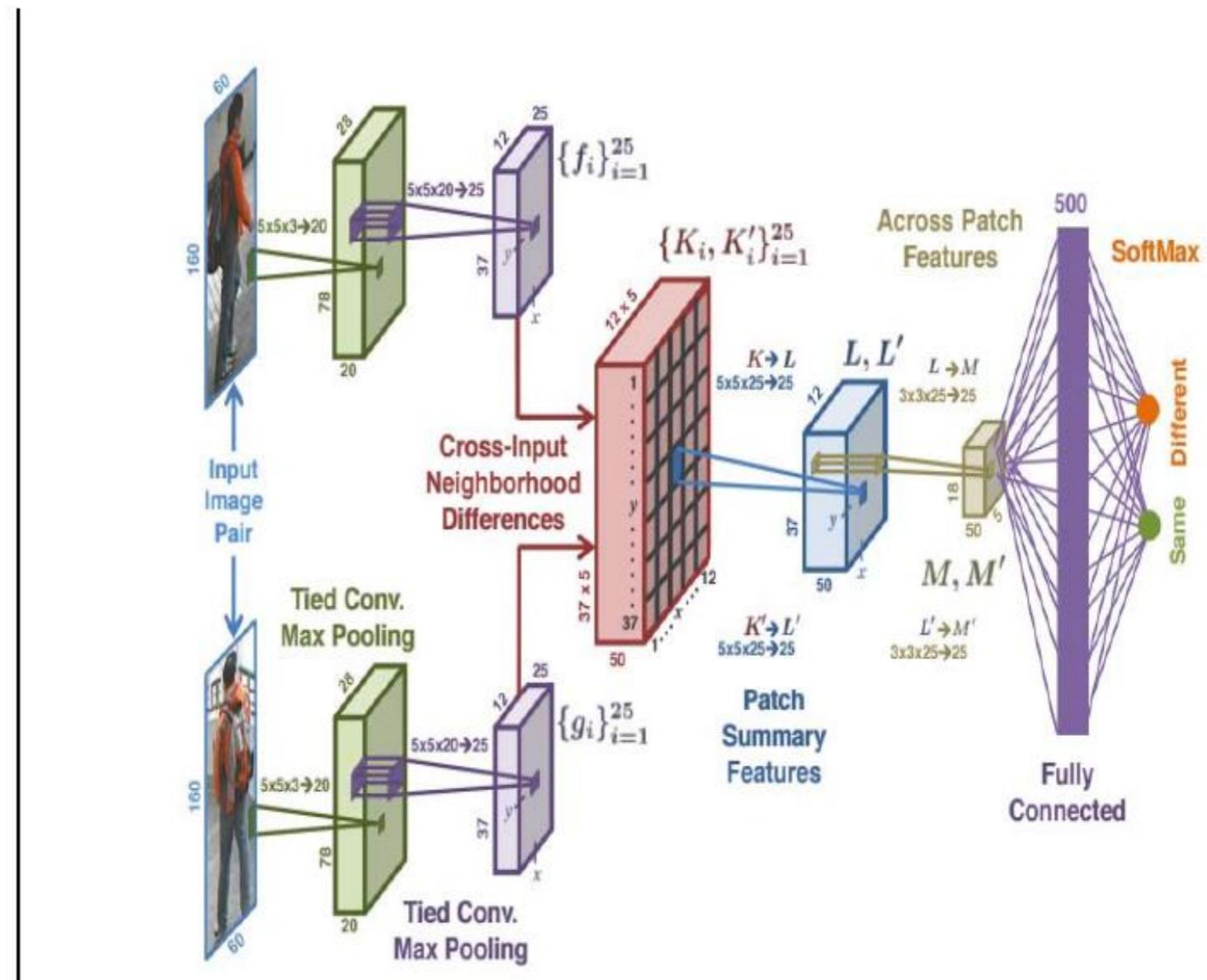
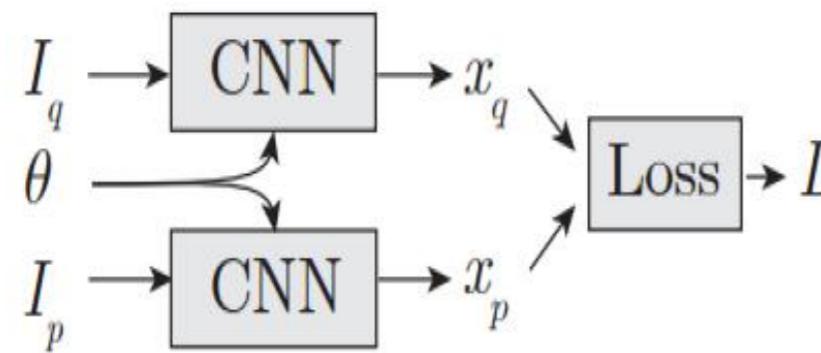
Vo, N.N. and Hays, J., 2016, October. Localizing and orienting street views using overhead imagery. In European Conference on Computer Vision (pp. 494-509)

## Сиамские сети: Person Re-Identification



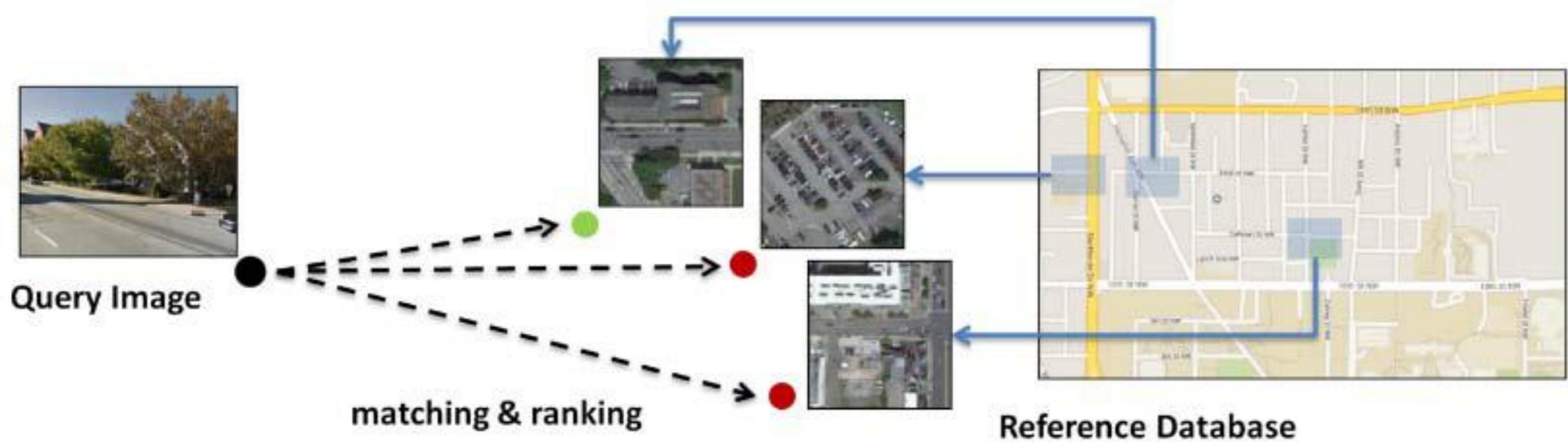
Ahmed, E., Jones, M. and Marks, T.K., 2015. An improved deep learning architecture for person re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(pp. 3908-3916).

## Сиамские сети: Person Re-Identification



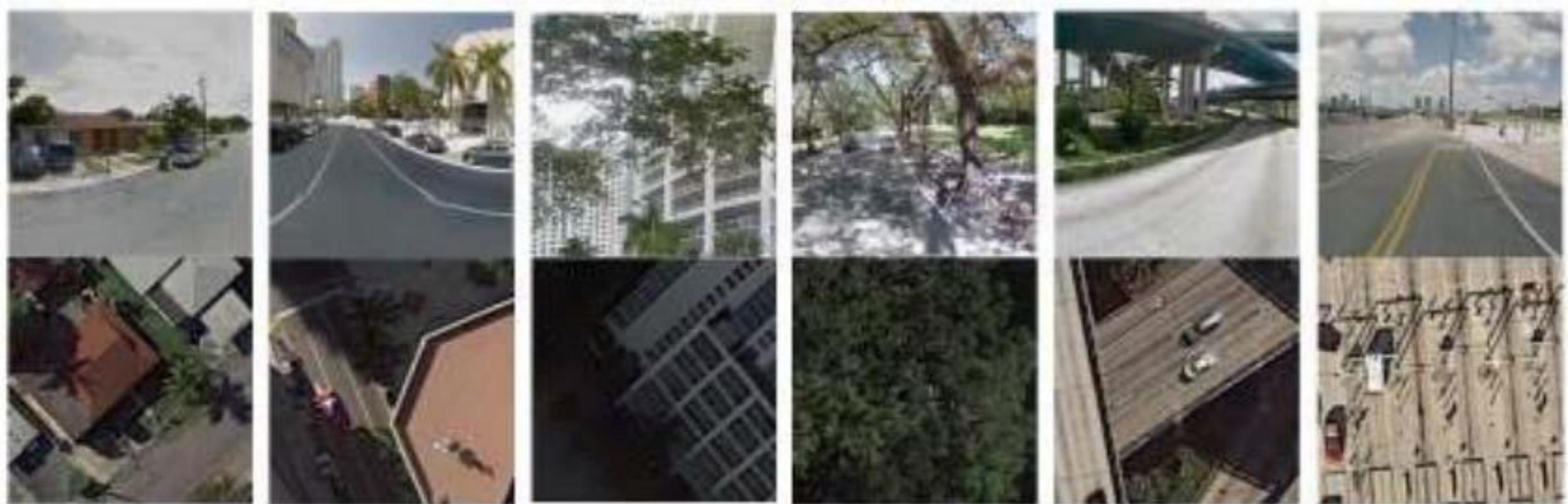
**тут есть тонкости**

## Сиамские сети: Street-View to Overhead-View Image Matching



Vo, N.N. and Hays, J., 2016, October. Localizing and orienting street views using overhead imagery. In European Conference on Computer Vision (pp. 494-509)

## Сиамские сети: Street-View to Overhead-View Image Matching



запросы (вверху), релевантное изображение (внизу)

## Сиамские сети: Street-View to Overhead-View Image Matching

Classification CNN:



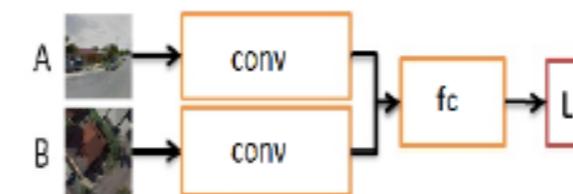
$$L(A, B, l) = \text{LogLossSoftMax}(f(I), l)$$

$I = \text{concatenation}(A, B)$

$f = \text{AlexNet}$

$l = \{0, 1\}$ , label

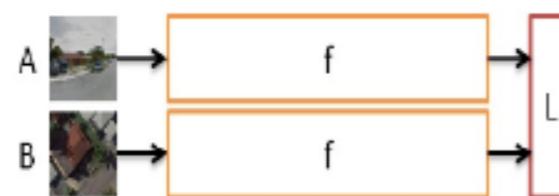
Siamese-classification hybrid network:



$$L(A, B, l) = \text{LogLossSoftMax}(f_{fc}(I_{conv}), l)$$

$$I_{conv} = \text{concatenation}(f_{conv}(A), f_{conv}(B))$$

Siamese-like CNN:

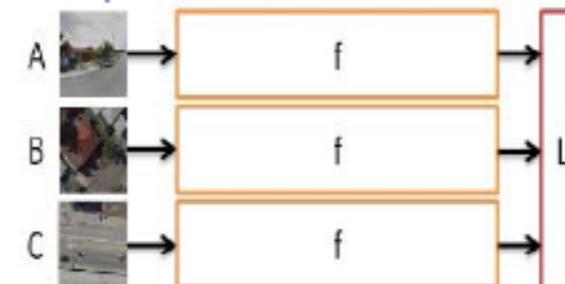


$$L(A, B, l) = l * D + (1 - l) * \max(0, m - D)$$

$$D = \|f(A) - f(B)\|_2$$

$m = \text{margin parameter}$

Triplet network CNN:



$$L(A, B, C) = \max(0, m + D(A, B) - D(A, C))$$

$(A, B)$  is a match pair

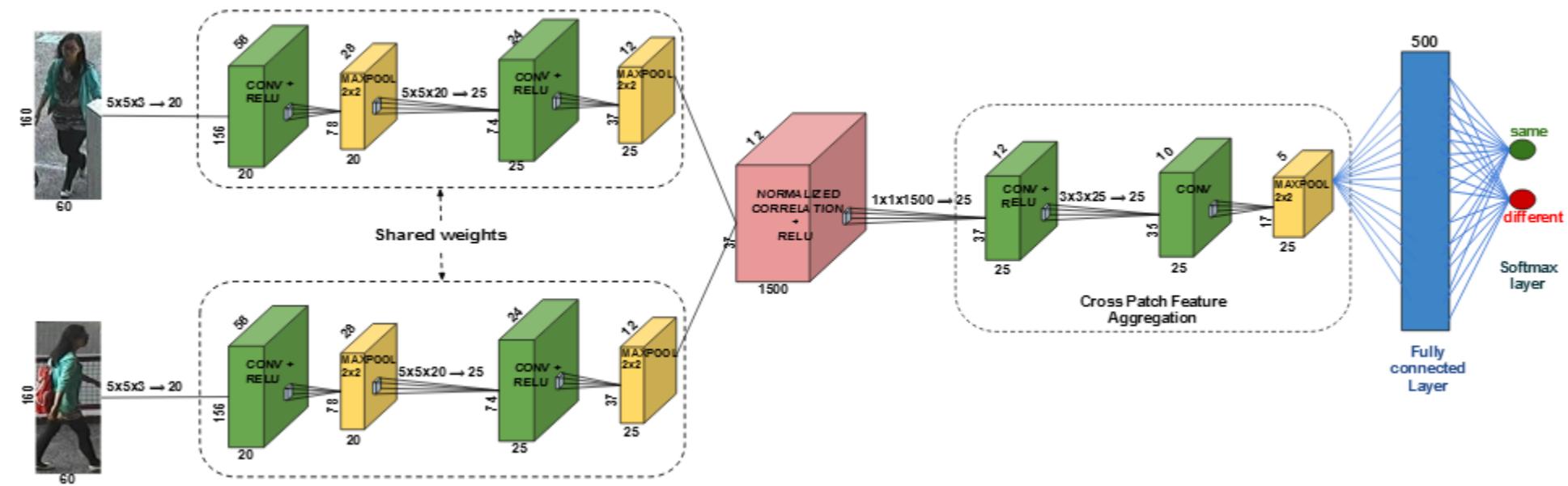
$(A, C)$  is a non-match pair

**тут термин «Triplet network»**

**Distance-based logistic (DBL) Loss (см)**

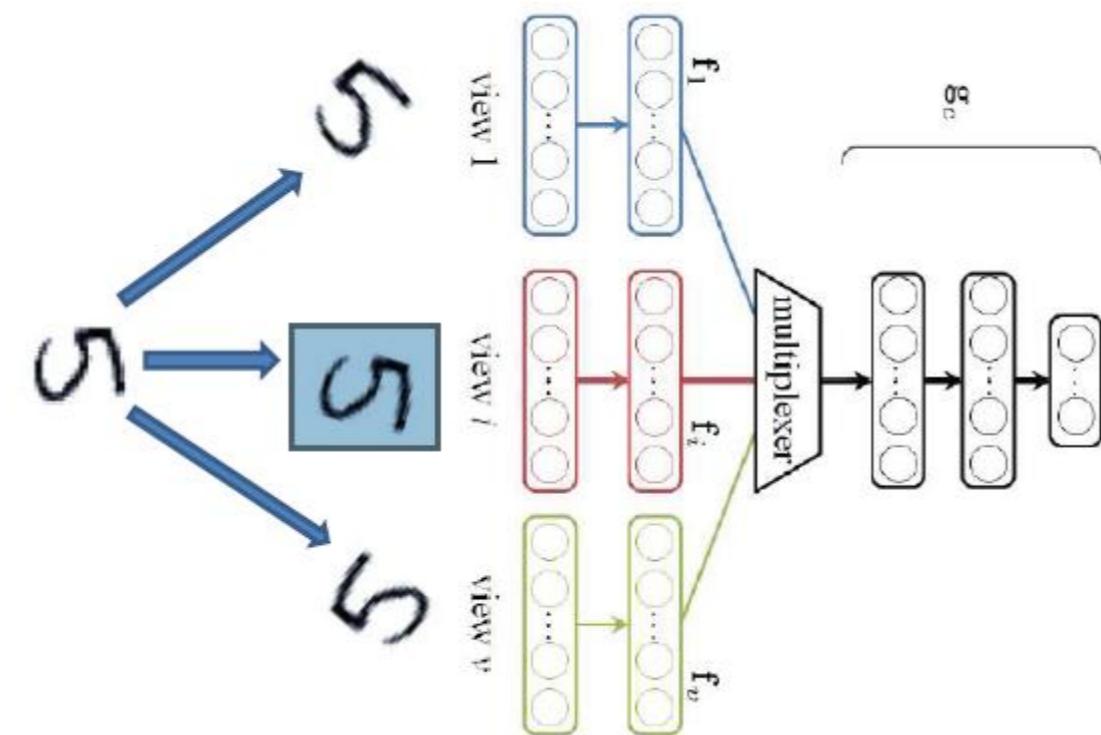
**гибридная сеть лучше всего**

## Сиамские сети: Intermediate merging



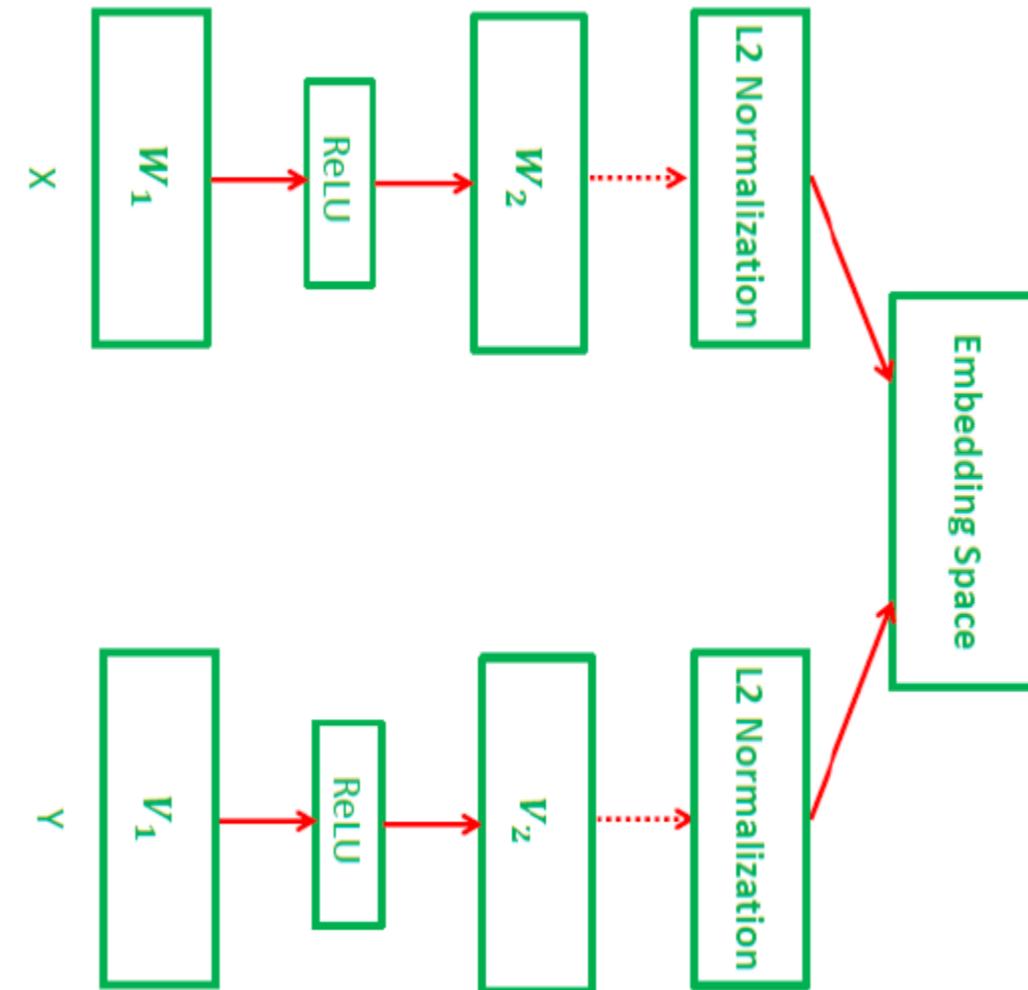
**Subramaniam, A., Chatterjee, M. and Mittal, A., 2016. Deep Neural Networks with Inexact Matching for Person Re-Identification. In Advances in Neural Information Processing Systems(pp. 2667-2675).**

## Сиамские сети: for viewpoint invariance



Kan, M., Shan, S. and Chen, X., 2016. Multi-view deep network for cross-view classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(pp. 4847-4855).

## Сиамские сети: for cross-modal embedding



Wang, L., Li, Y. and Lazebnik, S., 2016. Learning deep structure-preserving image-text embeddings. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(pp. 5005-5013).

Hu, Baotian, et al., Convolutional neural network architectures for matching natural language sentences, NIPS 2014

# FaceNet

**пространство изображений в вещественное пространство**

**определение человека – kNN**

<https://github.com/Skuldur/facenet-face-recognition>

<https://arxiv.org/pdf/1503.03832.pdf>

## Итог

**сегментация: опять свёрточные решения  
сегментация и детекция – есть общие принципы**

**Идея полностью свёрточных решений**

**Идея проектирования связей**

**Сиамские сети – метрики в DL**

## Курсы

**Convolutional Neural Networks for Visual Recognition**  
<http://cs231n.stanford.edu/>