Сегментация и плотное предсказание

План лекции

- Плотное предсказание
- Сегментация
- Другие виды плотного предсказания

Плотное предсказание

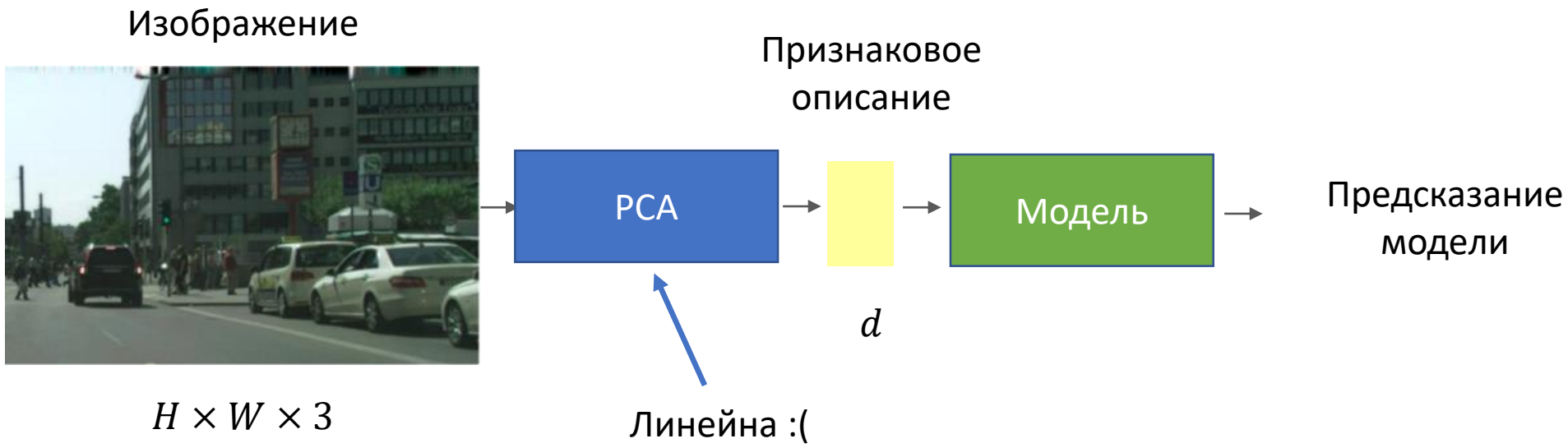
- Классификация
  - Картинка –> число / вектор.
  - Отвечает на вопрос, что нарисовано на картинке.
  - Оцениваем картинку целиком.

- Плотное предсказание
  - Картинка –> число / вектор для каждого пикселя
  - Отвечает на вопрос, что и где расположено на картиинке.
  - Оцениваем каждый пиксель изображения

Скрытые признаки картинки

Изображение содержит $H \times W \times 3$ признаков, что избыточно.
Многие модели работают по принципу извлечения некоторых скрытых признаков из картинки.
Например, количество людей, автомобилей, окон (они не обязаны быть интерпретируемы).

Что мы знаем из донейросетевой эпохи?
PCA – линейная проекция на подпространство, образованное компонентами с большей дисперсией.
Далее на этих признаках можно обучить другую модель.



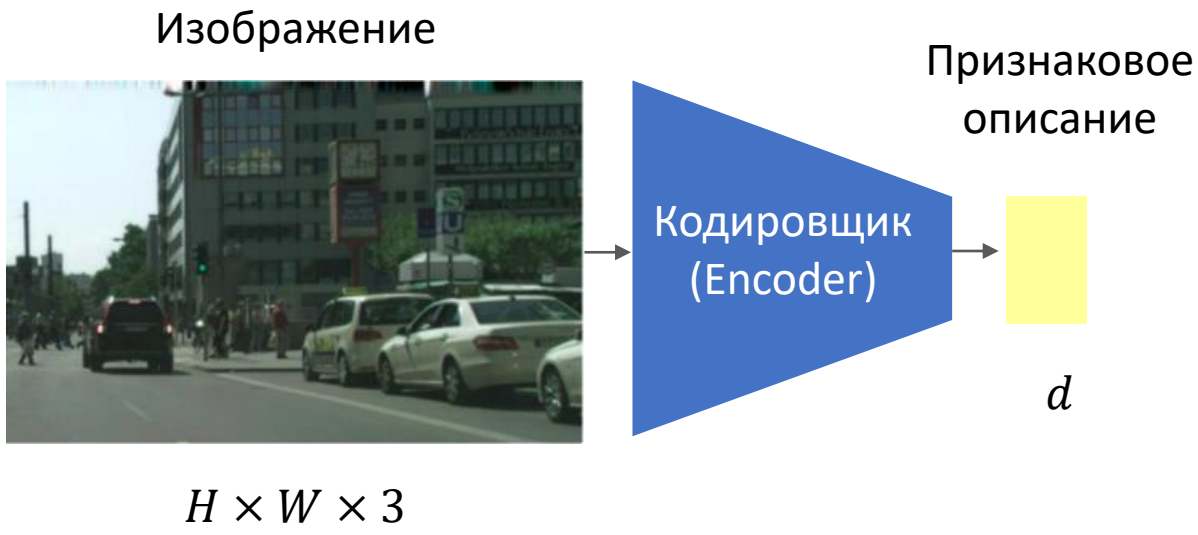
Скрытые признаки картинки

Идея: заменим в PCA линейное преобразование на нейросеть.

Проблемы:

- Как ее обучить?
- Как из скрытого пространства восстановить картинку?

В PCA для этого была линейная функция.



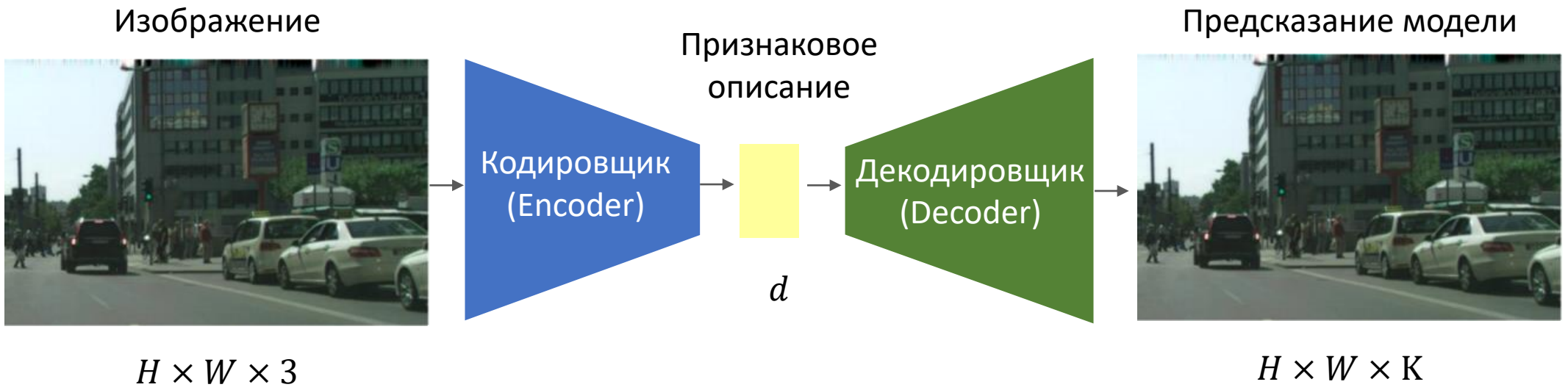
Автоэнкодер

Идея: заменим в PCA линейное преобразование на нейросеть.

Проблемы:

- Как ее обучить?
- Как из скрытого пространства восстановить картинку?
  В PCA для этого была линейная функция.

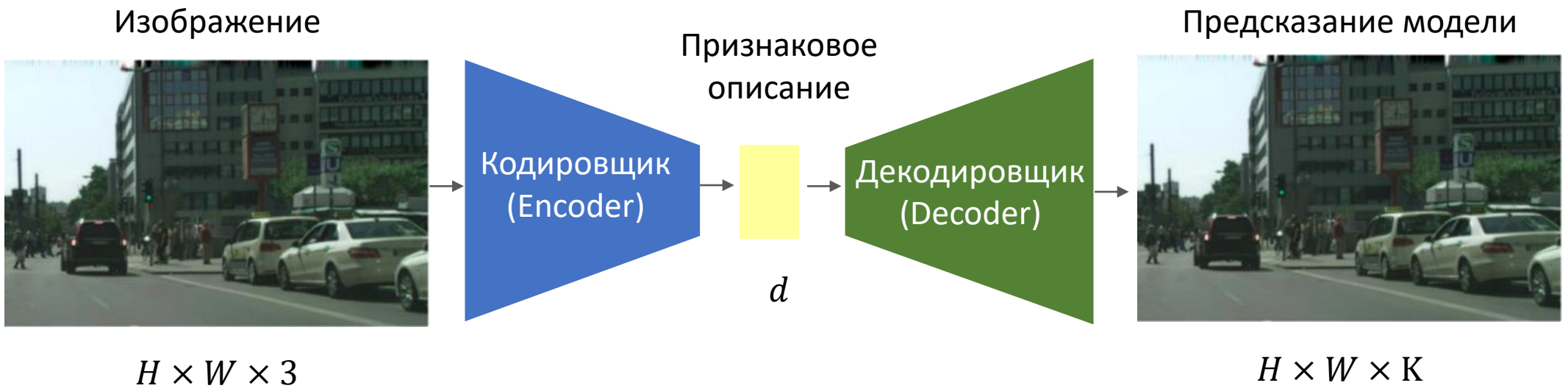
Идея: функцию восстановления тоже построим нейросетью и будем оптимизировать MSE исходной картинки и восстановленной.



Автоэнкодер

- **Кодировщик** (Encoder) преобразует изображение в его признаковое описание, тензор размера $h \times w \times d$.
- **Декодировщик** (Decoder) преобразует тензор признаков в приближение исходной картинки.
- **Оптимизируем MSE** между исходной картинкой и восстановленной.

Данная идея является основной многих нейросетевых моделей.



План лекции

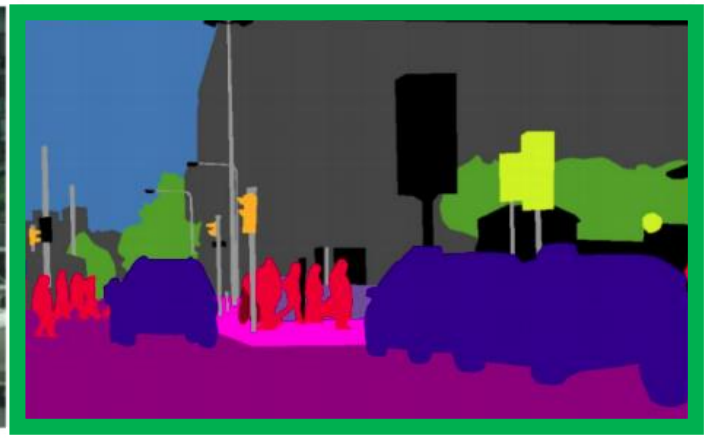
- Плотное предсказание
- Сегментация
  - Виды сегментации
  - Семантическая сегментация
  - Upsampling
  - Модели сегментации
  - Метрики и функции ошибки
- Другие виды плотного предсказания

Сегментация

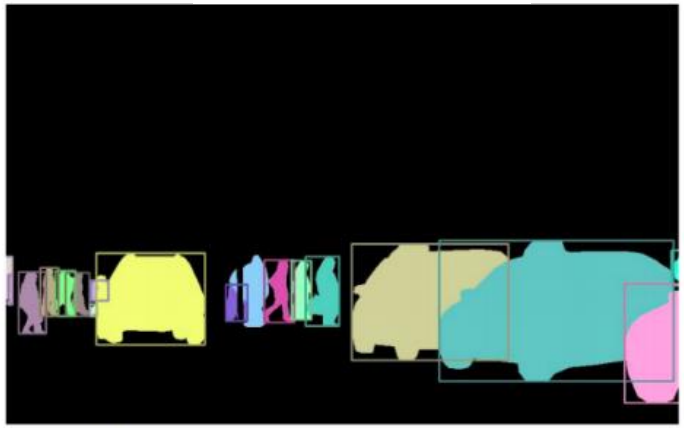
Семантическая сегментация (semantic segmentation)

- Находим класс для каждого пикселя изображения.
- Нет разделения между объектами: если объекты находятся рядом, то они сливаются в одну область.

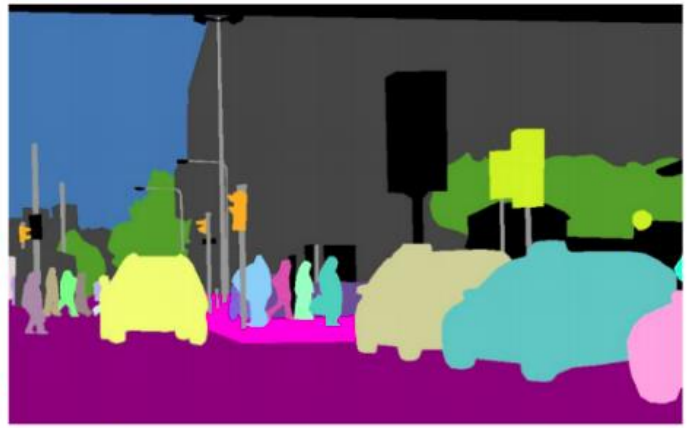
Изображение



Семантическая (semantic) сегментация



Объектная (instance) сегментация



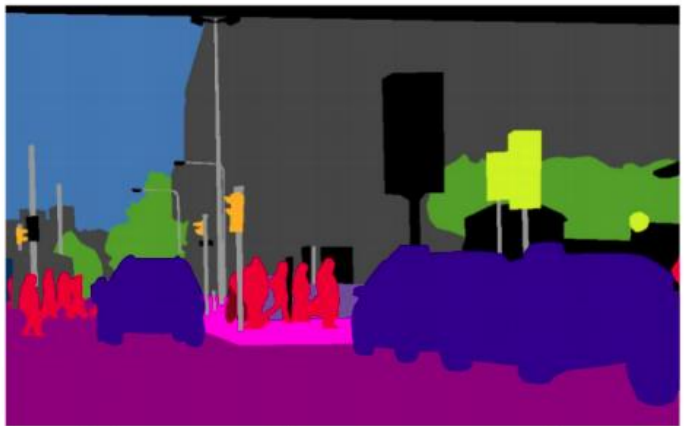
Паноптическая (panoptic) сегментация

Сегментация

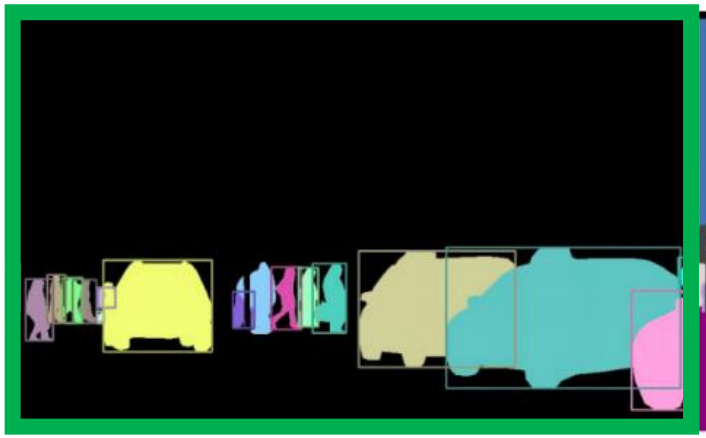
Объектная сегментация
(instance segmentation)

- Находим определенные объекты.
- Для каждого объекта находим все пиксели, которые ему принадлежат.

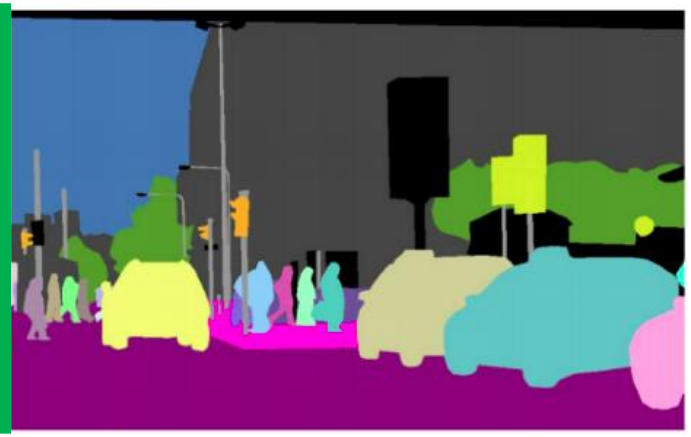
Изображение



Семантическая (semantic) сегментация



Объектная (instance) сегментация



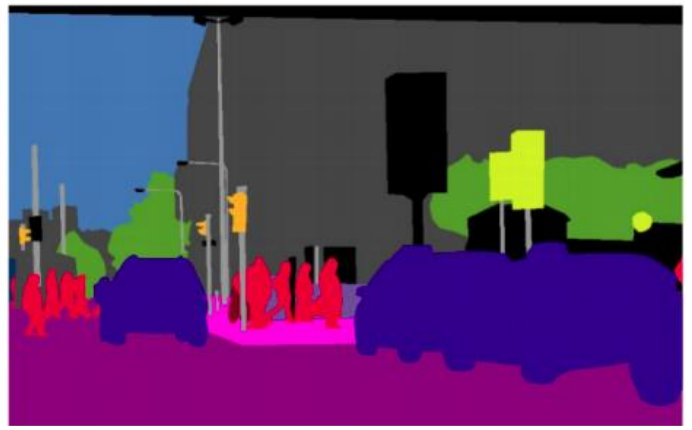
Паноптическая (panoptic) сегментация

Сегментация

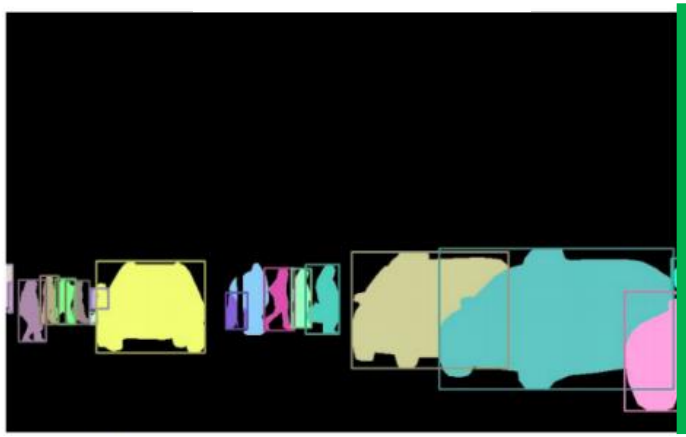
Паноптическая сегментация (panoptic segmentation)

- Находим класс для каждого пикселя изображения.
- Разделяем объекты одного класса.

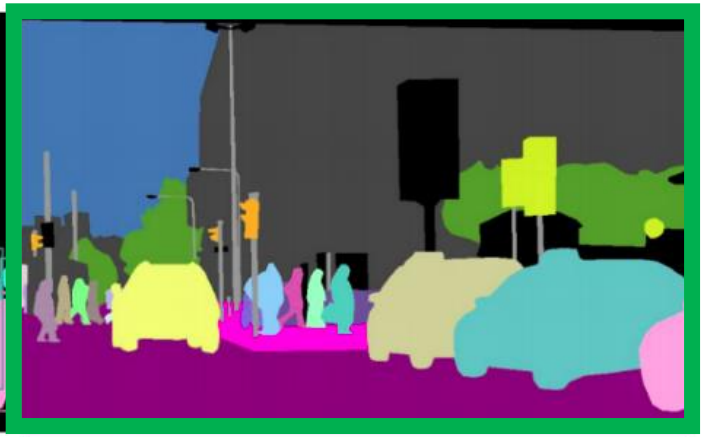
Изображение



Семантическая (semantic) сегментация



Объектная (instance) сегментация



Паноптическая (panoptic) сегментация

Задача семантической сегментации

Пусть

$\mathscr{X}$ — пространство картинок.

$\mathscr{Y}$ — набор классов, например
{человек, машина, дорога, тротуар, знак ПДД}.

$\mathscr{Y}_k$ — пространство картинок,
где каждый пиксель имеет значение из Y.

Требуется построить модель $f: \mathscr{X} \to \mathscr{Y}_k$,
определяющую, к какому классу из $\mathscr{Y}$
принадлежит каждый пиксель изображения X.

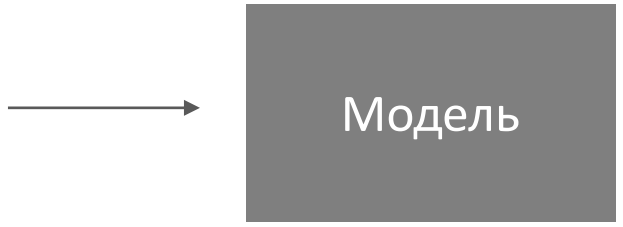
Модель семантической сегментации

Пусть H – высота изображения, W – ширина изображения, K – количество классов.
Будем строить модель семантической сегментации.

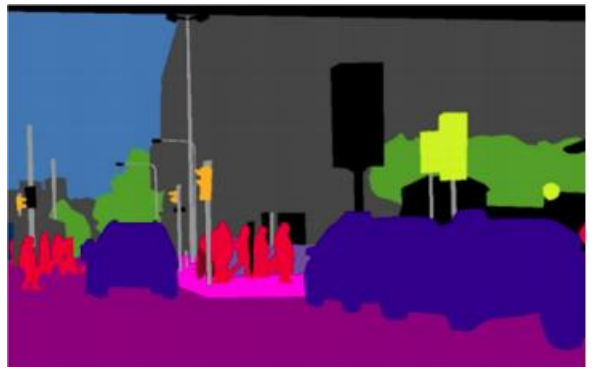
- Вход модели – изображение, тензор размера $H \times W \times 3$.
- Выход модели – предсказание для каждого пикселя, к какому классу он относится, тензор размера $H \times W \times \mathrm{K}$.

Изображение

$H \times W \times 3$



Предсказание модели

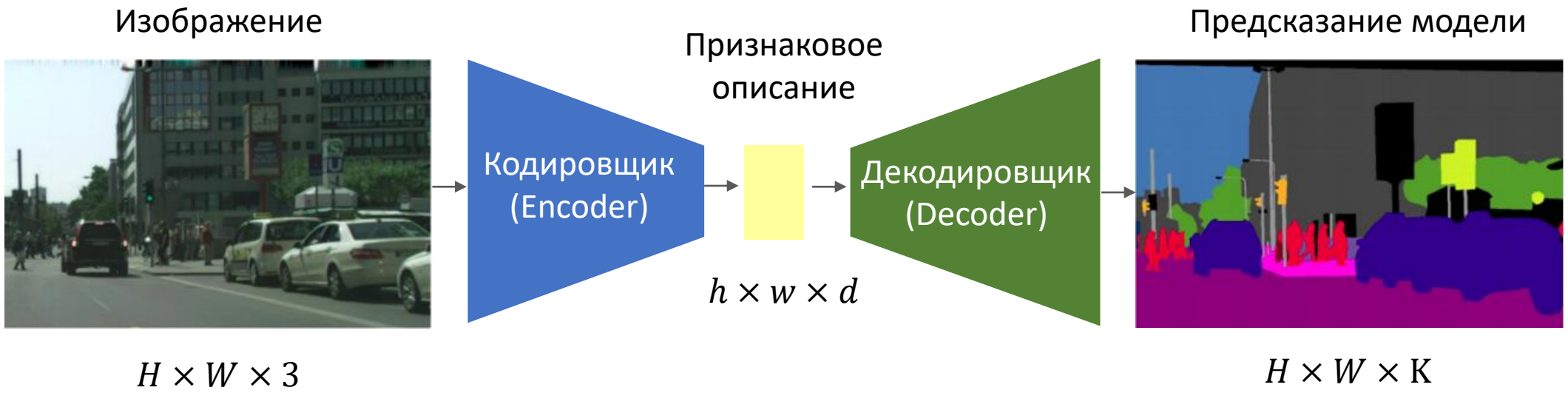


$H \times W \times \mathrm{K}$

Модель семантической сегментации

Большинство моделей семантической сегментации
устроено по принципу кодировщик-декодировщик (encoder-decoder).

- **Кодировщик** (Encoder) преобразует изображение в его признаковое описание,
  тензор размера $h \times w \times d$.
  Обычно кодировщик — Feature Extractor часть любой сети для классификации изображений.
- **Декодировщик** (Decoder) преобразует тензор признаков в картинку.

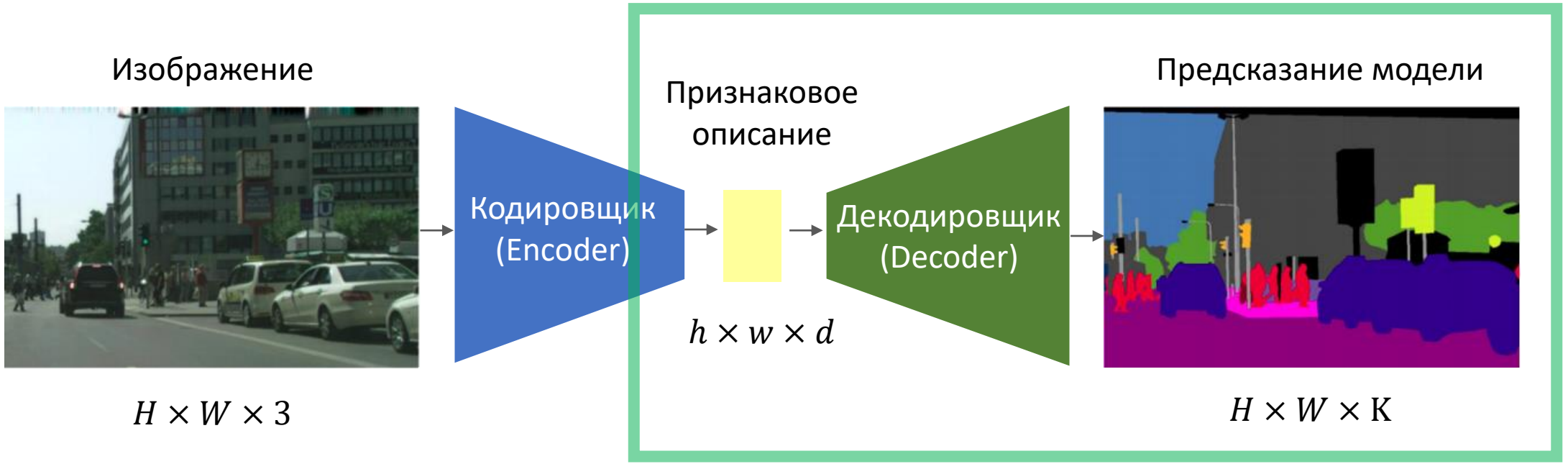


Модель семантической сегментации

Как делать декодировщик?

Нужна операция, которая из тензора размера $h \times w \times d$,
получает тензор размер $H \times W \times K$, где $h < H, w < W$.

Такая операция называется **Upsampling**.

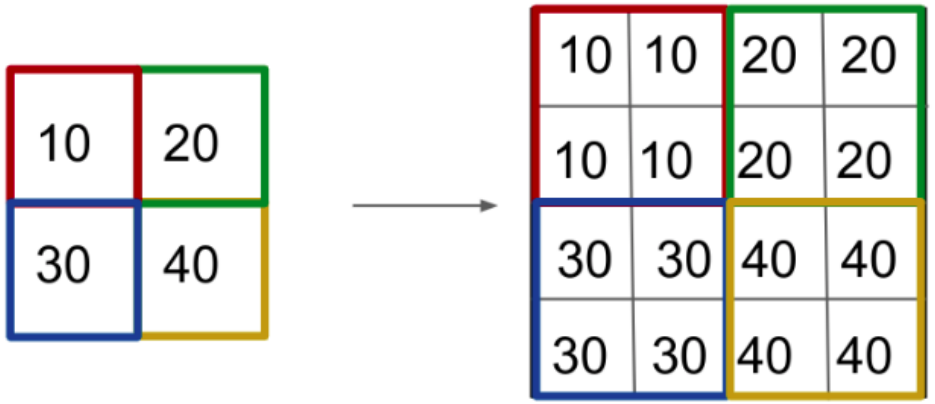


Виды Upsampling

- Интерполяция / interpolation
- Обратный пулинг / inverse pooling / unpooling
- Обратная свертка / Deconvolution / Transposed Convolution

Upsampling. Интерполяция

По ближайшему соседу
(Nearest Neighbours)



Upsampling. Интерполяция

Билинейная (Bilinear)

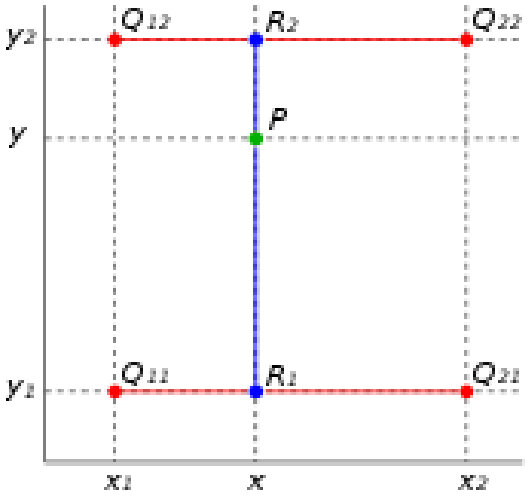
Сначала вычислим линейную интерполяцию по оси x

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Затем вычислим линейную интерполяцию по оси y

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2).$$

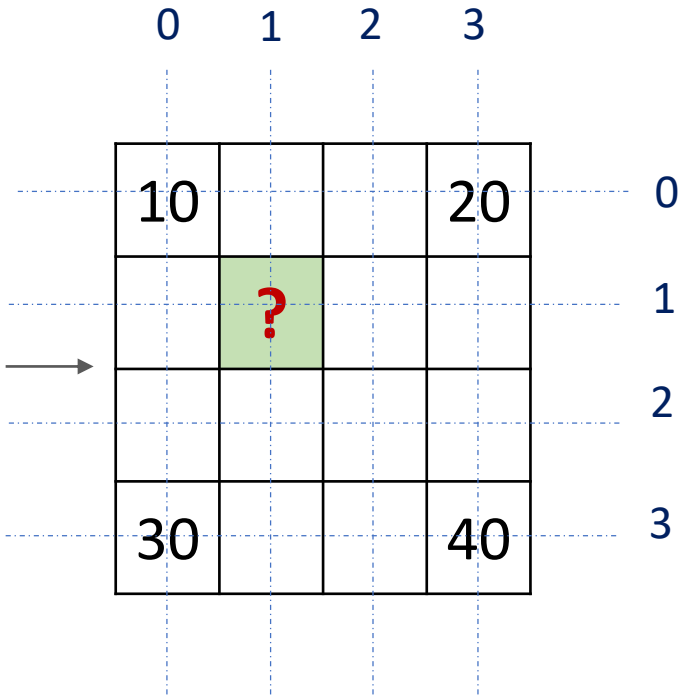


Upsampling. Интерполяция

Билинейная (Bilinear). Пример

Перенесем значения исходной матрицы таким образом, чтобы по краям новой матрицы матрицы оказались краевые значения исходной матрицы.

10	20
30	40



$$f(x,y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) = \frac{2}{3} * 10 + \frac{1}{3} * 20 = \frac{40}{3},$$

$$f(x,y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) = \frac{2}{3} * 30 + \frac{1}{3} * 40 = \frac{100}{3},$$

$$f(x,y) = \frac{y_2 - y}{y_2 - y_1} f(x,y_1) + \frac{y - y_1}{y_2 - y_1} f(x,y_2) = \frac{2}{3} * \frac{40}{3} + \frac{1}{3} * \frac{100}{3} = 20.$$

Upsampling. Интерполяция

Билинейная (Bilinear). Пример

Перенесем значения исходной матрицы таким образом, чтобы по краям новой матрицы матрицы оказались краевые значения исходной матрицы.

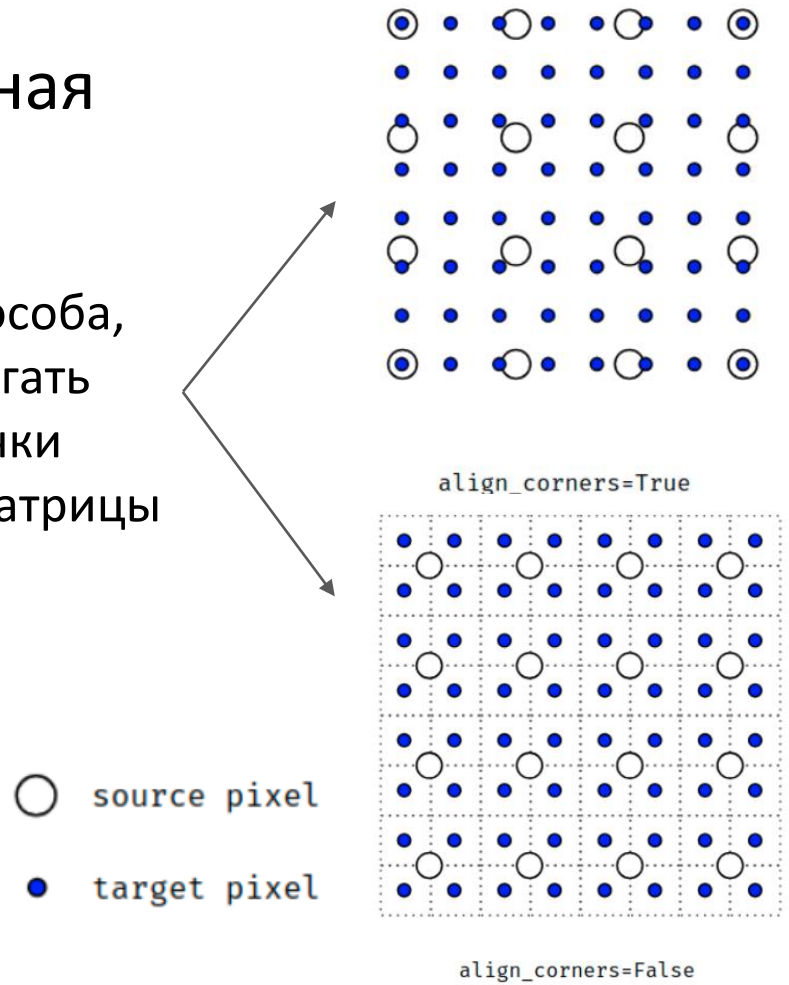
10	20
30	40

10	13	17	20
17	20	23	27
23	27	30	33
30	33	37	40

Upsampling. Интерполяция

Билинейная (Bilinear)

Есть два способа, как располагать краевые точки исходной матрицы в новой.



10	20
30	40

→

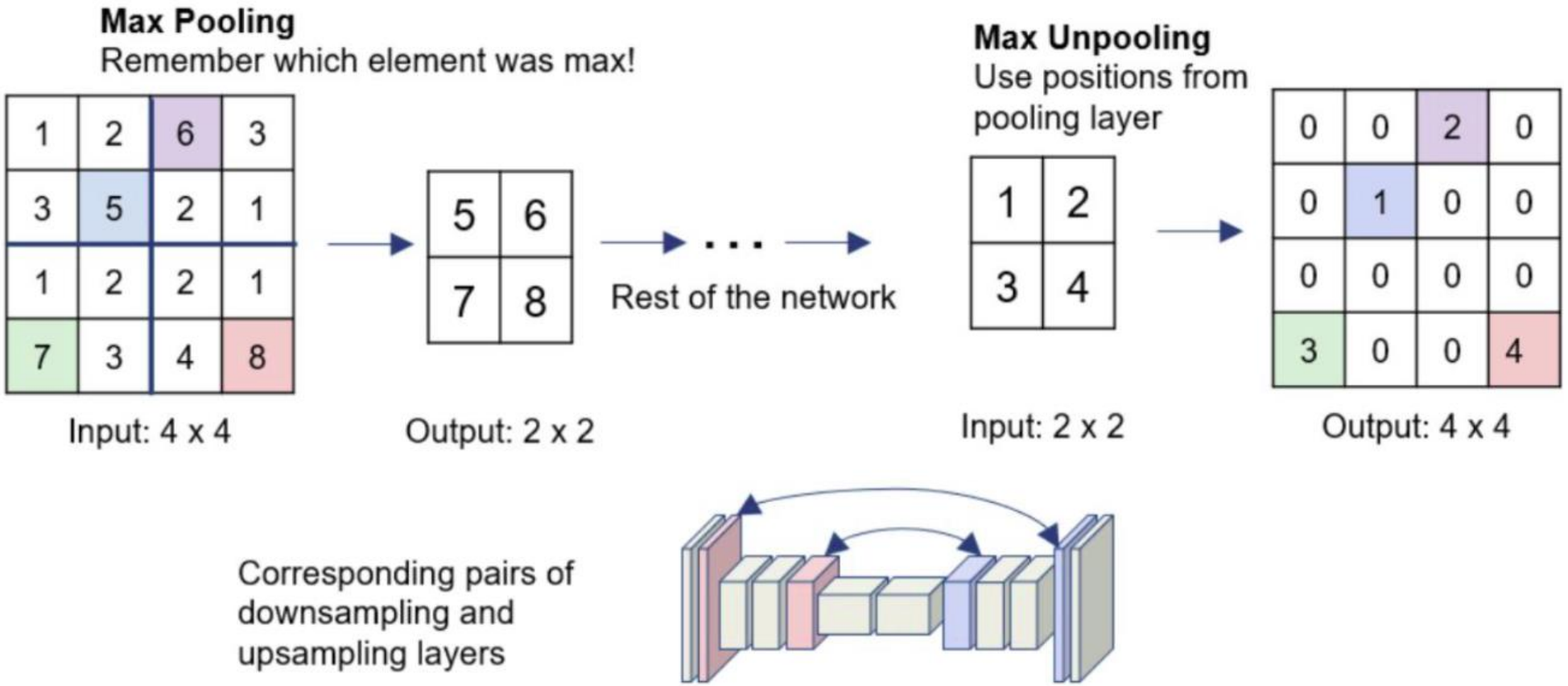
10	13	17	20
17	20	23	27
23	27	30	33
30	33	37	40

10	20
30	40

→

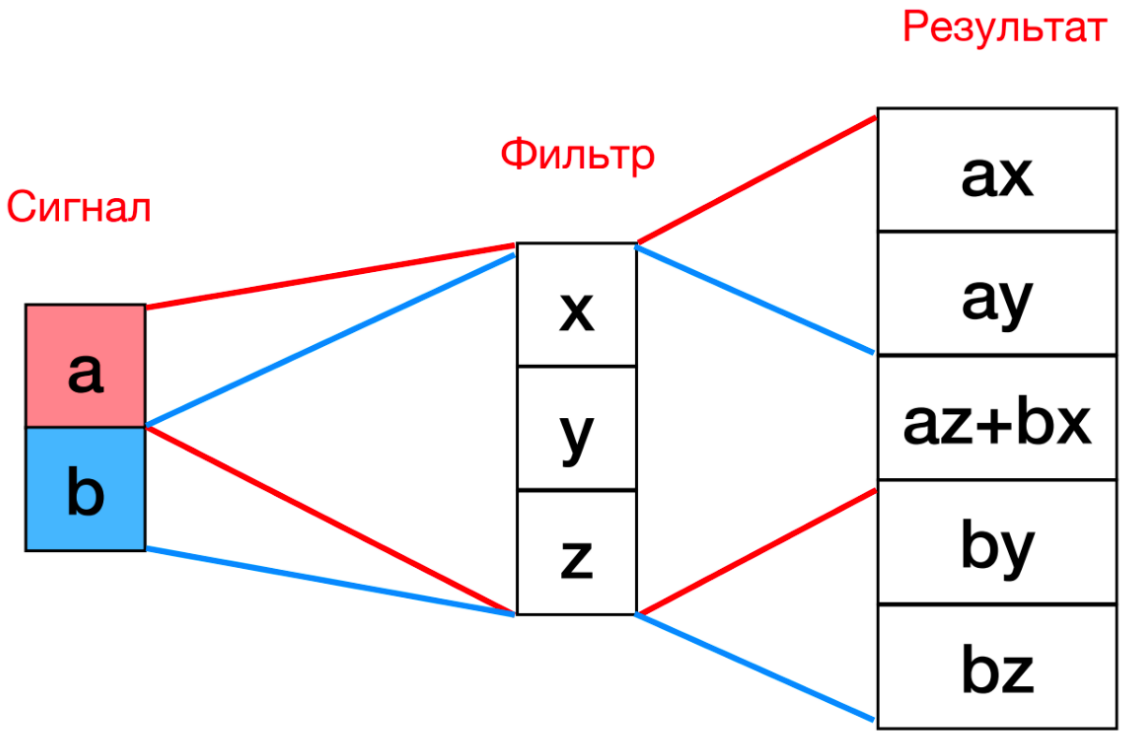
10	13	18	20
15	18	23	25
25	28	33	35
30	33	38	40

Upsampling. Обратный пулинг



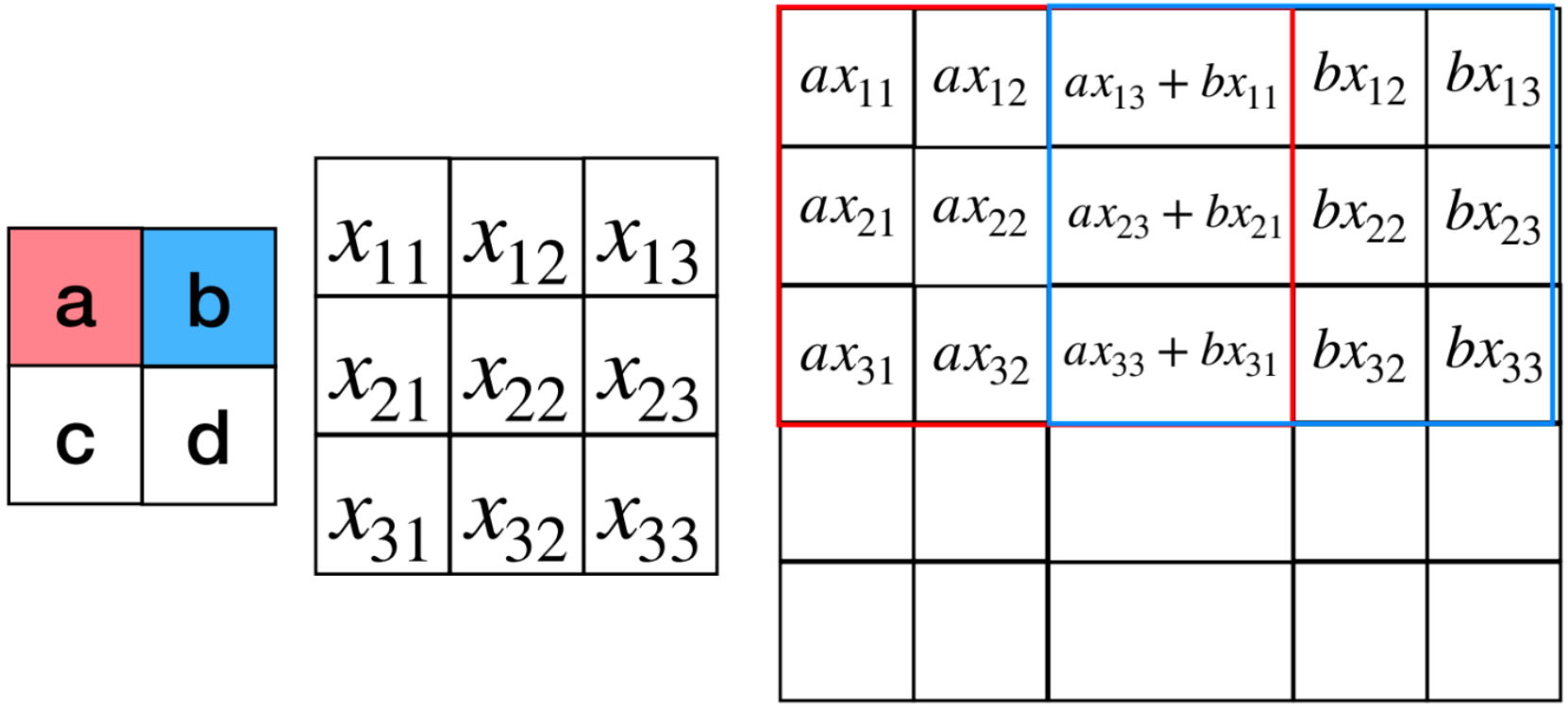
Upsampling. Обратная свертка

Одномерный случай



Upsampling. Обратная свертка

Двумерный случай



Upsampling. Обратная свертка

- Операцию обратной свертки можно представить в виде ленейной операции,
  по аналогии с обычной сверткой.
  Значит, она **обучаема**!
- Padding и stride в обратной свертке можно интерпертировать по-разному.
  Рассмотрим то, как они **представлены в pytorch**.
- **Padding** — используется для приведения к нужному размеру.
  Удаляем из полученного feature map нейронны так,
  чтобы если бы применили свертку (не обратную)
  с такими же параметрами padding и stride к новому feature map,
  то получили бы feature map такого же размера как исходное изображение.
- **Stride** — шаг, с которым результаты сверток накладываются друг на друга.
  Чем больше stride, тем больше feature map на выходе.

Upsampling. Обратная свертка

вход **ядро**

input

2	4
0	1

k

3	1
1	5

stride=1

s=1, p=0, выход 3x3

s=1

6	2	-
2	10	-
-	-	-

+

-	12	4
-	4	20
-	-	-

+

-	-	-
0	0	-
0	0	-

+

-	-	-
-	3	1
-	1	5

=

6	14	4
2	17	21
0	1	5

s=1, p=1, выход 1x1

s=1 p=1

6	2	-
2	10	-
-	-	-

+

-	12	4
-	4	20
-	-	-

+

-	-	-
0	0	-
0	0	-

+

-	-	-
-	3	1
-	1	5

=

6	14	4
2	17	21
0	1	5

Upsampling. Обратная свертка

вход

input

2	4
0	1

ядро

k

3	1
1	5

stride=2

s=2, p=0, выход 4x4

s=2

6	2	-	-
2	10	-	-
-	-	-	-
-	-	-	-

+

-	-	12	4
-	-	4	20
-	-	-	-
-	-	-	-

+

-	-	-	-
-	-	-	-
0	0	-	-
0	0	-	-

+

-	-	-	-
-	-	-	-
-	-	3	1
-	-	1	5

=

6	2	12	4
2	10	4	20
0	0	3	1
0	0	1	5

s=2, p=1, выход 2x2

s=2 p=1

6	2	-	-
2	10	-	-
-	-	-	-
-	-	-	-

+

-	-	12	4
-	-	4	20
-	-	-	-
-	-	-	-

+

-	-	-	-
-	-	-	-
0	0	-	-
0	0	-	-

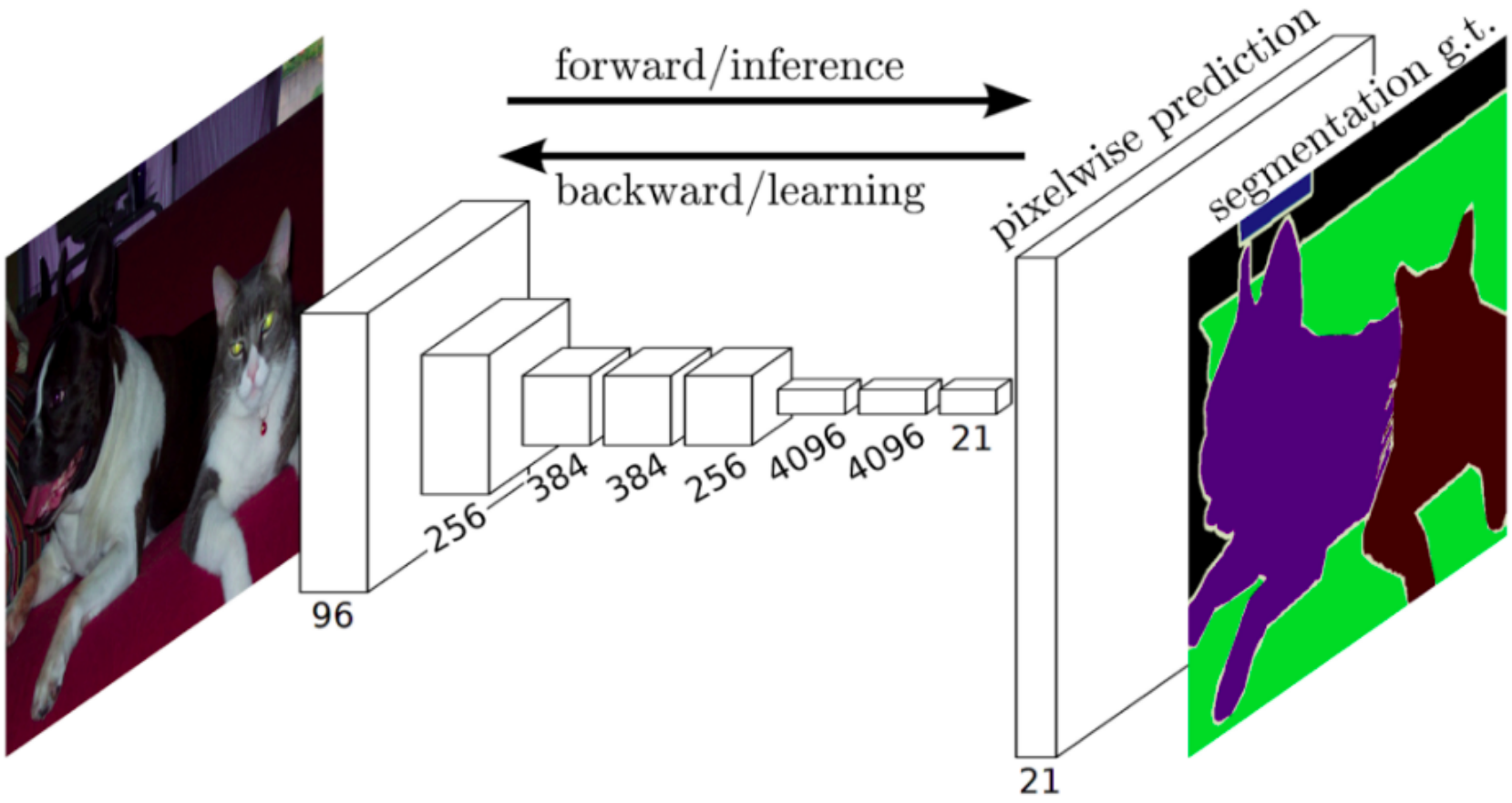
+

-	-	-	-
-	-	-	-
-	-	3	1
-	-	1	5

=

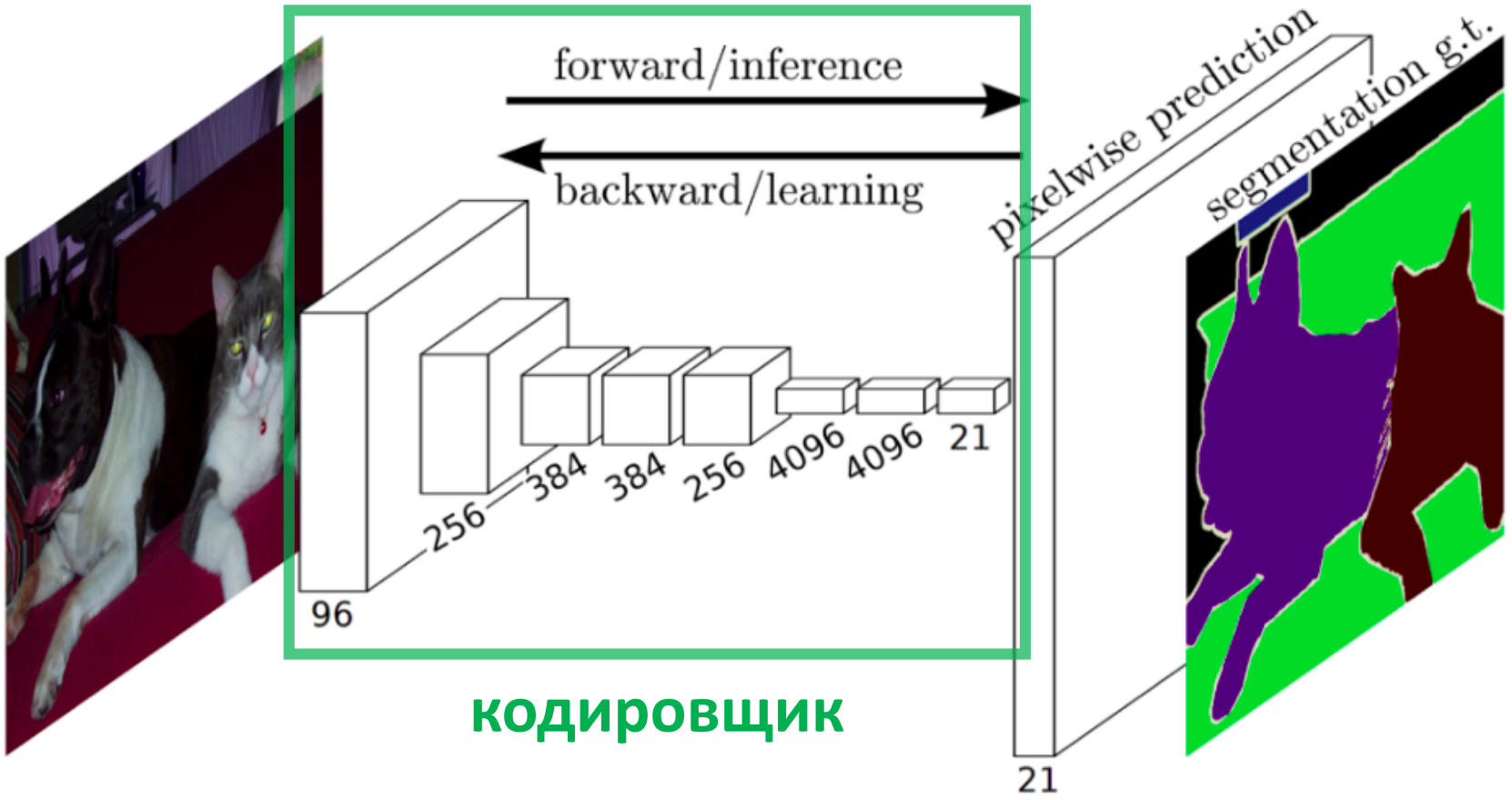
6	2	12	4
2	10	4	20
0	0	3	1
0	0	1	5

Fully-Convolutional Network (FCN)



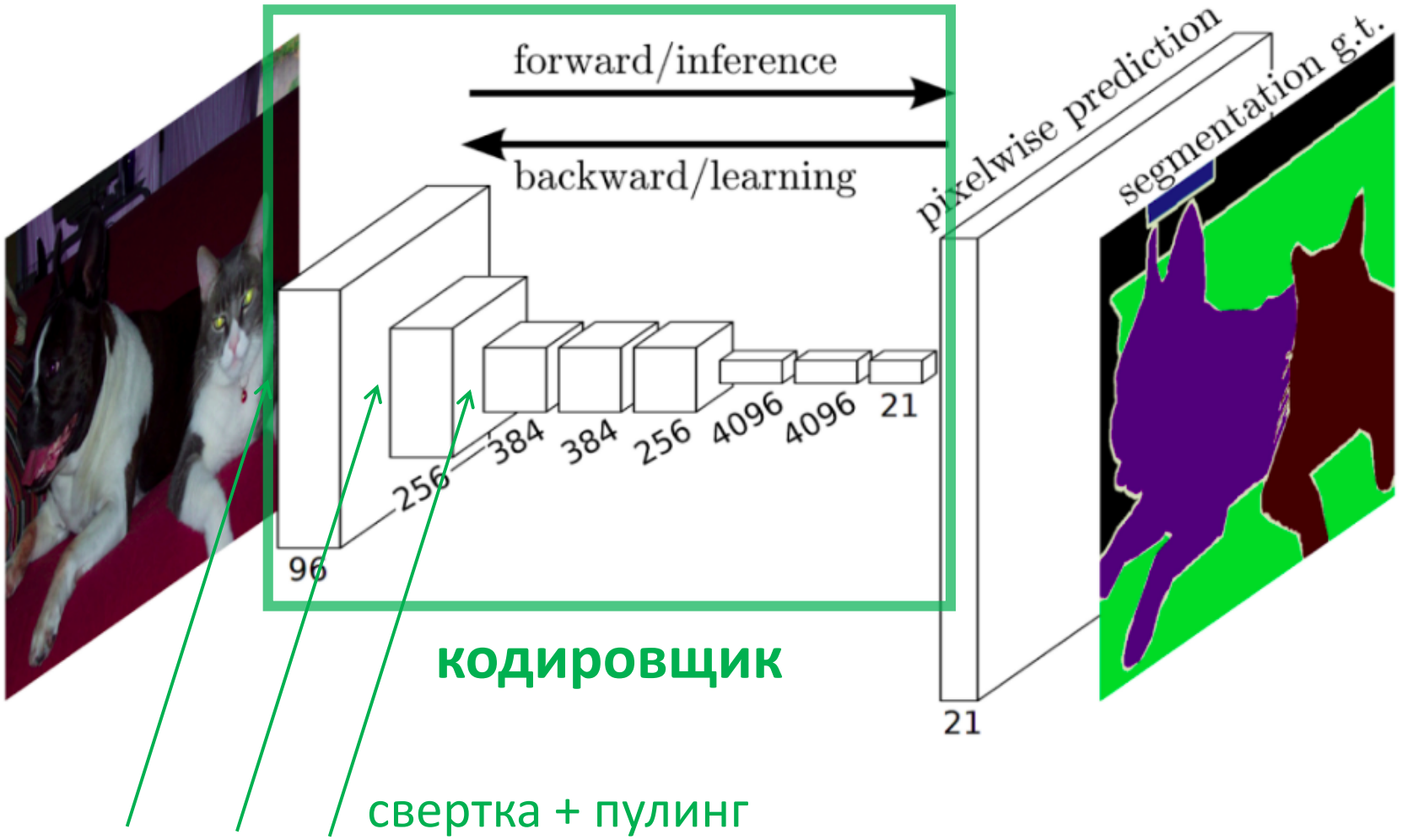
Статья

Fully-Convolutional Network (FCN)

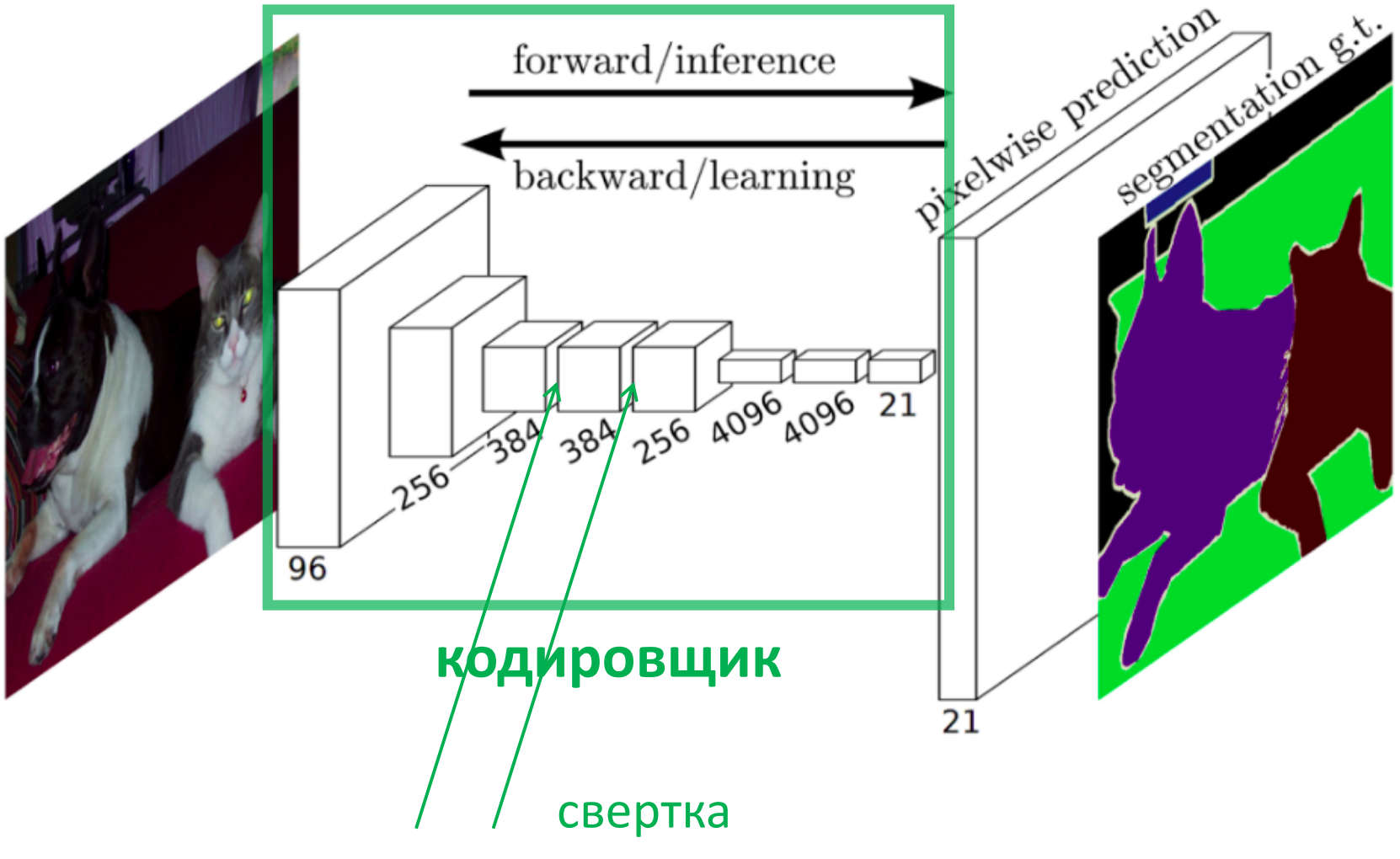


кодировщик

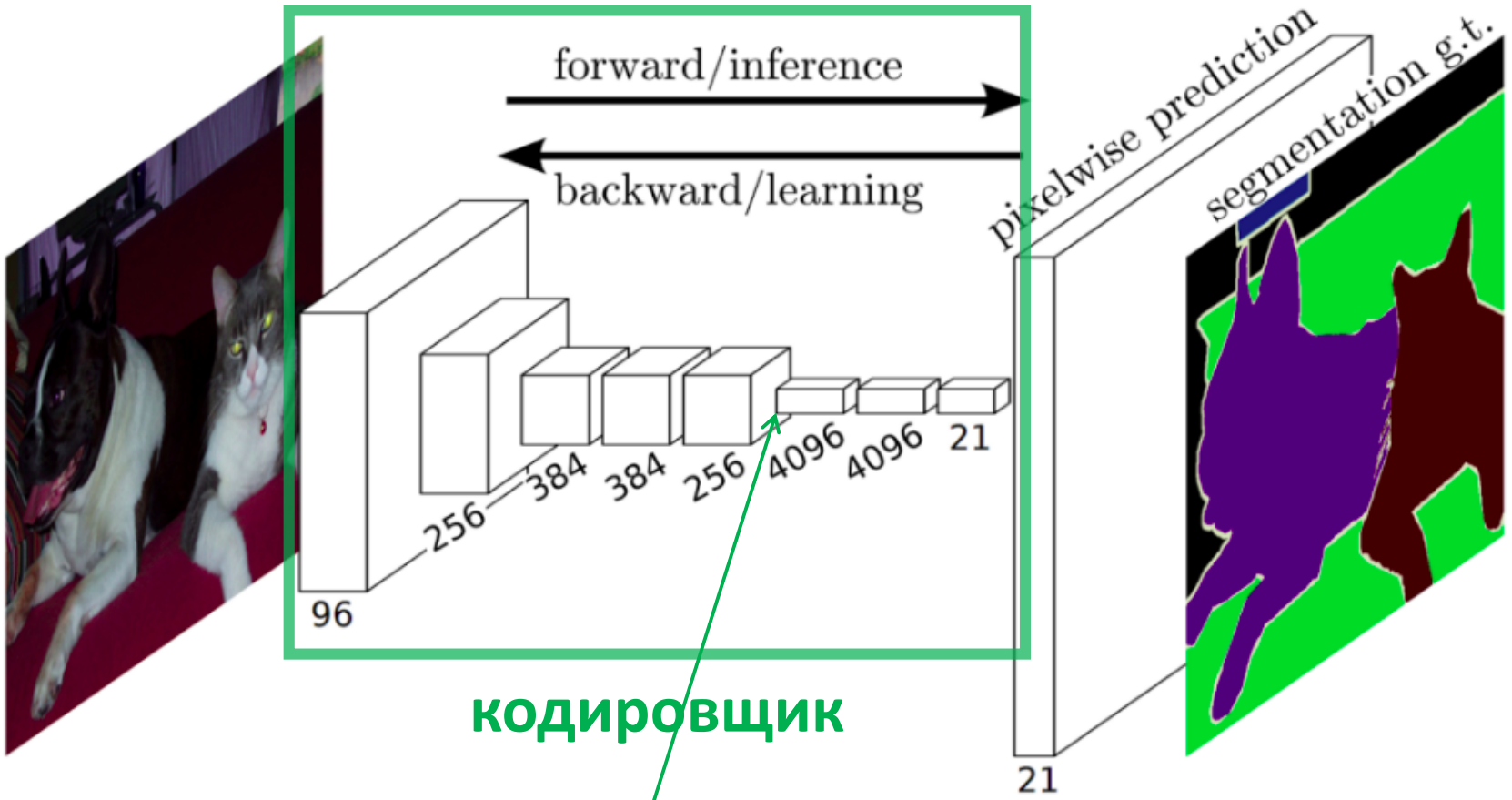
Fully-Convolutional Network (FCN)



Fully-Convolutional Network (FCN)



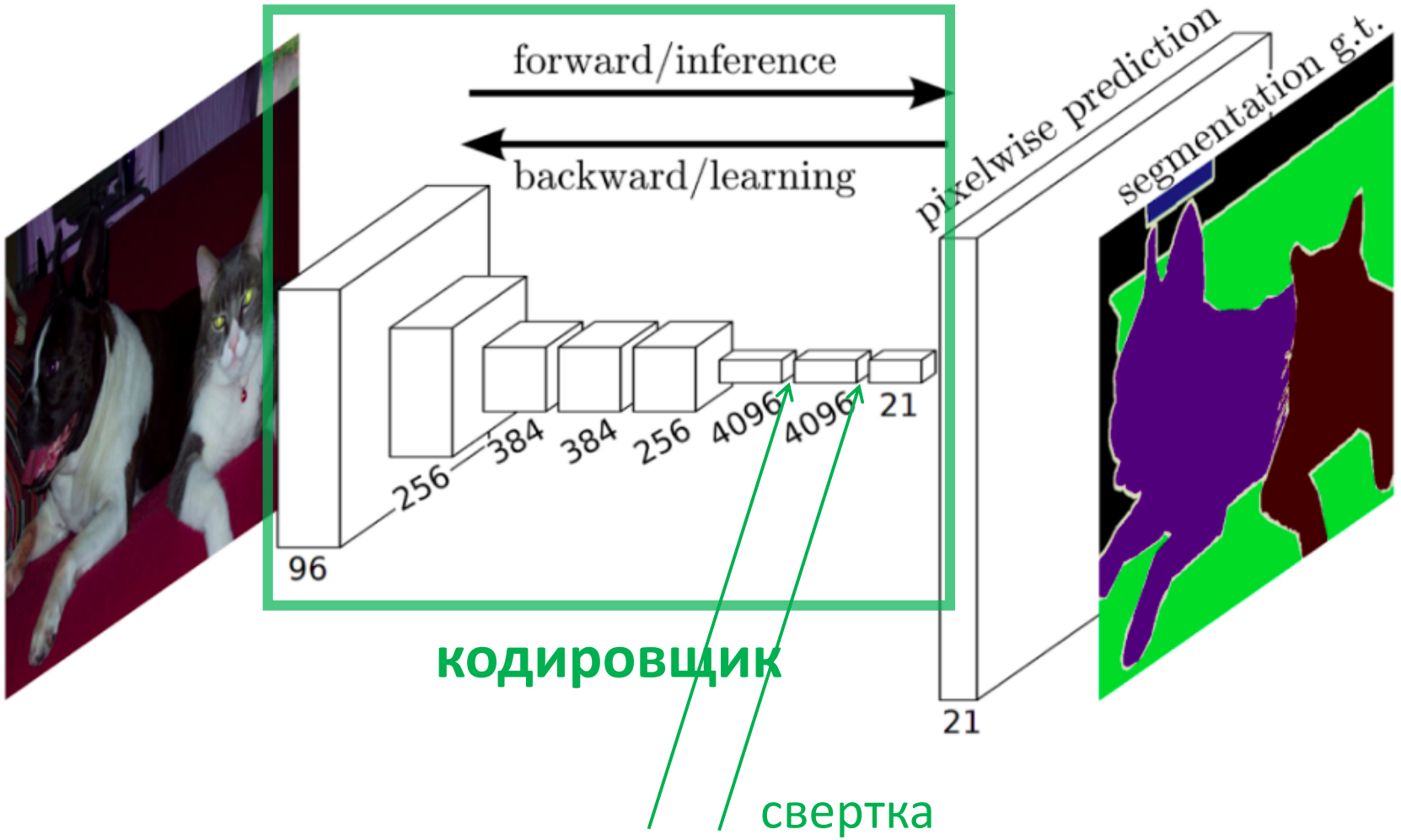
Fully-Convolutional Network (FCN)



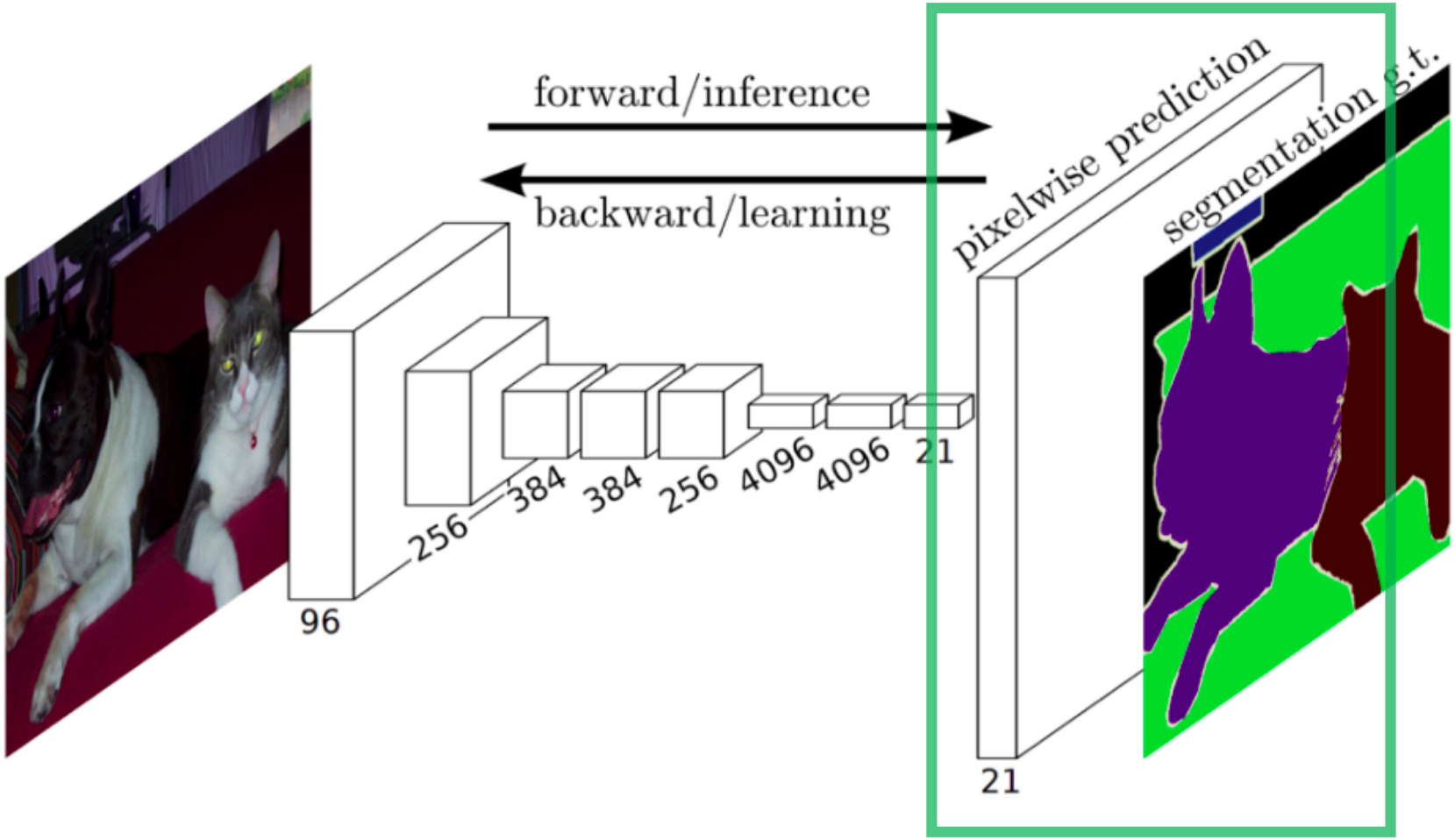
кодировщик

свертка + пулинг

Fully-Convolutional Network (FCN)

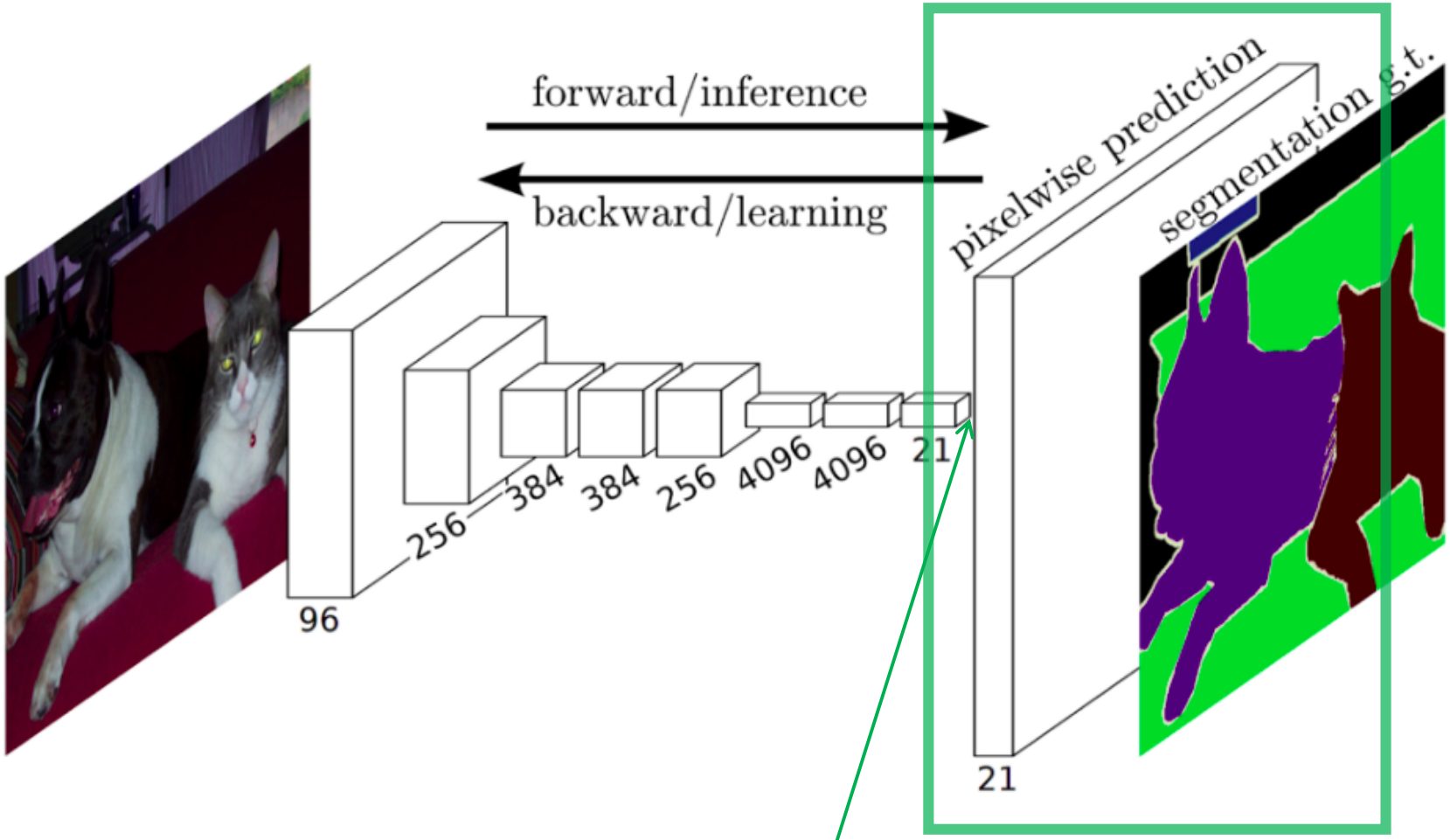


Fully-Convolutional Network (FCN)



декодировщик

Fully-Convolutional Network (FCN)



обратная свертка

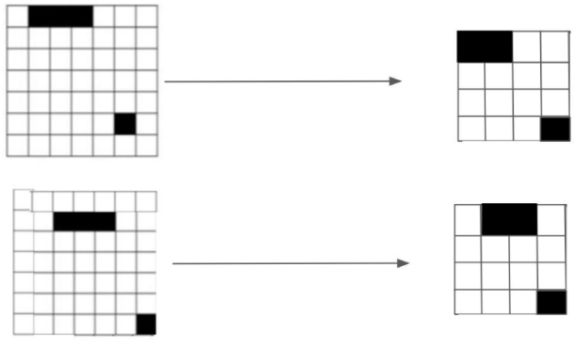
декодировщик

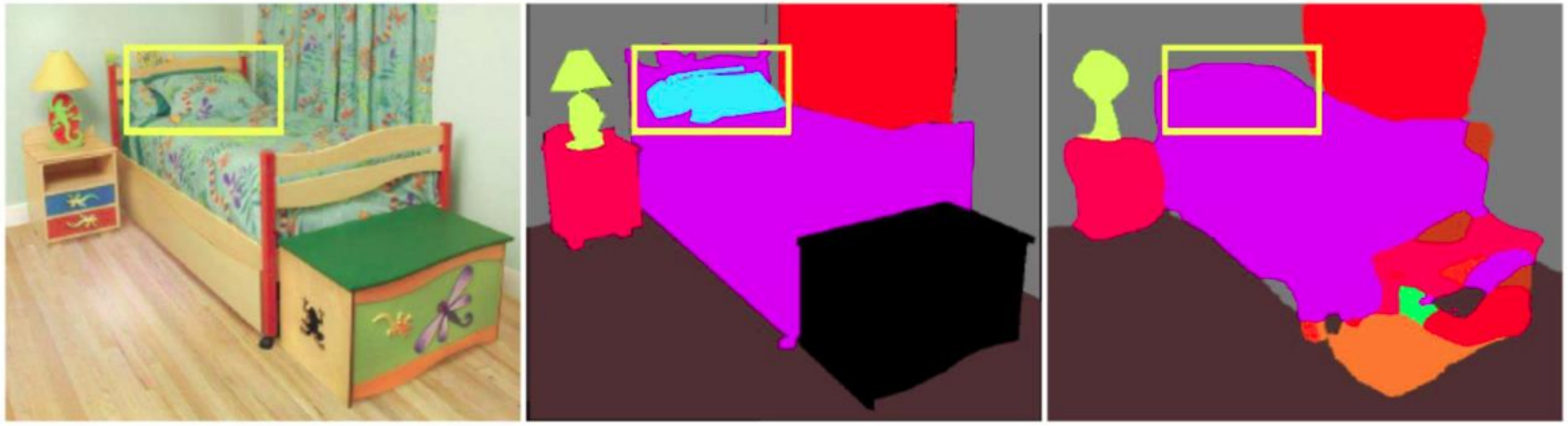
Fully-Convolutional Network (FCN)

Недостатки

При постоянном уменьшении размера возникает проблема **Scale Variability**.

Upsampling **плохо восстаналивает пространственную информацую** из сильно сжатого представления.



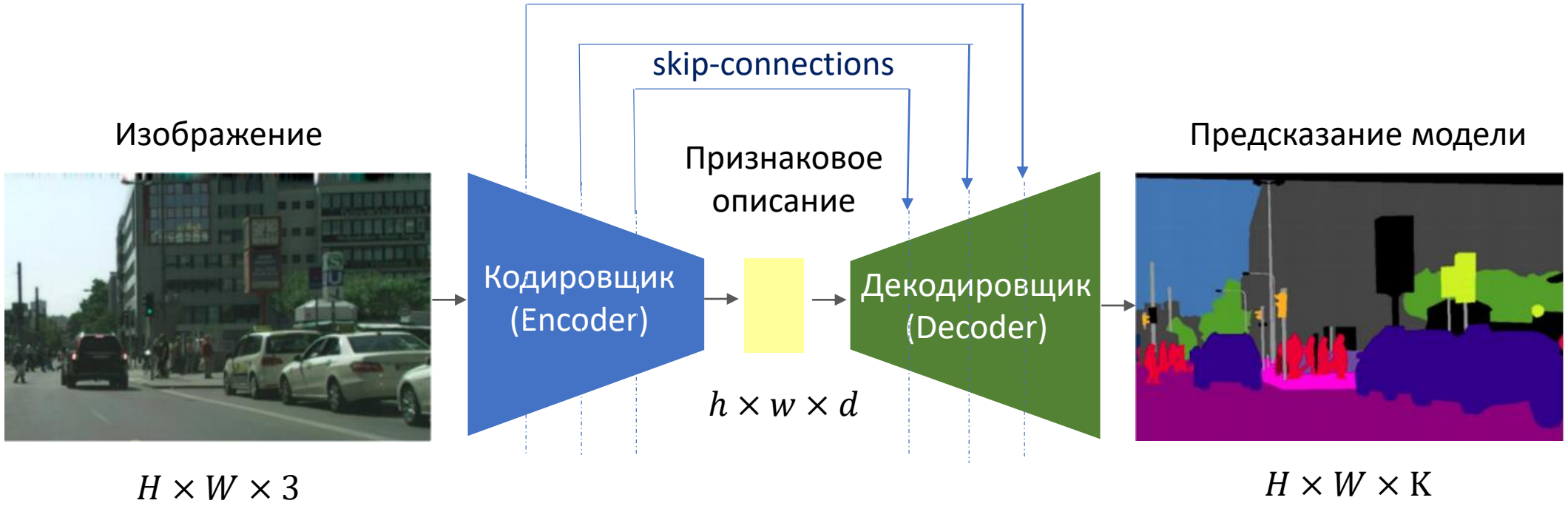


(a) Image (b) Ground Truth (c) FCN

Модели вида U-Net

Upsampling плохо восстанавливает пространственную информацию из сильно сжатого представления.

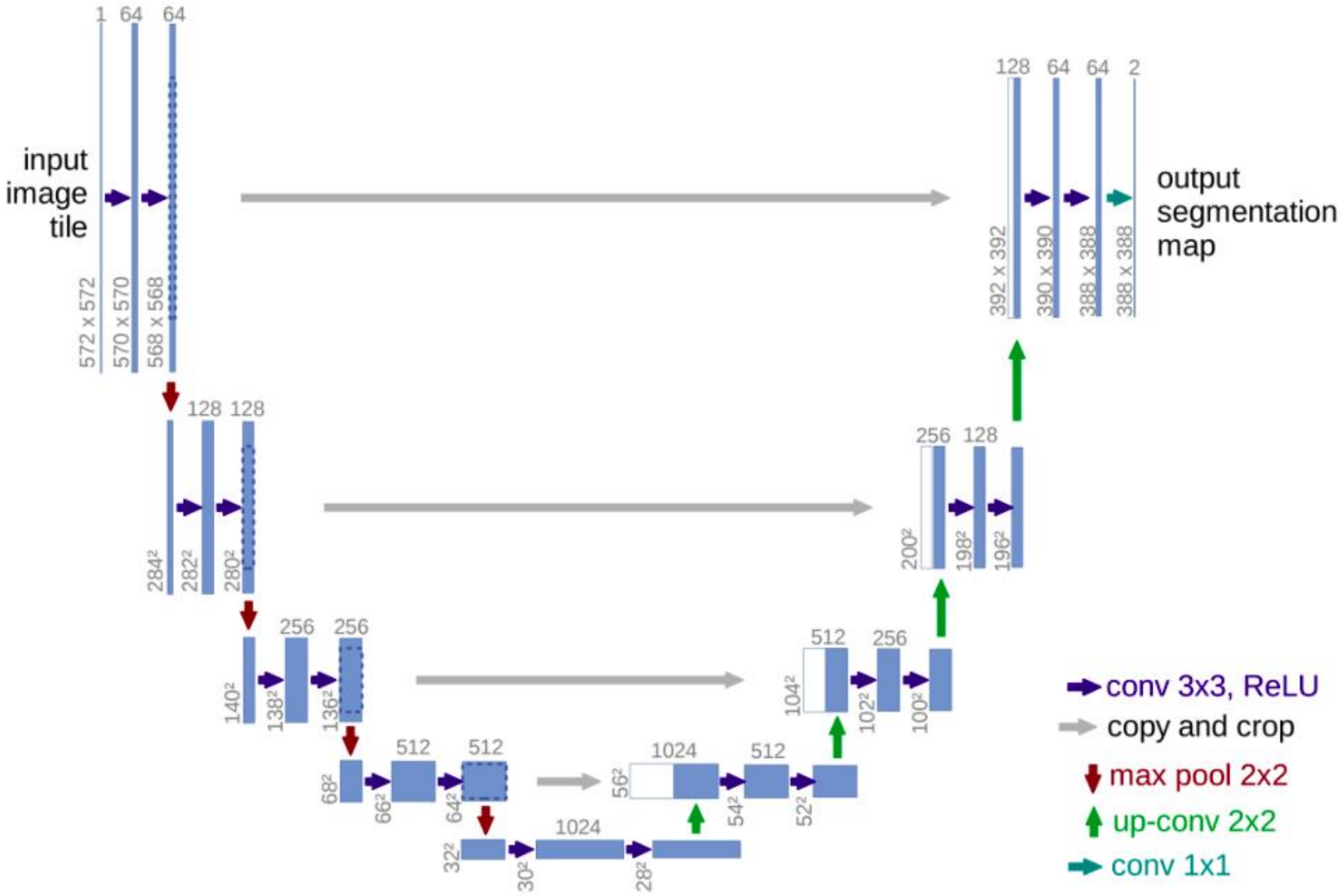
Будем использовать представления с более ранних слоев кодировщика в симметричных слоях декодировщика. Они содержат менее сжатую простр. информацию.



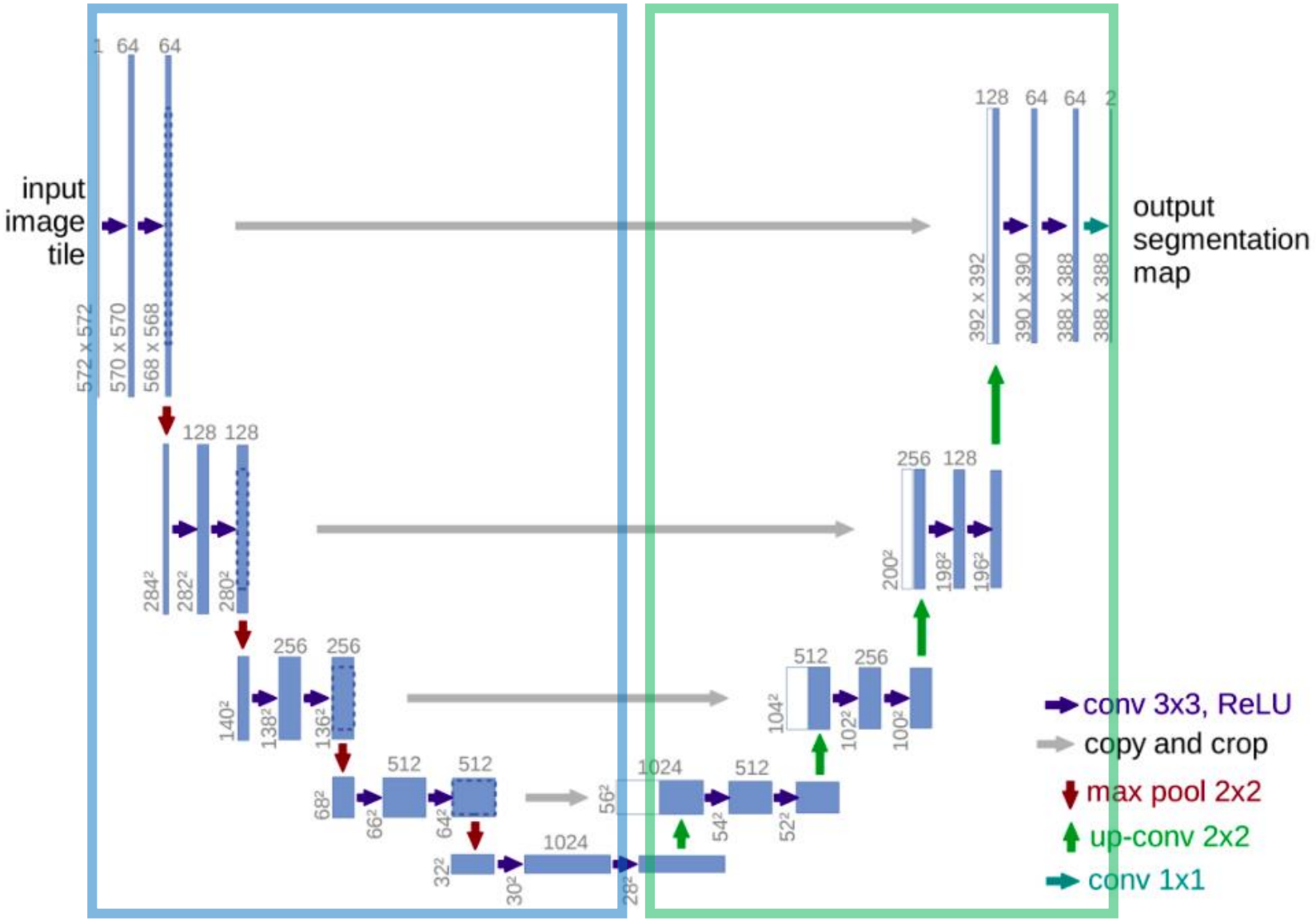
U-Net

Рассмотрим
первую версию Unet.

[Статья]



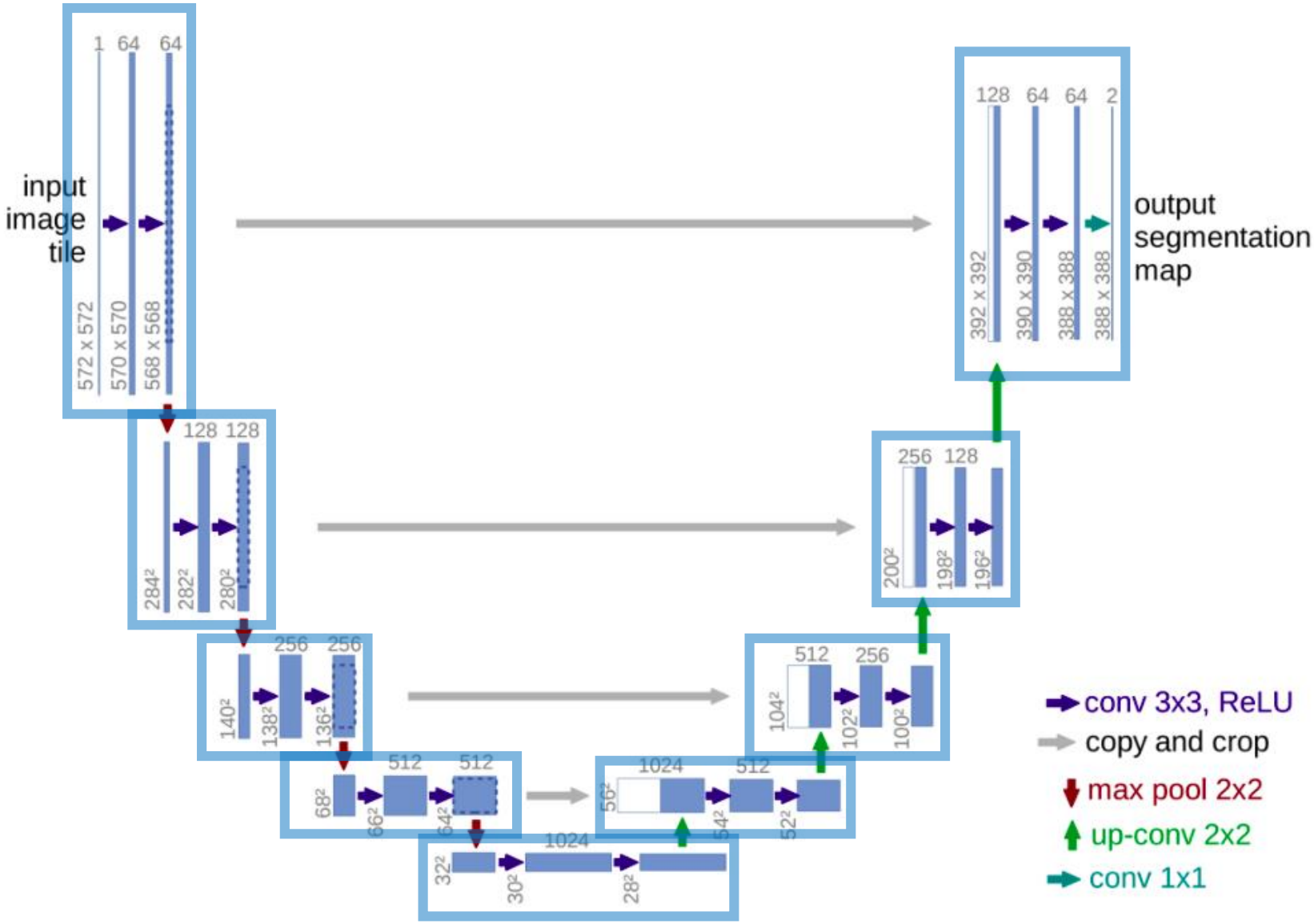
U-Net



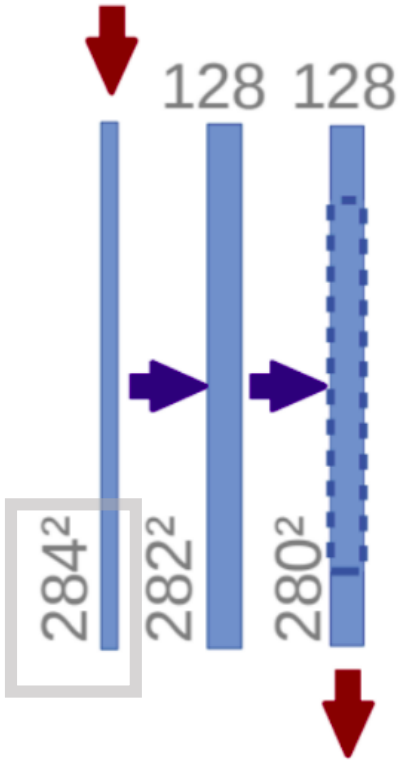
кодировщик **декодировщик**

U-Net

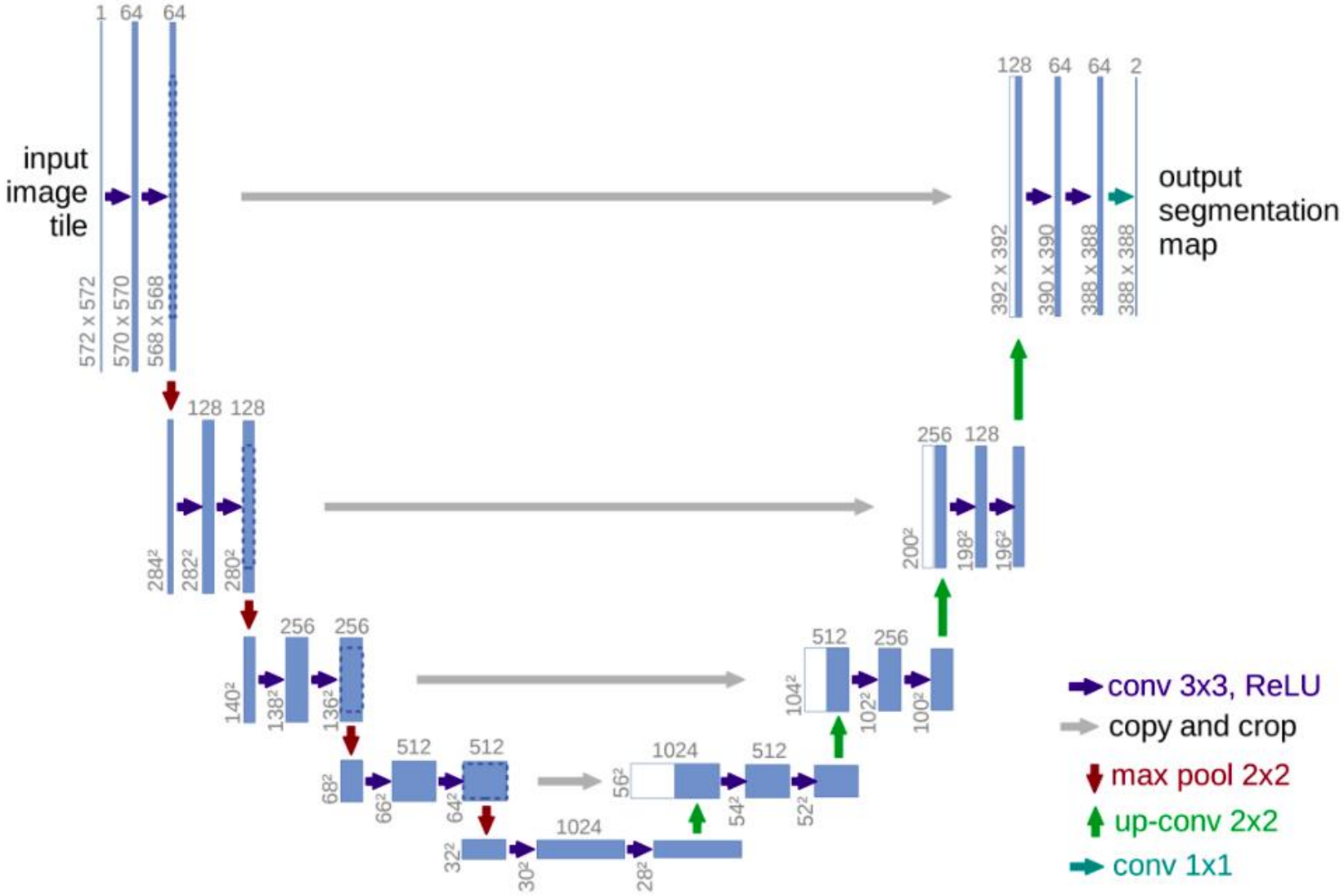
Обычные свертки



U-Net

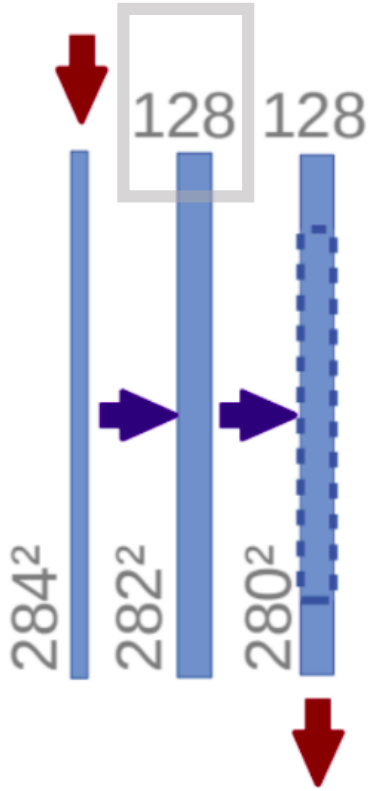


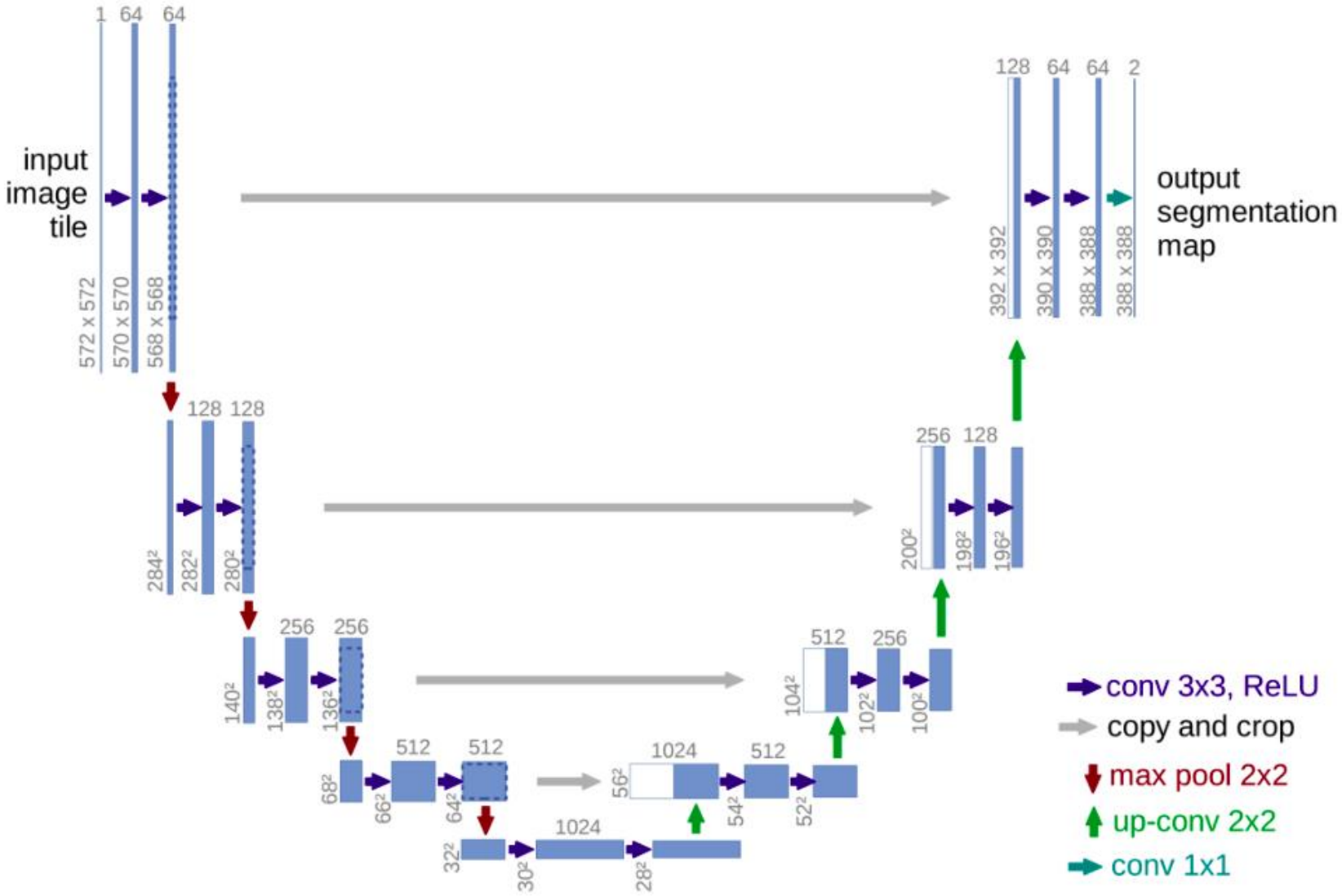
Текущие высота и ширина тензора



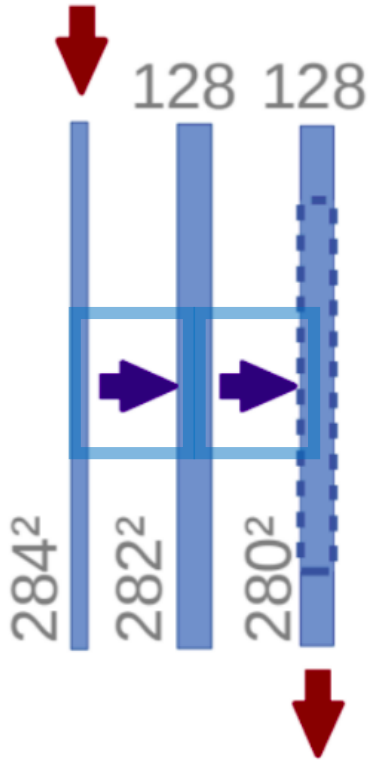
U-Net

Текущее кол-во каналов в тензоре

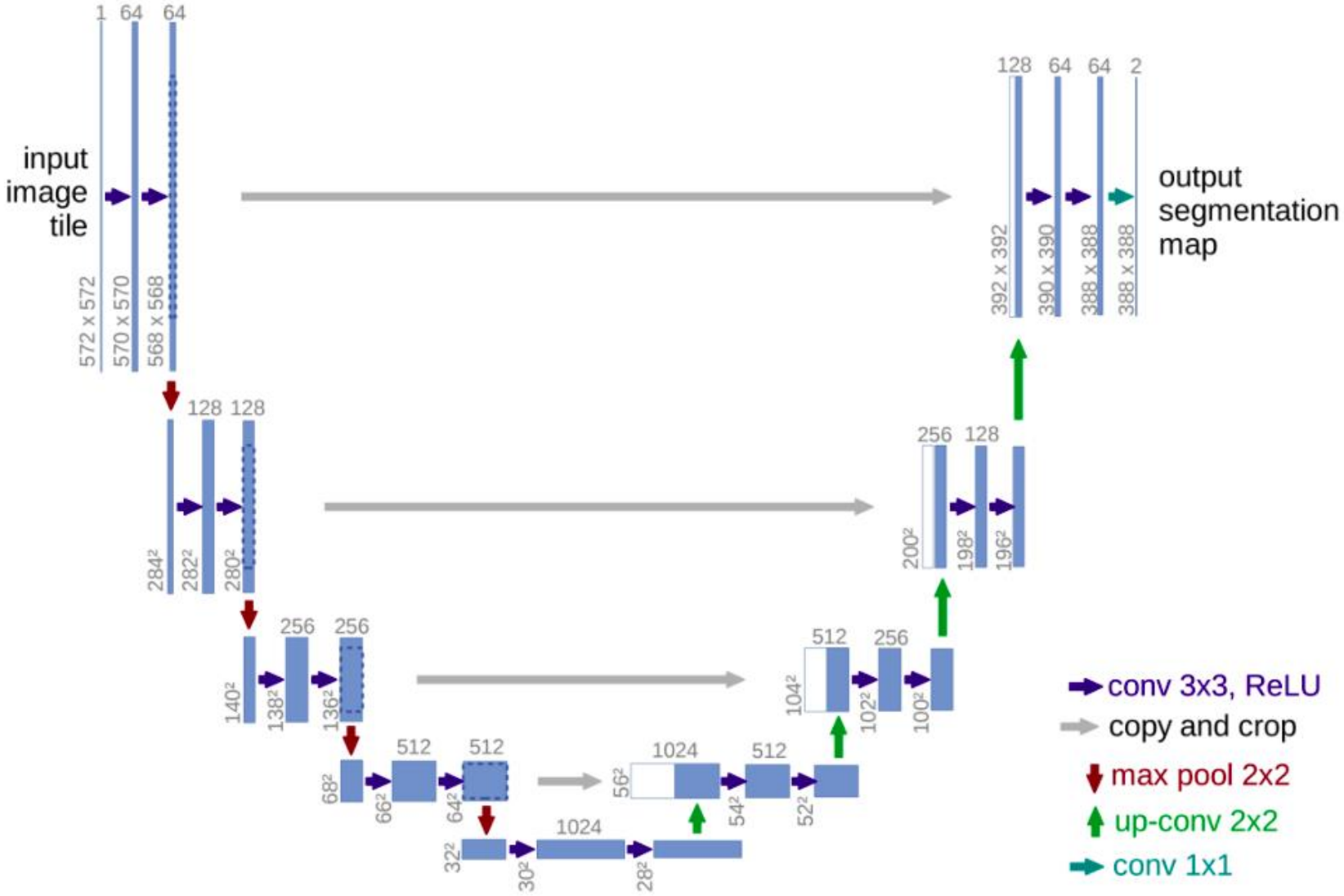




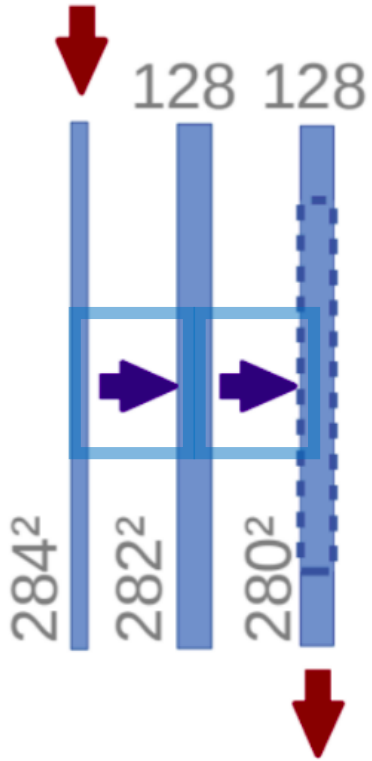
U-Net



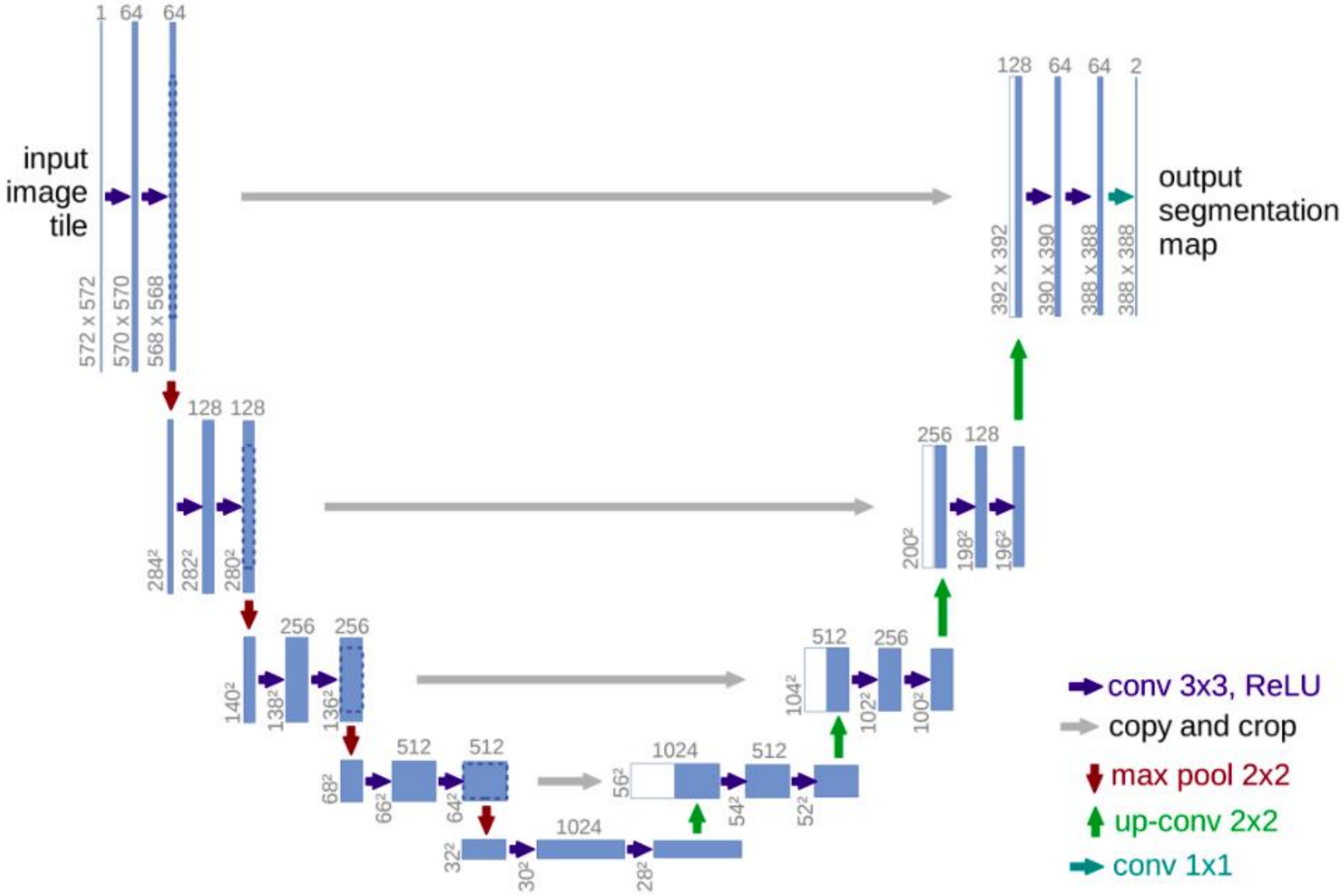
Свертка 3x3, p=0, s=1



U-Net

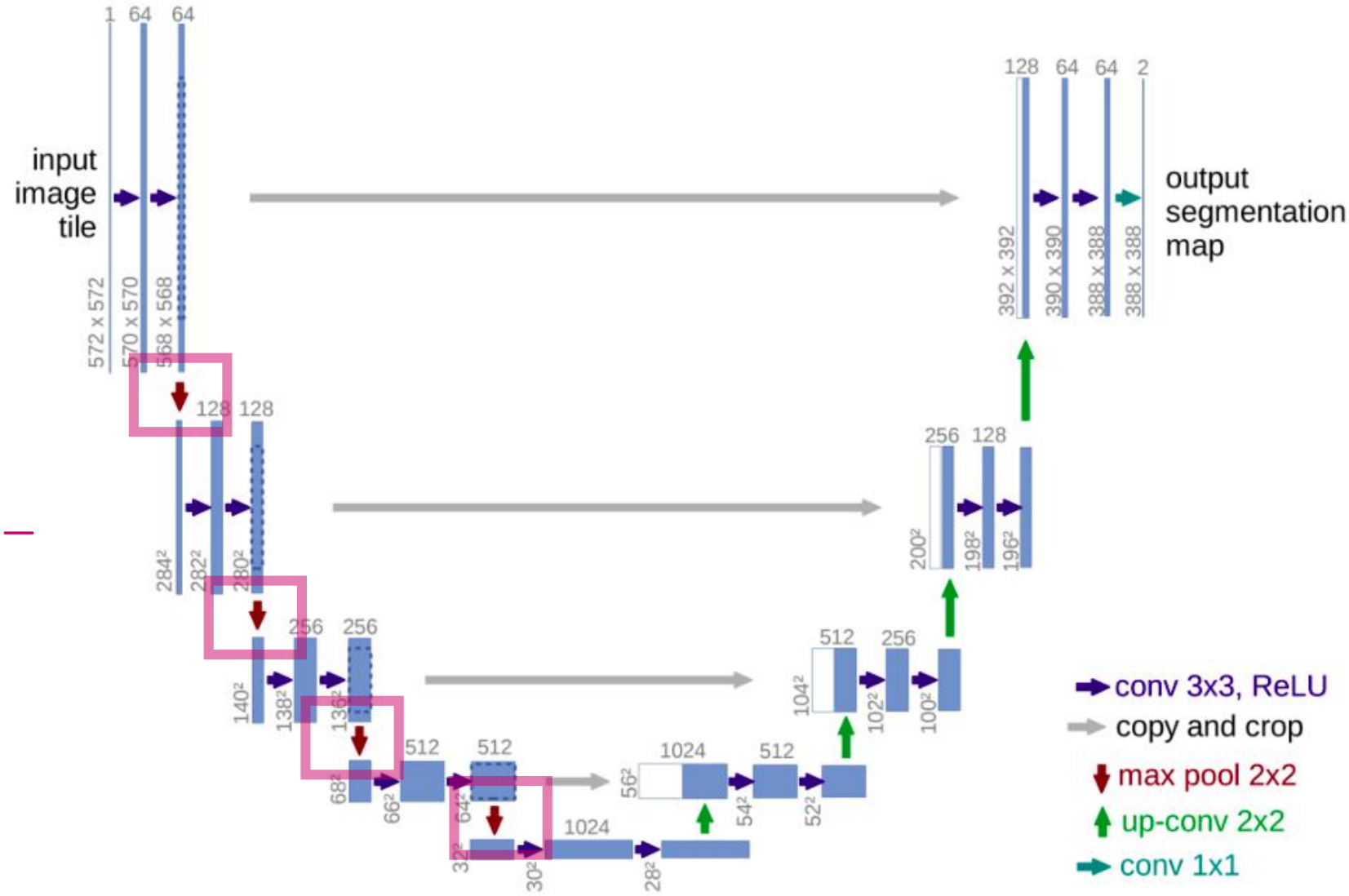


Свертка 3x3, p=0, s=1



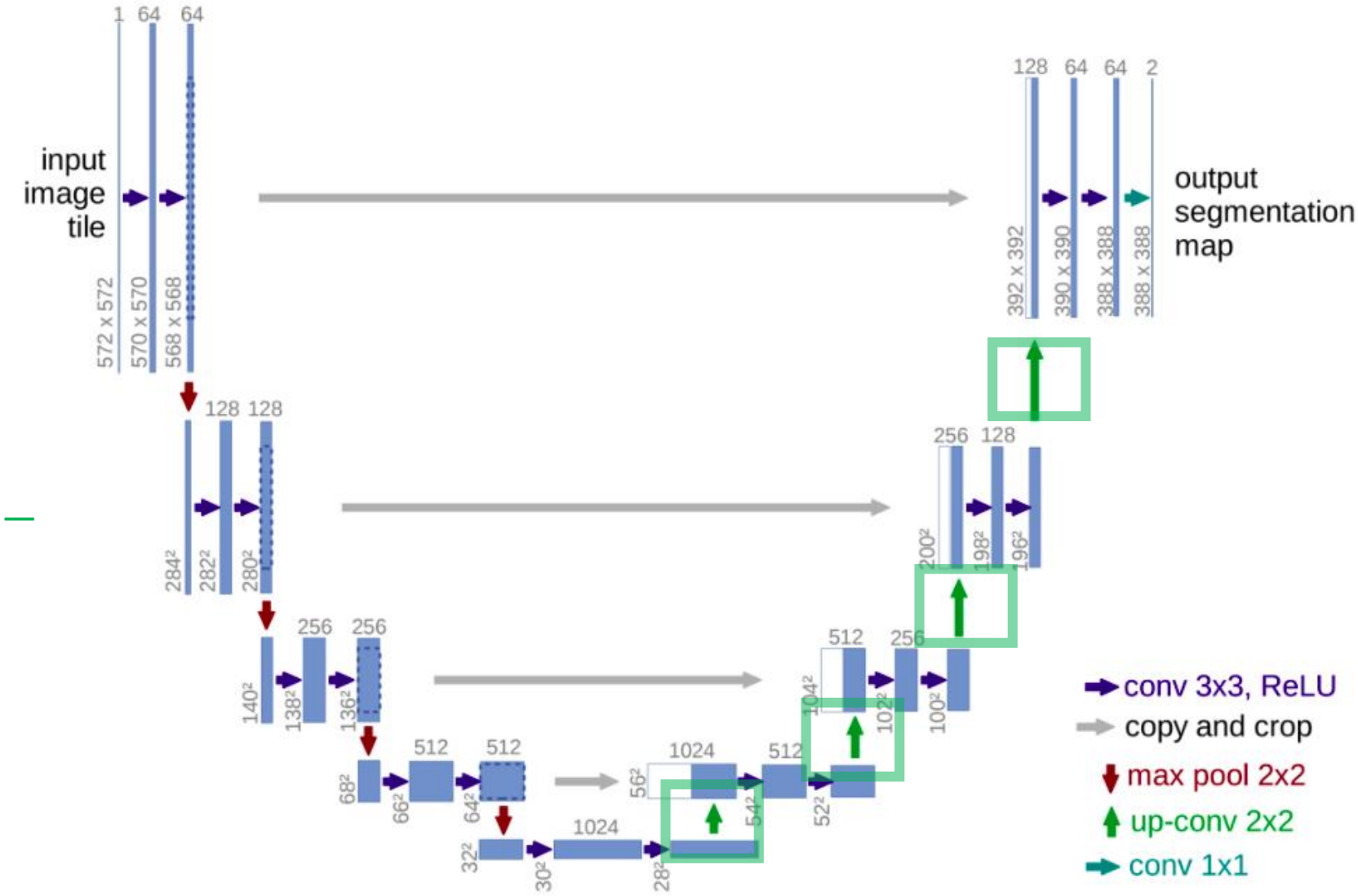
U-Net

Понижение размерности — max pooling 2 x 2



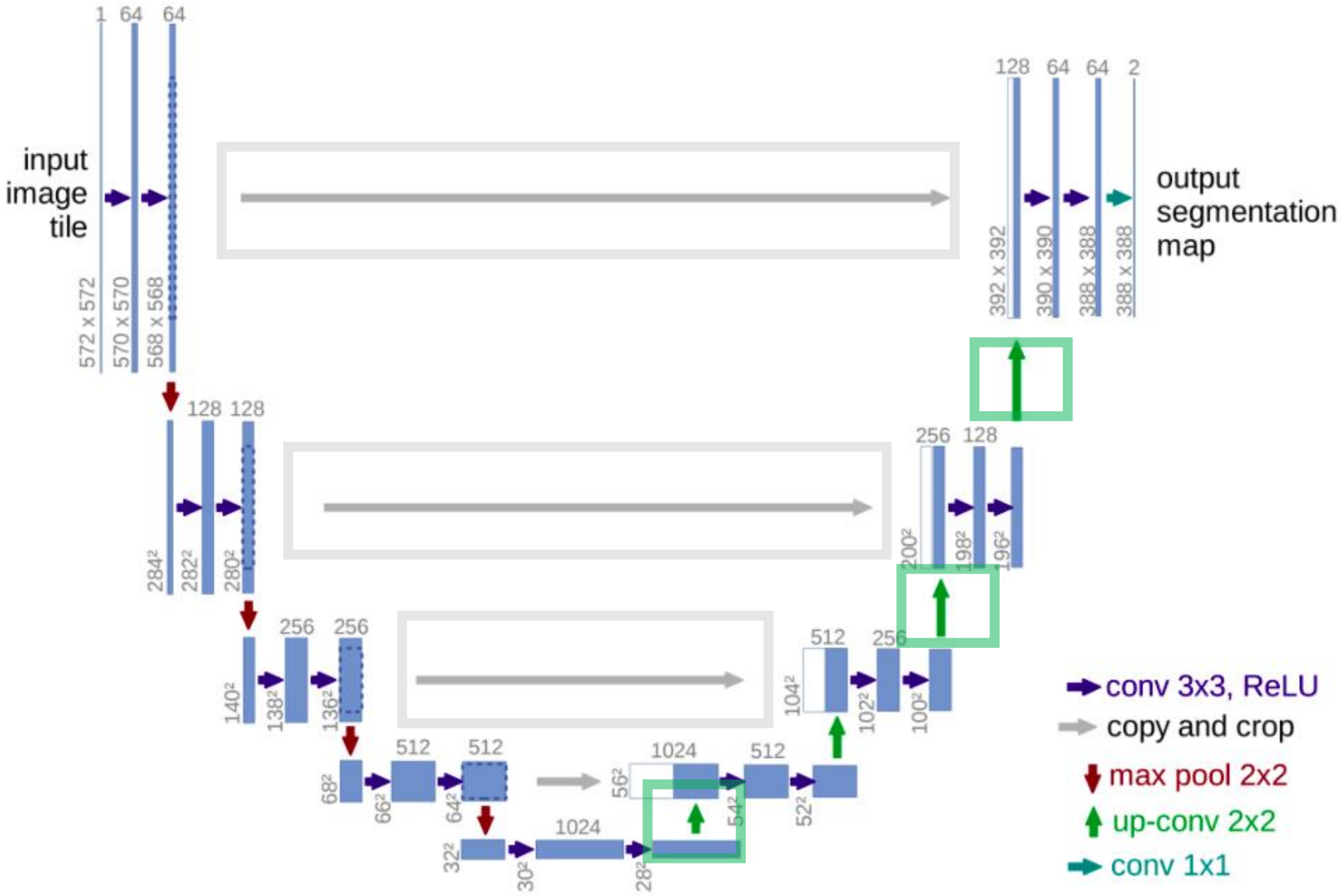
U-Net

Повышение размерности — обратная свертка 2 x 2



U-Net

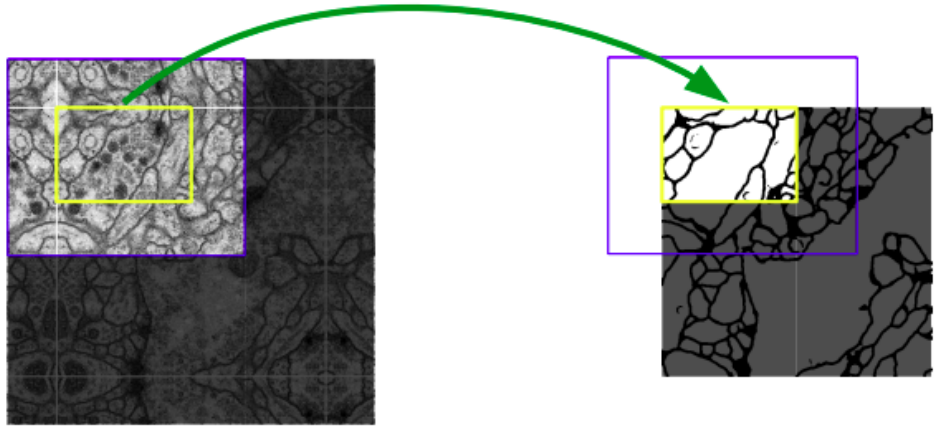
Обрезаем feature map с менее глубокого слоя. Конкатенируем его с соотв. более глубоким feature map по оси каналов

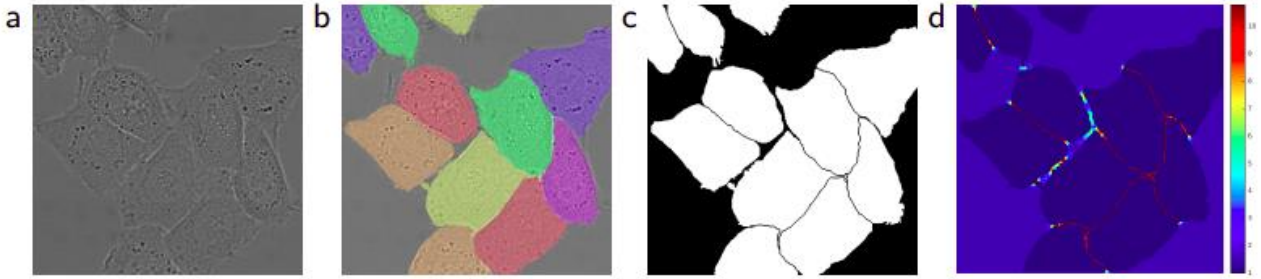


U-Net

Интересные особенности модели

- Модель была предложена для решения задачи **сегментации медицинских картинок**.
- Для того, чтобы не теряла информация на границах изображения, сделали **большой зеркальный padding**.
- Для того, чтобы границы клеток четко отображались нейронной сетью, **для лосса задавали доп. веса**, которые были больше там, где граница между клетками.





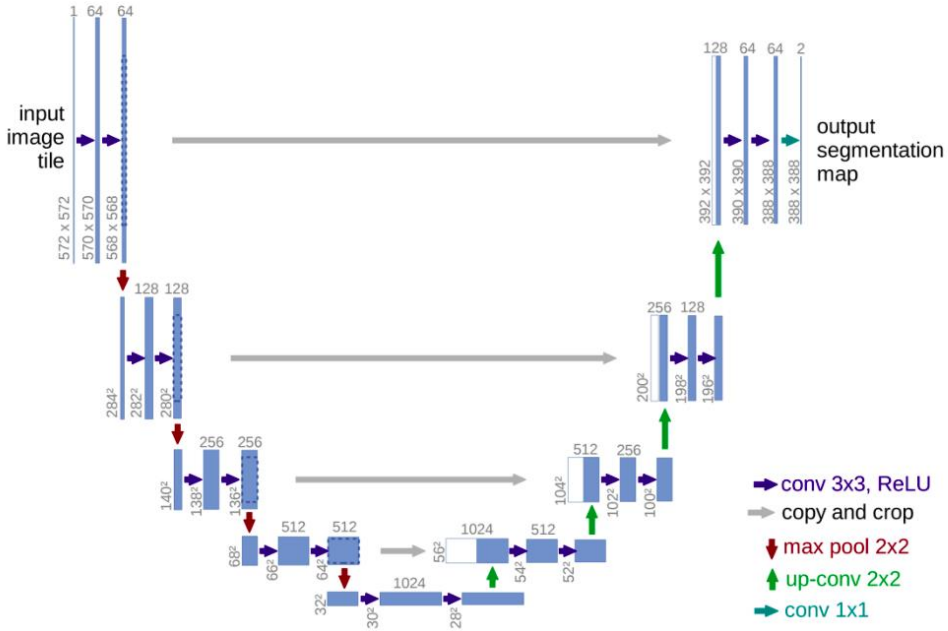
U-Net

Достоинства

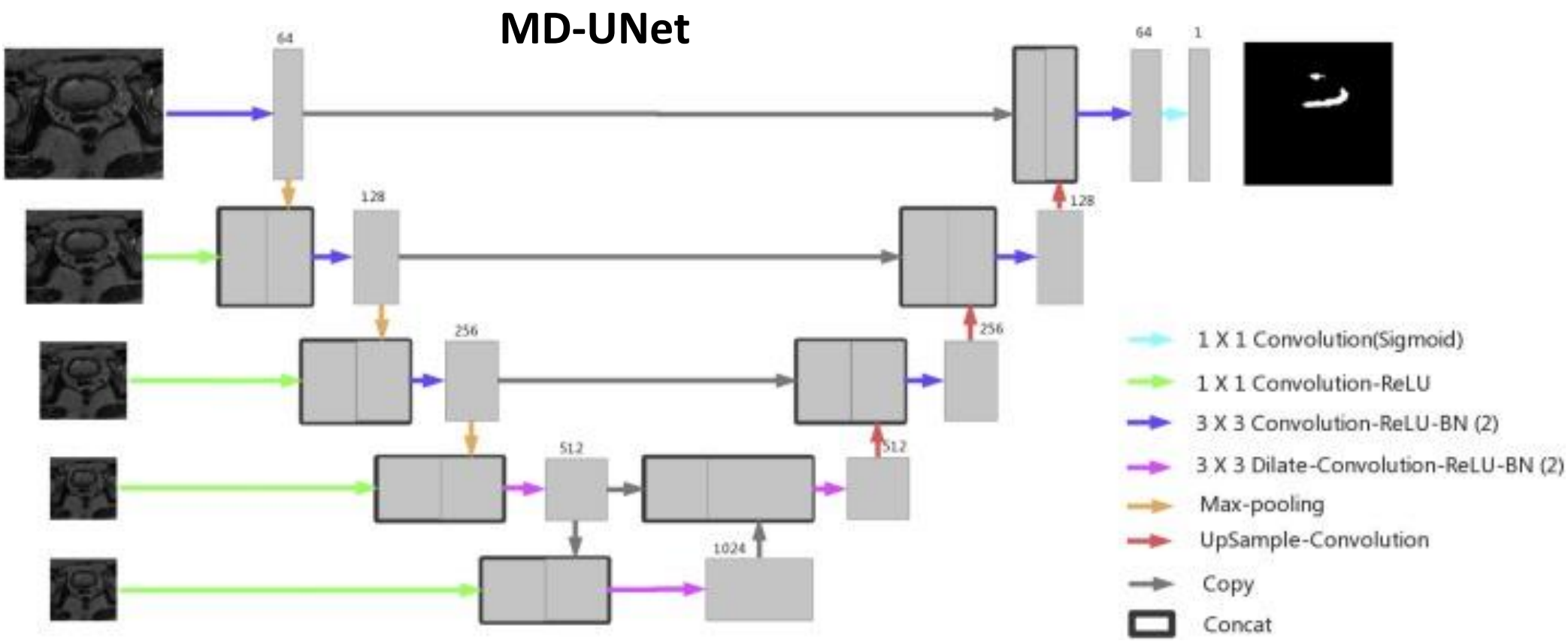
- Решается проблема с потерей пространственной информации
  за счет объединения с менее глубокими признаковыми представлениями.
- Можно достичь хорошего качества сегментации.

Недостатки

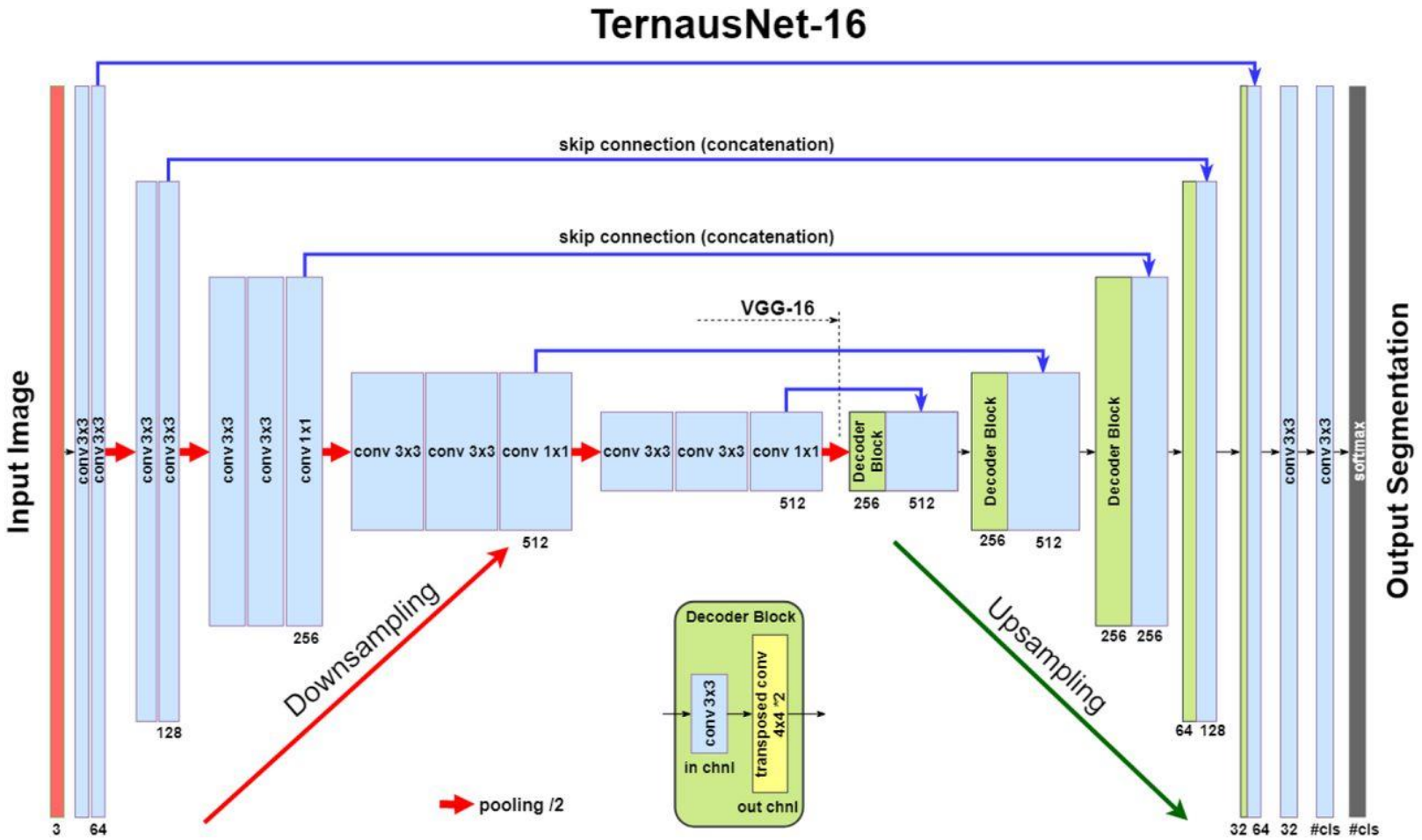
- Может переобучиться таким образом,
  что будут работать только верхние слои U - модели.
  Тогда модель в основном будет работать
  с локальными паттернами, но не будет изучать
  структуру данных в целом.



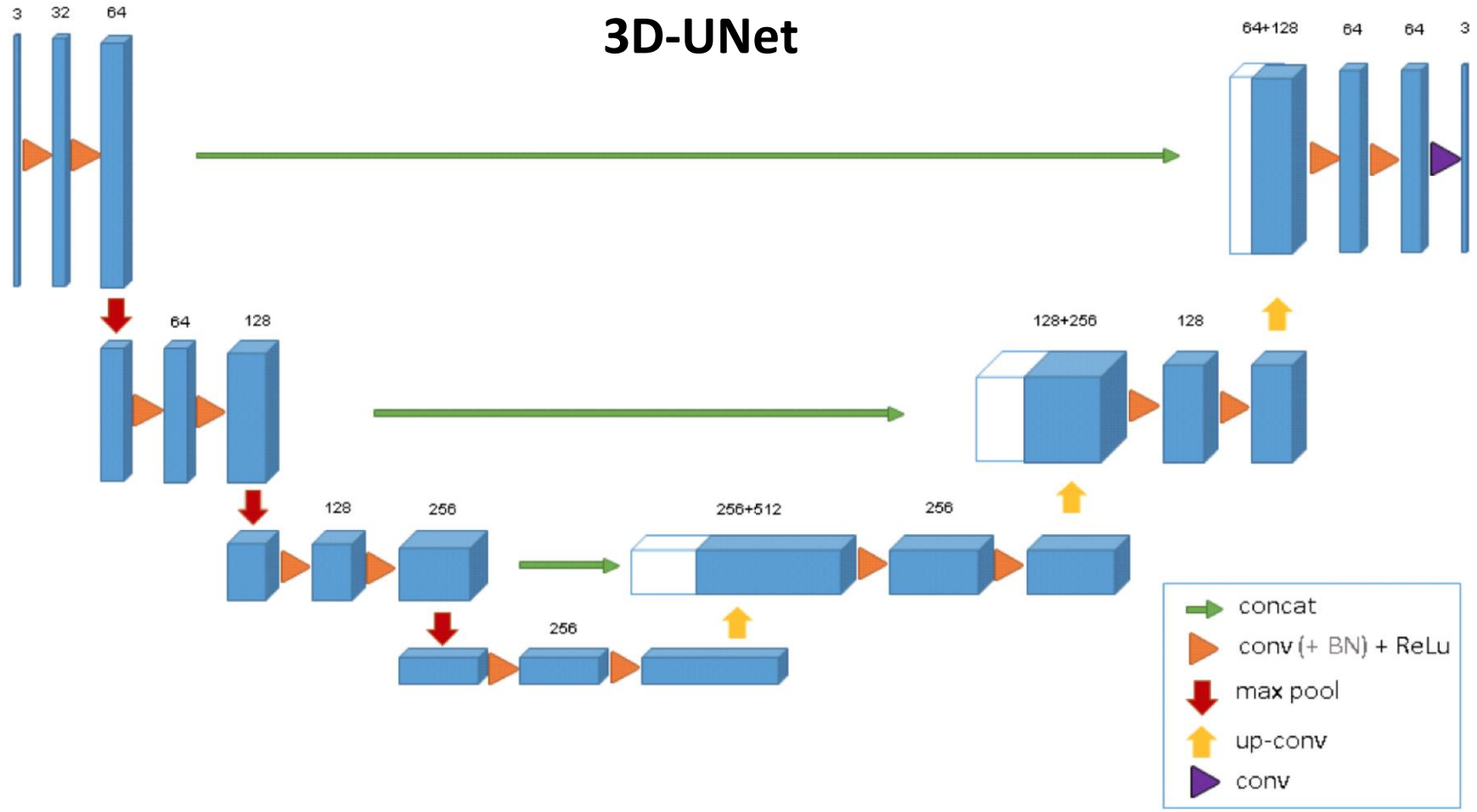
Модификации U-Net



Модификации U-Net



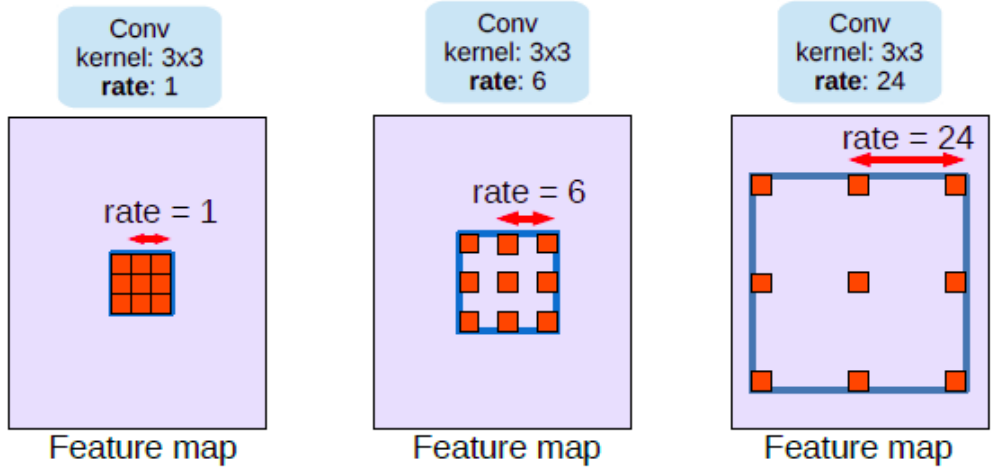
Модификации U-Net



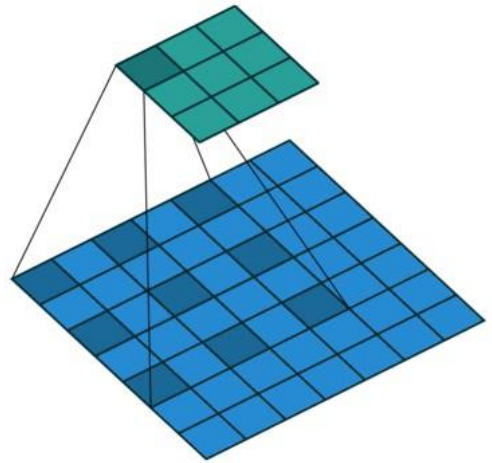
DeepLabV3: Atrous / Dillated Convolution

Выход y для входа x :

$$y[i] = \sum_{k}^{K} x[i + rk]w[k].$$



- Atrous свертка помогает **расширить рецертивное поле** фильтров, при этом **количество обучаемых параметров меньше**, чем если использовать для этого обычную свертку.



- Ее действие аналогично тому, что использовать обычную свертку, в которой вместо некоторых параметров стоят **нули**.

- В Pytorch из Conv2d в Pytorch можно получить Atrous свертку, установив параметр dilation > 1.

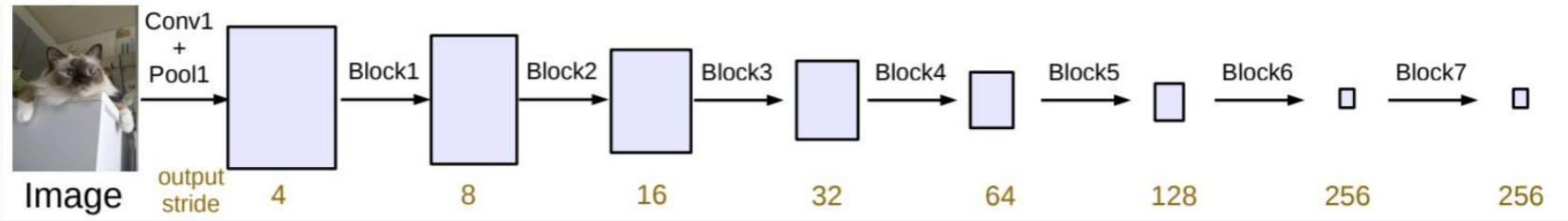
DeepLabV3: Atrous / Dillated Convolution

Идем глубже с Atrous сверткой

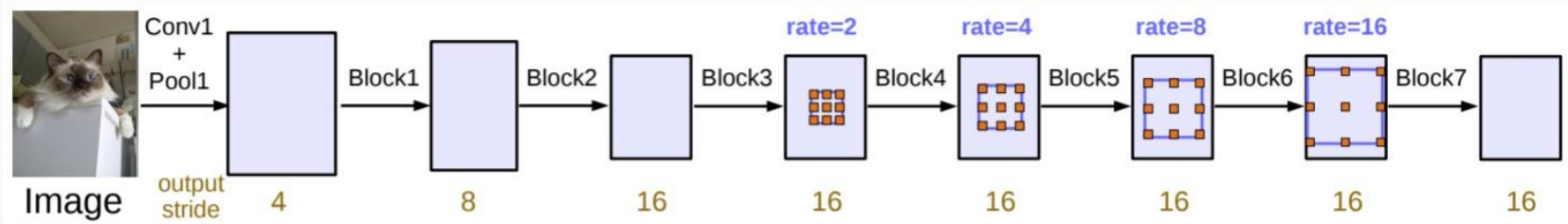
Без использования Atrous свертки каждый следующий **слой становится меньше** по размеру, из-за использования сверток со stride > 1 и пулингов.

Причем **stride** тут был необходим **для увеличения области видимости нейронов**.

При использовании Atrous свертки мы можем **сохранить stride и область видимости нейронов**, при этом **размер выхода не изменится**, что важно для семантической сегментации.



(a) Going deeper without atrous convolution.



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

DeepLabV3: ASSP

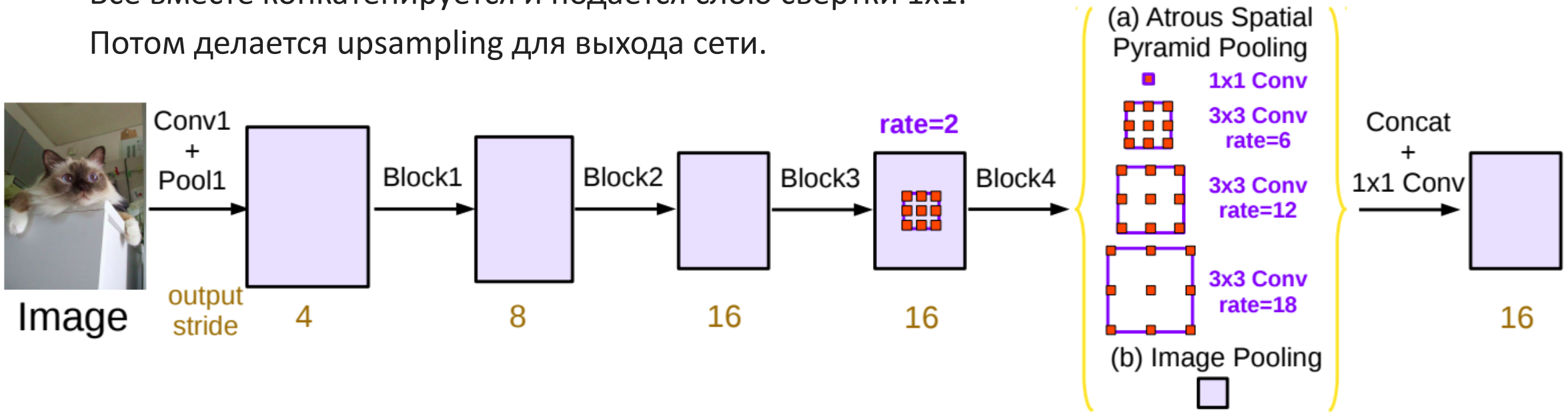
Atrous Spatial Pyramid Pooling (ASPP)

ASPP состоит из следующих компонент, которые вычисляются параллельно.

- одна 1x1 свертка сохраняет контекст — с 256 фильтрами + батч нормализация.
- три свертки с rate равным 6, 12, 18 соответственно и stride = 16, все с 256 фильтрами + батч нормализация — учитывают разного размера контекст.
- Image Pooling / Global average Pooling — учитывает глобальный контекст.

Все вместе конкатенируется и подается слою свертки 1x1.

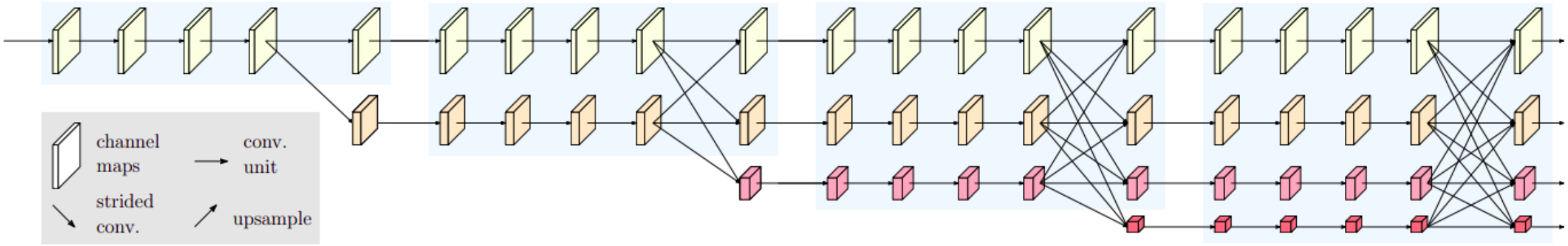
Потом делается upsampling для выхода сети.



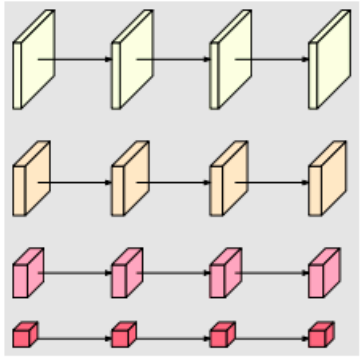
HR-Net

Одна из относительно недавних моделей сегментации (2020).
Дает предсказания с высокой точностью.

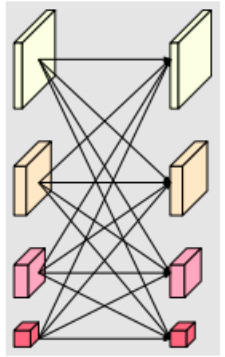
Пример HR-Net — основная часть сети



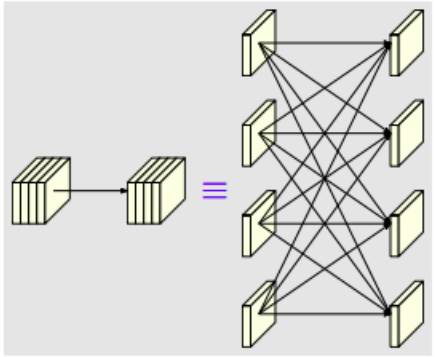
Параллельные свертки для разных разрешений



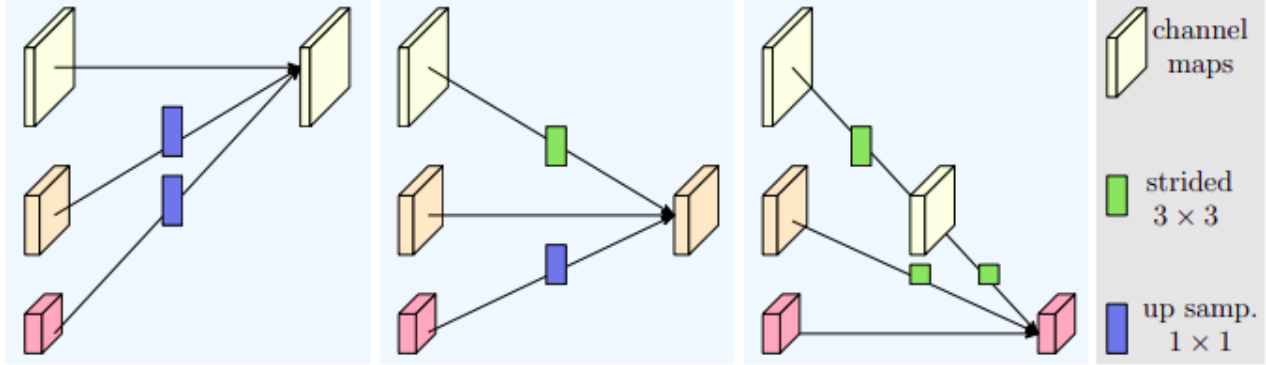
Fusion модуль для разных разрешений



Fusion модель для одинаковых разрешений эквивалентна свертке



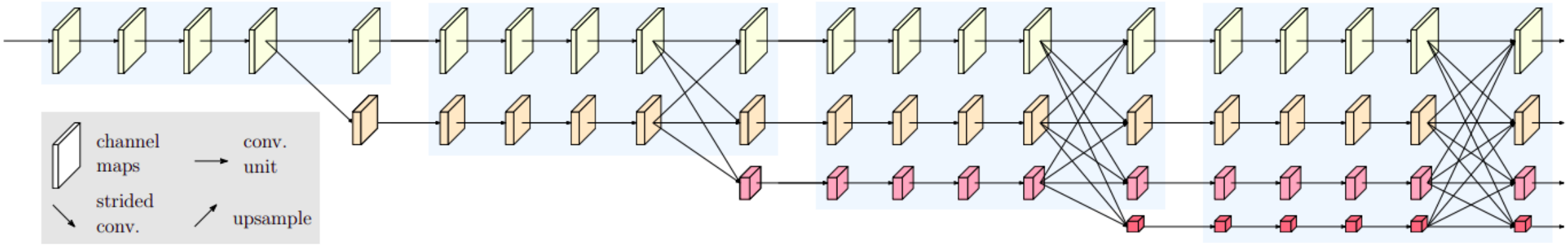
Fusion-модель — подробнее



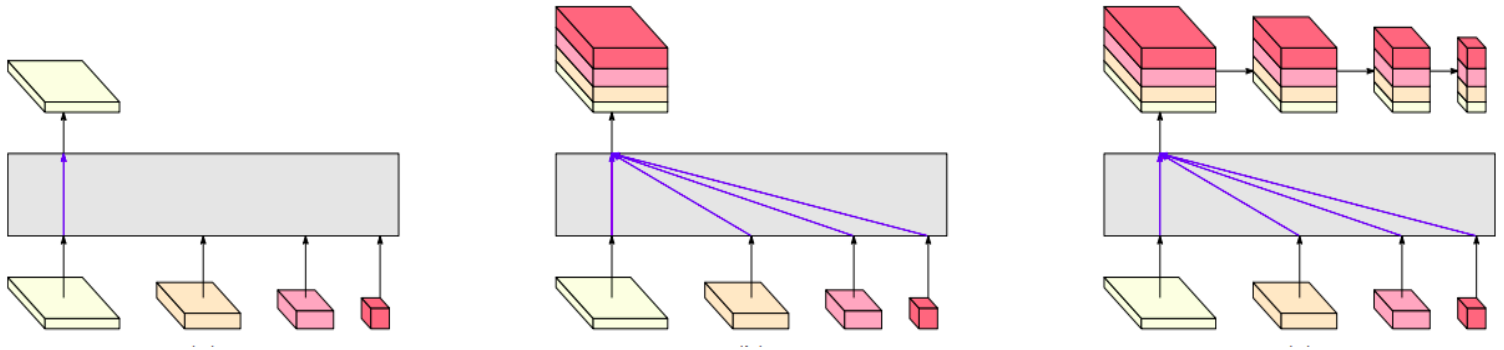
HR-Net

Одна из относительно недавних моделей сегментации (2020).
Дает предсказания с высокой точностью.

Пример HR-Net — основная часть сети



Пример HR-Net — выходная часть сети, 3 варианта



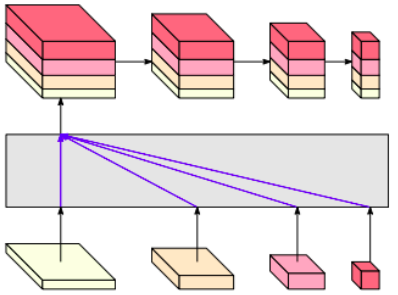
HR-Net

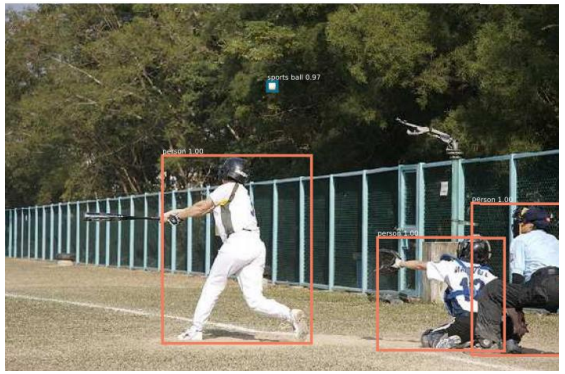
Чем примечатьна статья HR-Net

- Предложили архитектуру с параллельными свертками и fusion модулями.
- Получили хорошее качество сегментации.
- Применили модель для других задач: оценка точек позы человека и детекция объектов.

Оценка позы
Сначала детектировали объект, затем вырезали область с объектом
и предсказывали карты вероятностей нахождения определенных точек тела.
Карт столько же, сколько точек задается на человека задается в задаче.

Детекция объектов
Использовали выходы сети
в виде feature-piramid как вход
для модели детекции.





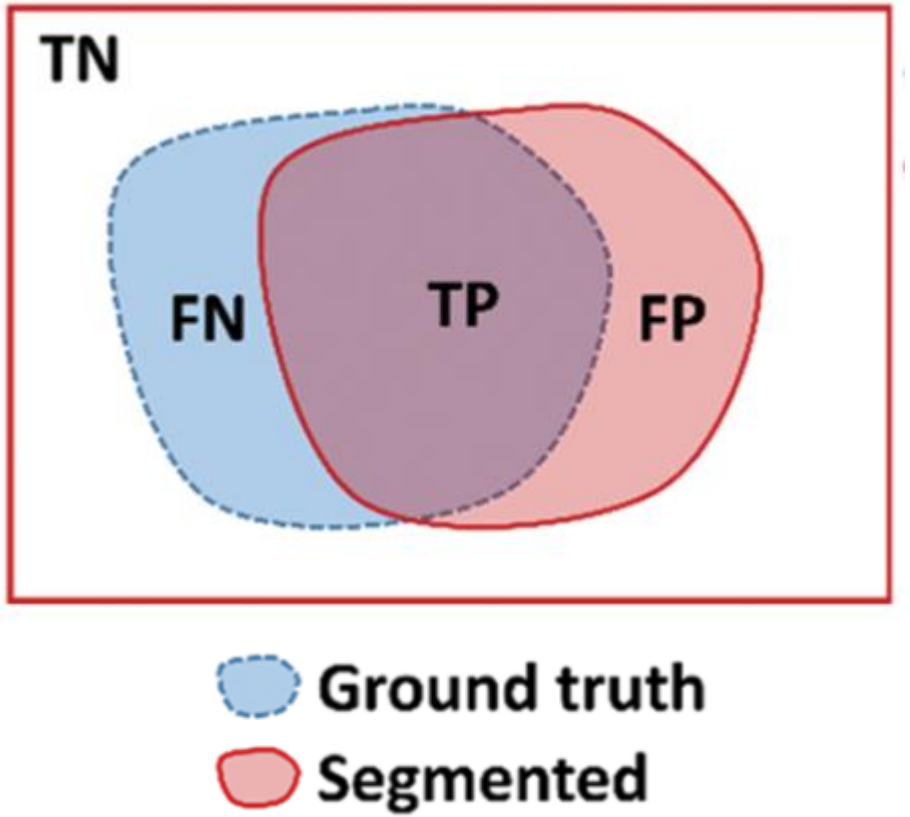
Метрики

На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

Pixel Accuracy

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Считаем долю пикселей,
для которых сделали
правильное предсказание.



Недостатки

Плохо в случае дисбаланса классов.
Например, если пикселей заданного класса мало,
то $TP + FN \ll TN$ и метрика всегда близка к 1,
даже если модель неправильно сработала.

Метрики

На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

Pixel Precision

$$\frac{TP}{TP + FP}$$

Считаем долю предсказ. пикселей целевого класса, для которых предсказ. правильное .

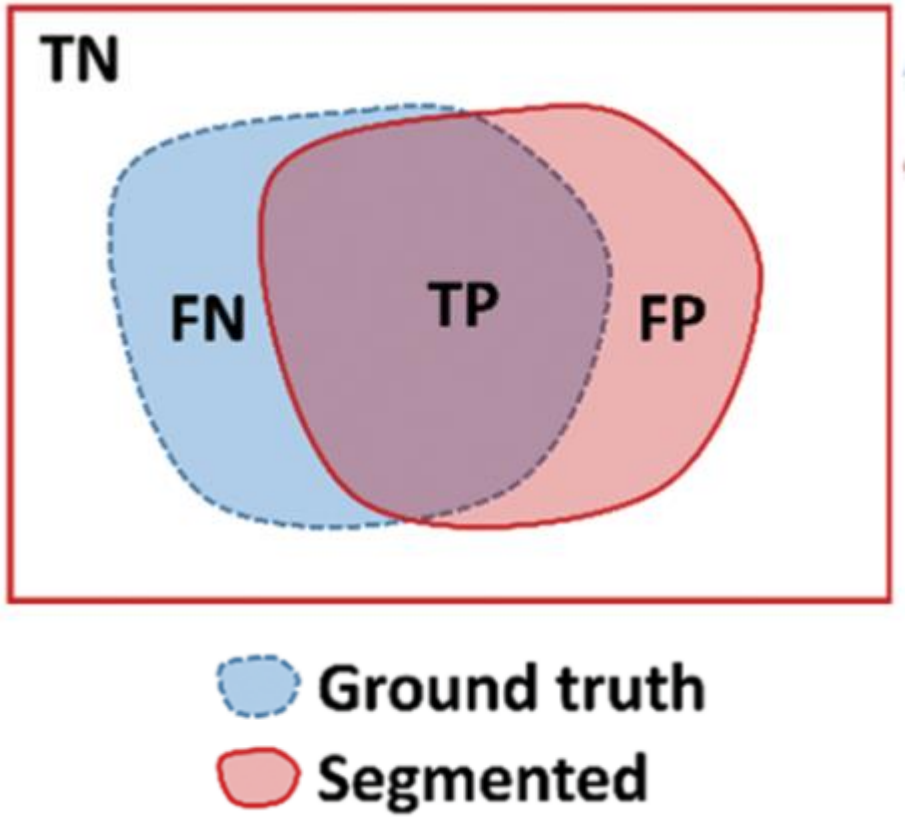
Недостатки
Чем меньше предсказ. пикселей целевого класса, тем больше метрика.

Pixel Recall

$$\frac{TP}{TP + FN}$$

Считаем долю пикселей целевого класса, для которых сделали правильное предсказание.

Недостатки
Чем больше предсказ. пикселей целевого класса, тем больше метрика.



Метрики

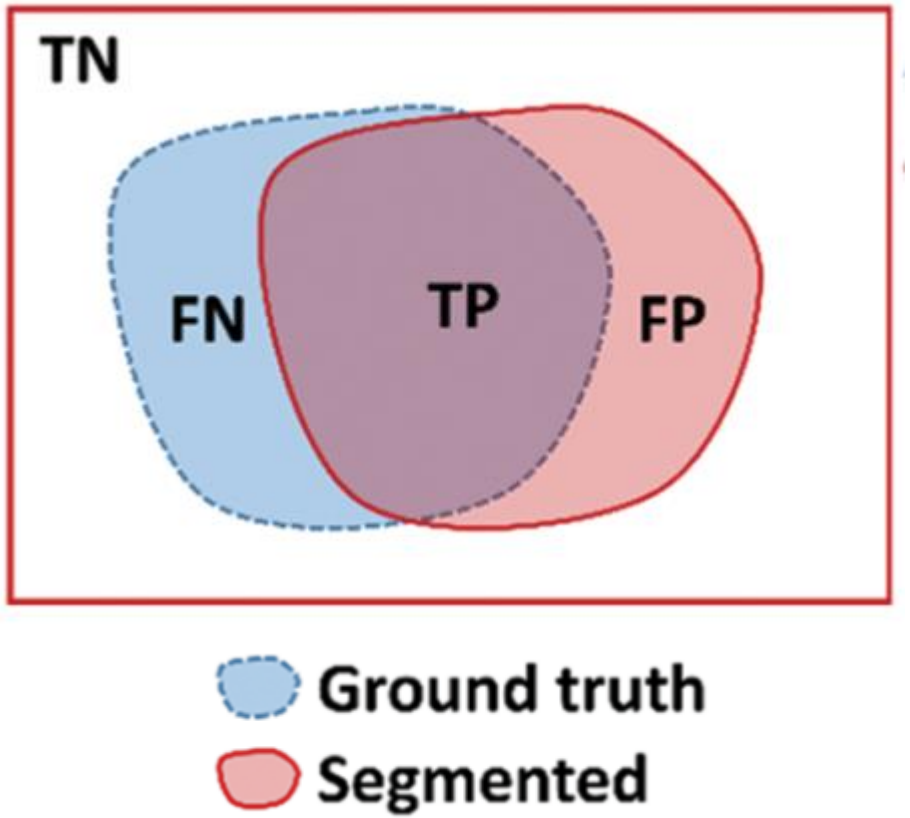
На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

Pixel F1-score

$$\frac{2\,precison\ *\ recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

Среднее гармоническое между precision и recall.
Нивелируем недостатки обеих метрик.

Недостатки
Не учитывается TN.



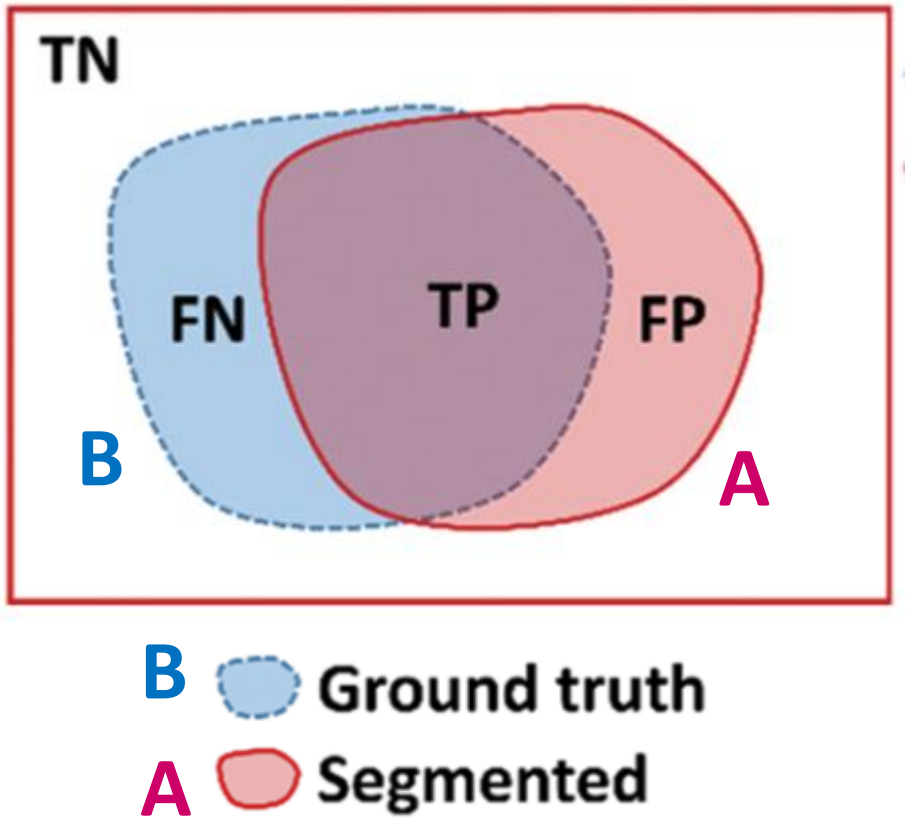
Метрики

На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

Pixel F1-score = Dice score

$$F_1 = \frac{2TP}{2TP + FP + FN} = \frac{2|A \cap B|}{|A| + |B|} = Dice$$

Геометрически смысл:
Считаем отношение пересечением областей предсказания и таргета к сумме областей предсказания и таргета.



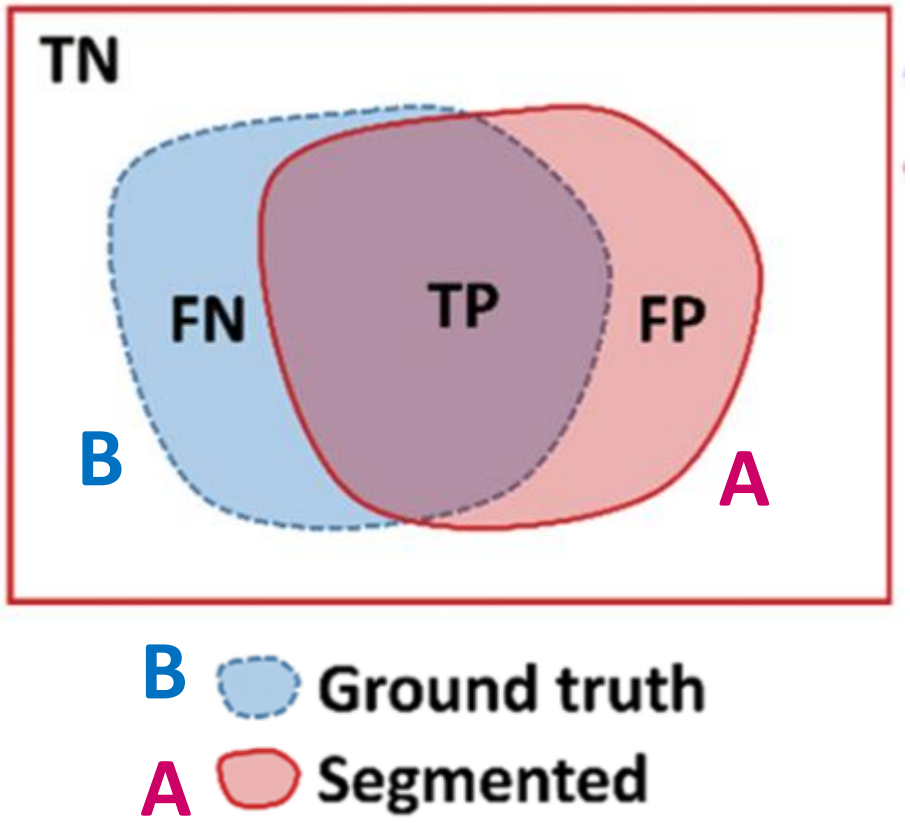
Метрики

На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

IoU – Intersection over Union /
Jacard score

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN}$$

Геометрически смысл:
считаем отношение пересечением областей предсказания и таргета к объедигнению предсказания и таргета.



Метрики

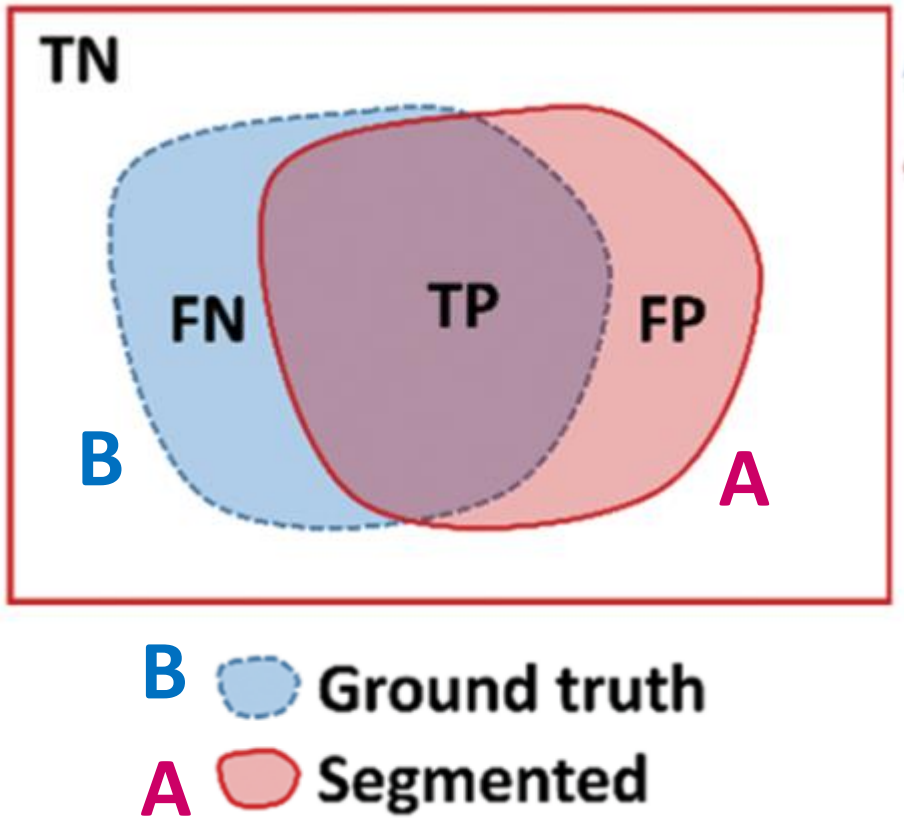
На выходе модели имеем предсказание класса для каждого пикселя изображения.
Рассмотрим предсказания для **одного класса**.

IoU – Intersection over Union /
Jacard score

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN}$$

$$Dice = \frac{2IoU}{1 + IoU}$$

Если IoU близка к 1, то и Dice близок к 1.
Если IoU близка к 0, то Dice почти в 2 раза больше.



Функции потерь

Рассмотрим модель сегментации для **одного класса**.
На выходе модели для каждого i-го пикселя изображения имеем предсказание вероятности класса $\hat{p}_i$, которому соответствует метка класса y_i (0 или 1).

Бинарная кросс-энтропия

$$L_{BCE}(y, \hat{p}) = -\sum_i \left(y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\right)$$

Взвешенная бинарная кросс-энтропия

$$L_{W-BCE}(y, \hat{p}) = -\sum_i \left(\beta y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\right)$$

Если $\beta < 0$, то уменьшаем количество FP, если $\beta > 0$, то уменьшаем количество FN.

Функции потерь

Рассмотрим модель сегментации для $\boldsymbol{K}$ **классов**.
На выходе модели для каждого i-го пикселя изображения имеем для класса k предсказание вероятности $\hat{p}_{ki}$, которому соответствует метка класса y_{ki} (0 или 1).

Кросс-энтропия

$$L_{CE}(y_{ki}, \hat{p}_{ik}) = \begin{cases} -\log \hat{p}_{ik}\,, & \text{если } y_{ki} = 1 \\ -\log(\,1 - \hat{p}_{ik}), & \text{иначе} \end{cases}$$

Положим

$$q_{ki} = \begin{cases} \hat{p}_{ik}, & \text{если } y_{ki} = 1 \\ 1 - \hat{p}_{ik}, & \text{иначе} \end{cases}$$

тогда

$$L_{CE}(y_{ki}, \hat{p}_{ik}) = L_{CE}(q_{ik}) = -\log q_{ik}$$

Функции потерь

На выходе модели имеем предсказание **вероятности класса** p для каждого пикселя изображения, которому соответствует действительное значение класса y .
Рассмотрим предсказания для **нескольких непересекающихся классов**.

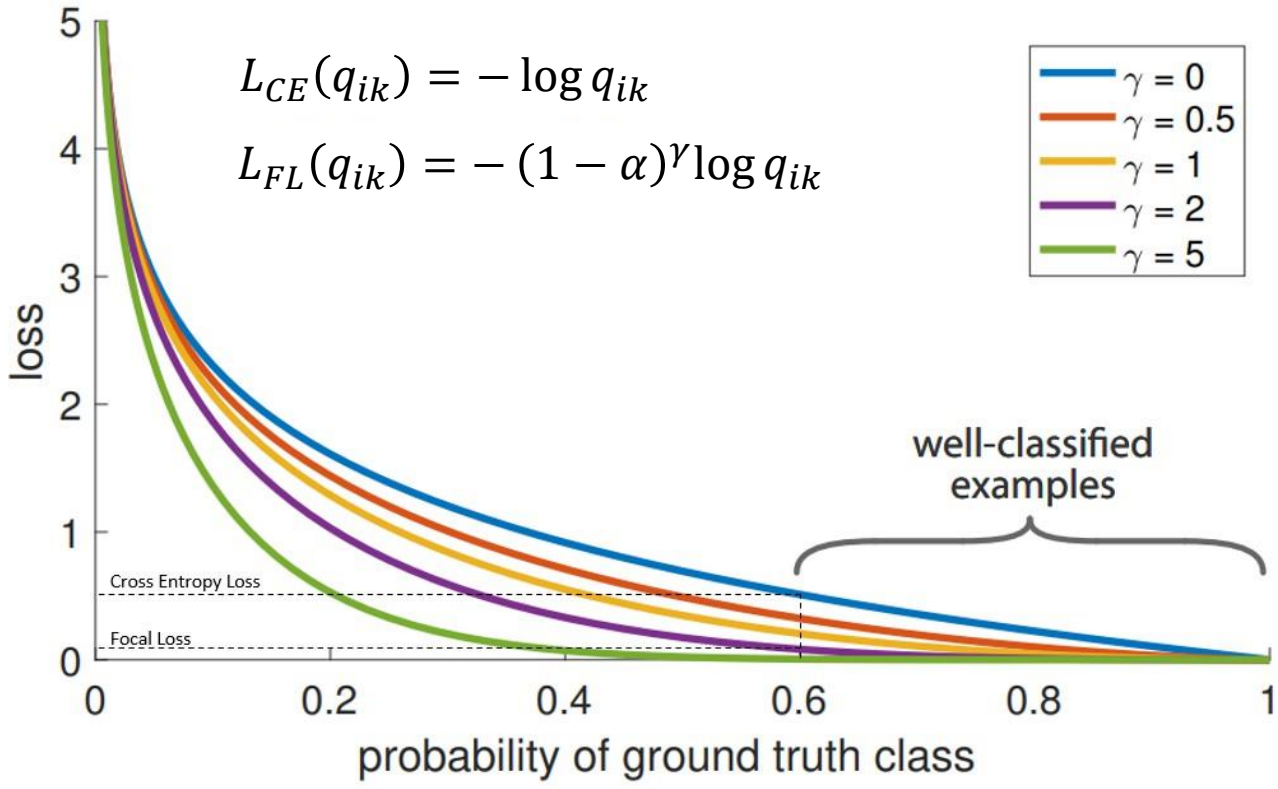
Focal Loss

$$q_{ki} = \begin{cases} \hat{p}_{ik}, & \text{если } y_{ki} = 1 \\ 1 - \hat{p}_{ik}, & \text{иначе} \end{cases}$$

$$L_{FL}(q_{ik}) = -(1-\alpha)^{\gamma} \log q_{ik}$$

Чем больше γ , тем меньше модель штрафует хорошо размеченные классы.

Чем больше α, тем меньше модель штрафует объекты, где $y = 1$.



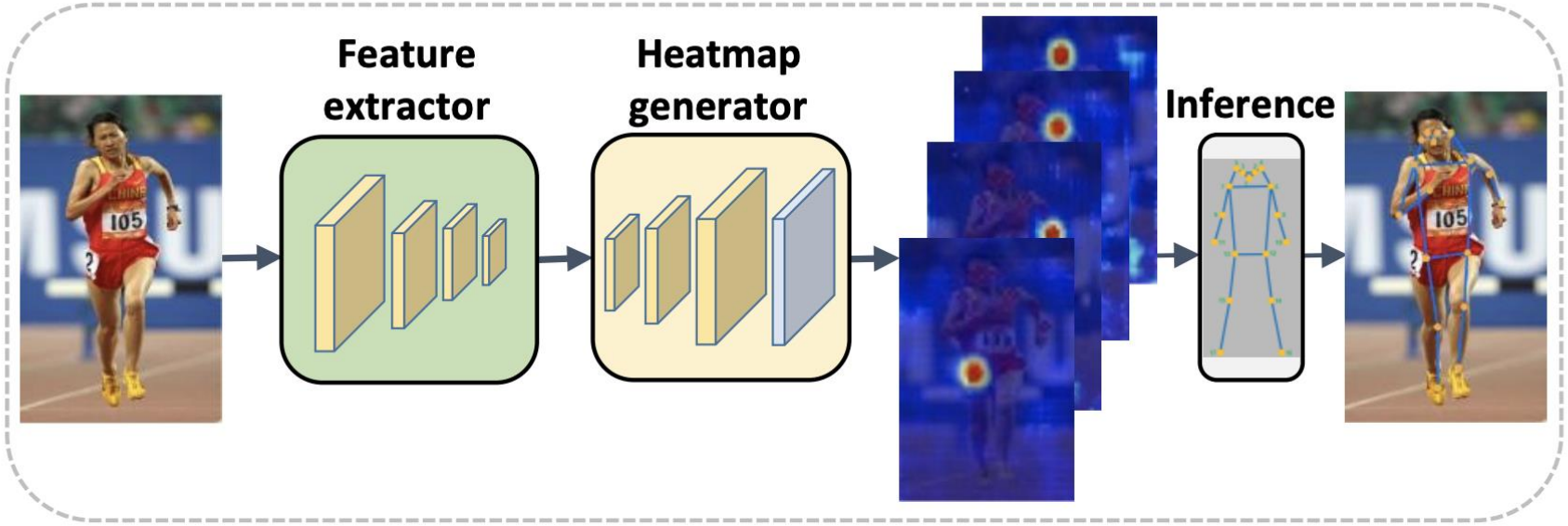
Функции потерь

Рассмотрим предсказания для **одного класса**.
На выходе модели имеем предсказание **вероятности класса** p_i для каждого пикселя i, которому соответствует метка класса y_i.
Пусть $\widehat{y_i}$ предсказание класса для пикселя i.

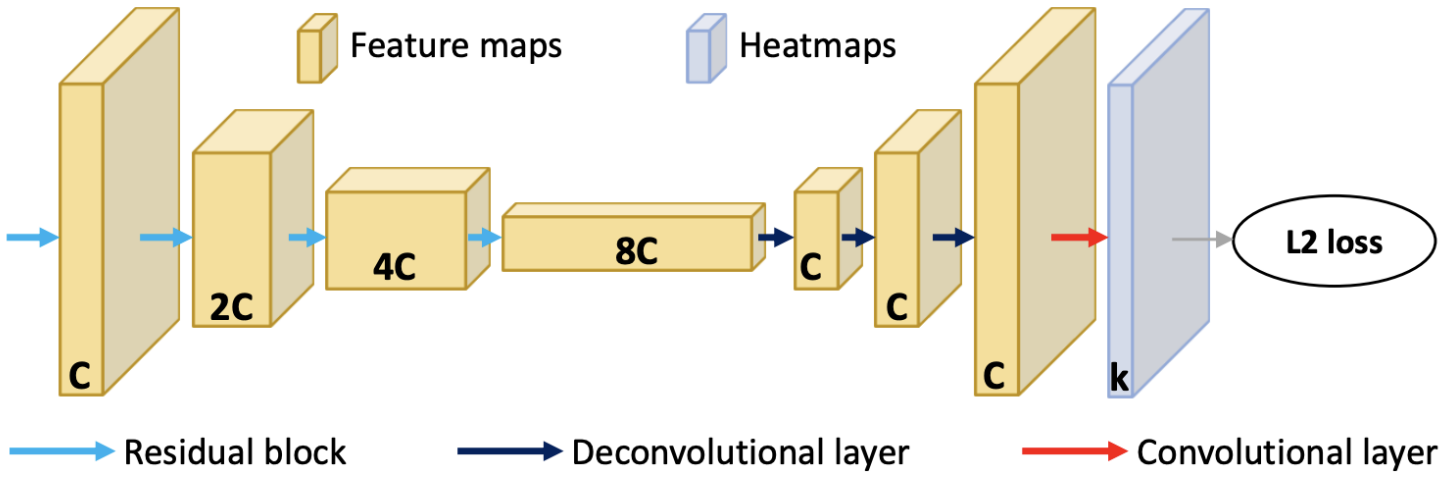
Dice Loss

$$Dice = \frac{2|A \cap B|}{|A| + |B|} \longrightarrow Dice = \frac{2\sum \widehat{y_i} y_i}{\sum \widehat{y_i} + \sum y_i} \longrightarrow L_{Dice} = 1 - \frac{2\sum p_i y_i + \varepsilon}{\sum p_i + \sum y_i + \varepsilon}$$

Предсказание позы
как регрессия тепловой карты

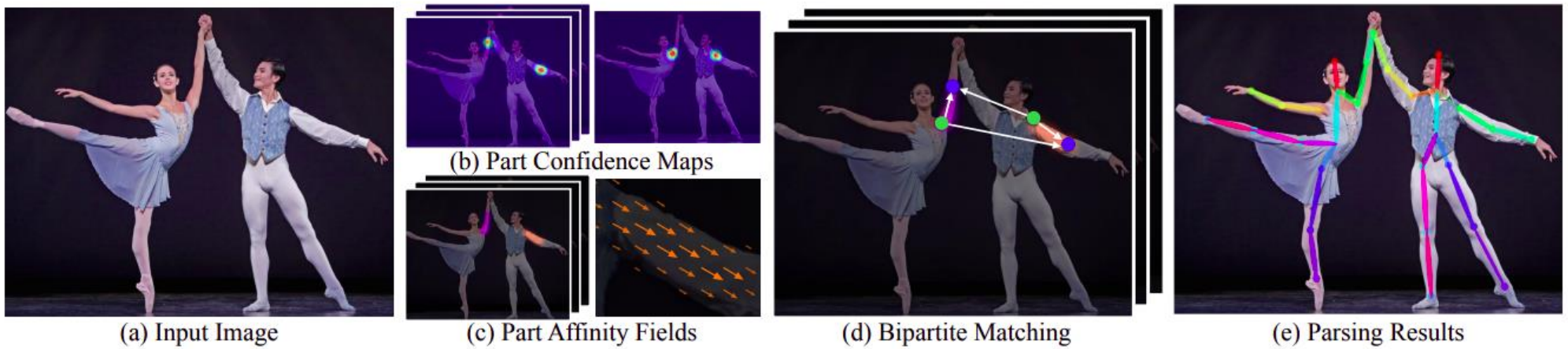


Предсказание позы
как регрессия тепловой карты

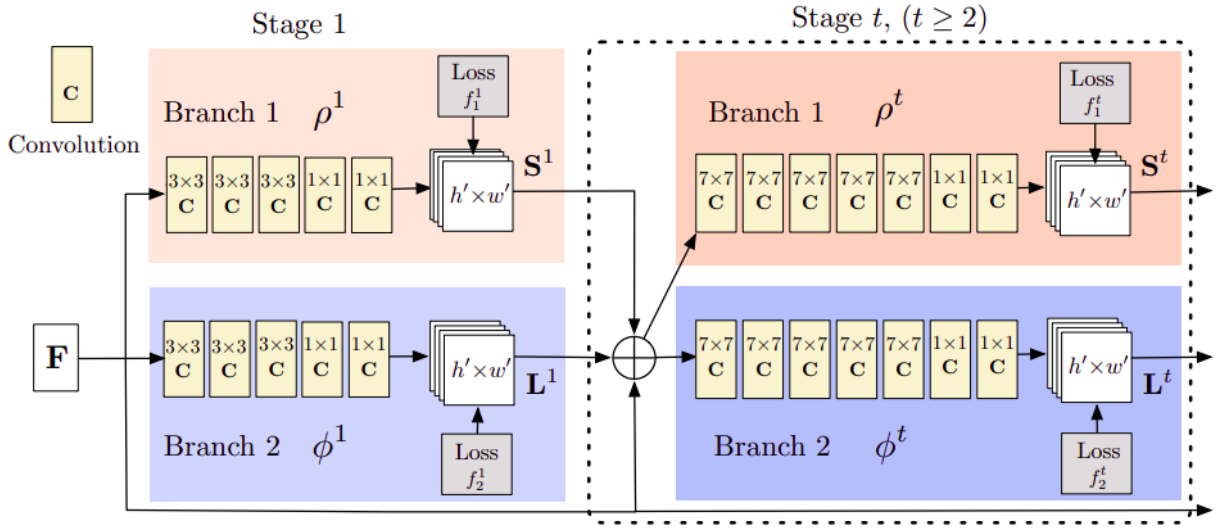


Статья

Предсказание позы. Open Pose



2 ветви сети предсказывают карту веротяности точек позы и афинное поле, по которому восстнавливается взаимосвзяь между точками позы.



[Статья]

ВСЁ!