

# Сегментация изображений

Виктор Китов

[v.v.kitov@yandex.ru](mailto:v.v.kitov@yandex.ru)



# Содержание

## 1 Введение

- Постановка задачи
- Меры качества
- Подходы решения задачи
- Повышение пространственного разрешения

## 2 Нейросетевые архитектуры

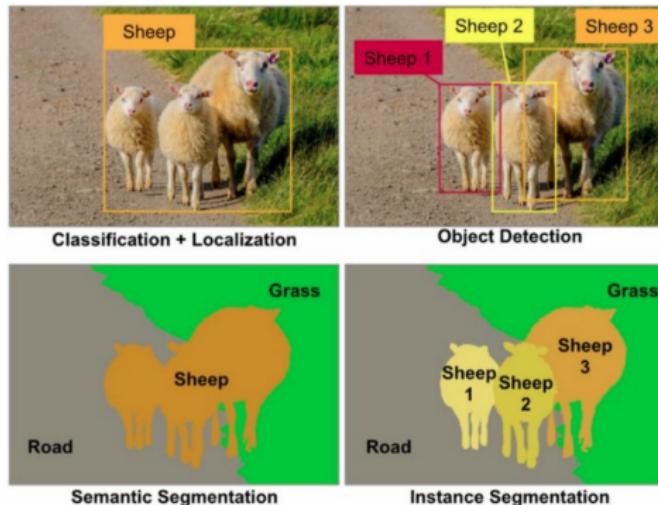
## 3 Instance и panoptic сегментация

1

## Введение

- Постановка задачи
- Меры качества
- Подходы решения задачи
- Повышение пространственного разрешения

# Типы задач



- Классификацию+локализацию можно выполнить, добавив к классификатору регрессионный вывод ( $x, y, h, w$ ).
- Instance сегментацию можно решить, используя object detection+semantic segmentation.

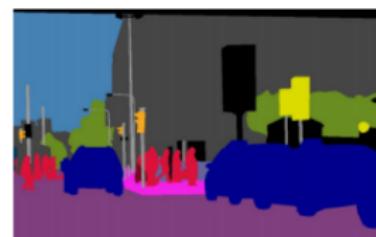
## Panoptic segmentation: instance seg+сегментация фона



Image



Instance segmentation



Semantic segmentation



Panoptic segmentation

# Применения

- Беспилотные системы:
  - автоуправляемые машины: сегментировать людей, машины, знаки, дорожные препятствия.
  - мониторинговые роботы: определить грузы на складе, их расположение и количество.
- Изображения со спутника:
  - определить городские и сельскохозяйственные районы, дороги, транспортные средства, ход стройки.
  - сегментировать поля с разными видами растений, их рост, найти лесные пожары, области загрязнений.
- Медицина:
  - сегментировать кости, ткани, заболевания.
- Понимание сцены и событий на ней, замена фона.

Выход задачи сегментации<sup>1</sup>



## Input

## Segmented

- 1: Person
  - 2: Bench
  - 3: Plant/Grass
  - 4: Cat

## Semantic Labels

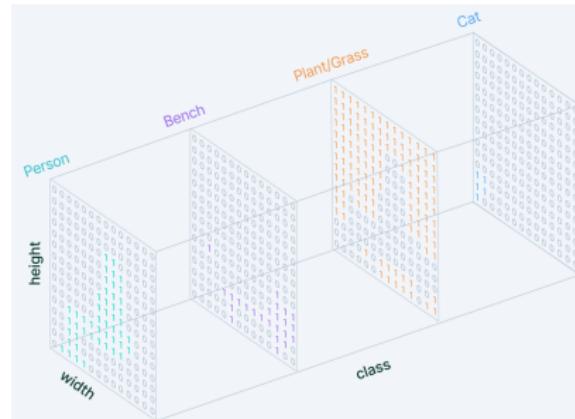
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	1	1	3	3	3	3
3	3	3	3	3	3	3	1	1	1	3	3	3	3
3	3	3	3	3	3	3	1	1	1	3	3	3	3
3	3	3	3	3	3	3	1	1	1	3	3	3	3
3	3	3	3	3	3	3	1	1	1	1	3	3	3
3	3	3	3	3	3	3	1	1	1	1	3	3	3
3	3	2	3	3	3	1	1	1	1	1	2	3	3
3	3	1	1	1	1	1	1	1	1	1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

<sup>1</sup><https://www.v7labs.com/blog/semantic-segmentation-guide>

## Выход задачи сегментации

Для каждого пикселя  $(i, j)$ :  $y$  - one-hot класс,  $\hat{y}$  - вероятности классов. Настройка модели ( $\uparrow$  логарифма правдоподобия  $\Leftrightarrow \downarrow$  кросс-энтропии):

$$\sum_{n=1}^N \sum_{c=1}^C \mathbb{I}[y_n = c] \ln p_\theta(y = c|x_n) \rightarrow \max_{\theta}$$



1

## Введение

- Постановка задачи
- **Меры качества**
- Подходы решения задачи
- Повышение пространственного разрешения

## Меры качества: один класс

- $Y$  - истинное выделение класса,  $\hat{Y}$  - предсказанное выделение.
- Точность (accuracy) - завышает качество, когда объект мал ( $|(\neg Y) \cap (\neg \hat{Y})| \gg 1$ )

$$\frac{|Y \cap \hat{Y}| + |(\neg Y) \cap (\neg \hat{Y})|}{W \cdot H}$$

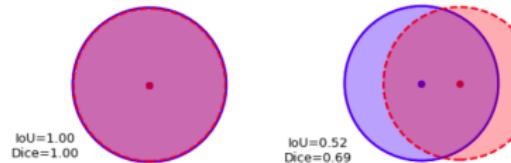
## Меры качества: один класс<sup>3</sup>

- Мера intersection-over-union (IoU)=близость Жаккарда:

$$\text{IoU} = \text{Jaccard} = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cup Y|} = \frac{TP}{TP + FP + FN} \in [0, 1]$$

- Мера Dice=F-мера<sup>2</sup>

$$\text{Dice} = \frac{2 |\hat{Y} \cap Y|}{|\hat{Y}| + |Y|} = \frac{2 TP}{2 TP + FP + FN} = \frac{1}{\frac{1}{2} \frac{\hat{P}}{TP} + \frac{1}{2} \frac{P}{TP}} \in [0, 1]$$

IoU=1.00  
Dice=1.00IoU=0.52  
Dice=0.69

<sup>2</sup>Докажите.

<sup>3</sup><https://ilmonteux.github.io/2019/05/10/segmentation-metrics.html>

## Меры качества: один класс

- Меры связаны монотонно<sup>4</sup>:

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

- По сравнению с IoU мера Dice меньше штрафует несоответствия для в целом верно классифицированных объектов (с большим  $TP$ ):

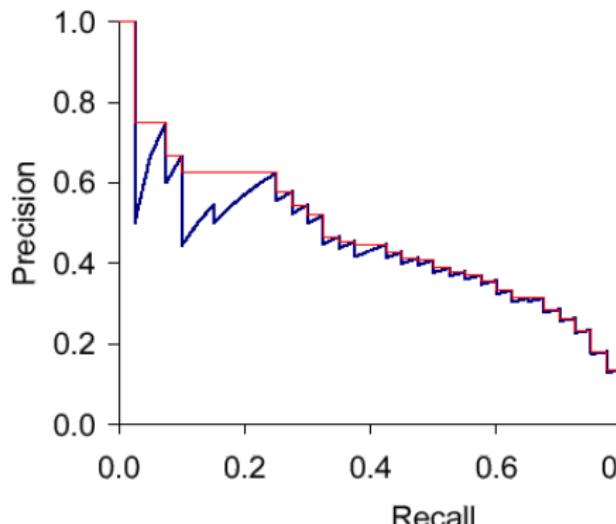
$$Dice = \frac{TP + TP}{TP + TP + FP + FN}$$

---

<sup>4</sup>Докажите.

## Меры качества: один класс

- Можно задать порог  $\alpha$ , с которого начинается распознавание класса, считать  $\text{Precision}(\alpha)$ ,  $\text{Recall}(\alpha)$ , построить  $\text{Prec}(\text{Recall})$  посчитать плотность под графиком (на практике он ещё сглаживается).



# Оптимизация Dice напрямую

- Проблема cross-entropy loss: мало объекта и много фона.
  - вариант решения: взвешивание по редкости классов.
- Другой вариант - оптимизировать меры качества напрямую.
- В архитектуре V-net<sup>5</sup>:
  - выходы после SoftMax  $p_i$ , бинарная истинная разметка  $g_i$ ,  $i$ -позиция.
- Модель оптимизируется по Dice:

$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \quad \frac{\partial D}{\partial p_j} = 2 \left[ \frac{g_j \left( \sum_i^N p_i^2 + \sum_i^N g_i^2 \right) - 2 p_j \left( \sum_i^N p_i g_i \right)}{\left( \sum_i^N p_i^2 + \sum_i^N g_i^2 \right)^2} \right]$$

<sup>5</sup><https://arxiv.org/pdf/1606.04797.pdf>

# Качество определения границ<sup>6</sup>

- Важный показатель-качество вокруг границ.
- Пусть  $B_r(Y)$  - полоса ширины  $r$  вокруг границы маски  $Y$ ,
- Trimap IoU: IoU в окрестности истинных границ  $Y$ :

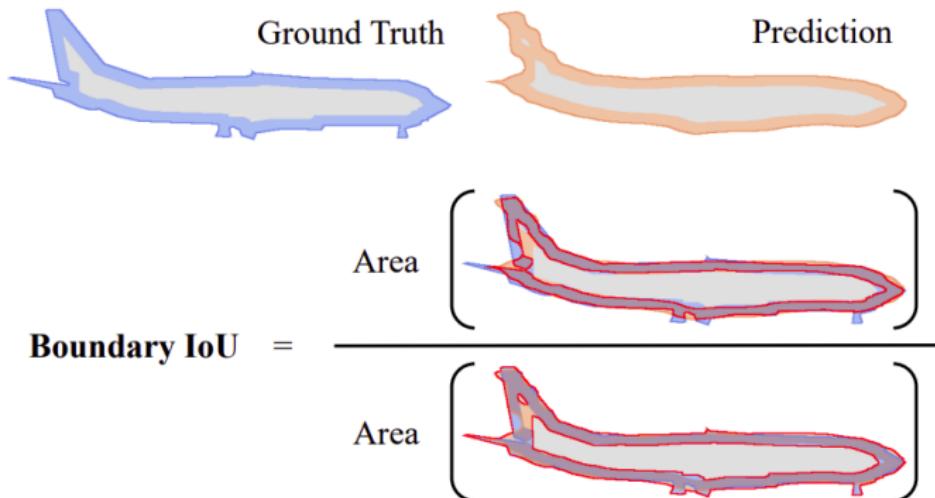
$$\text{Trimap IoU} = \frac{|B_r(Y) \cap \hat{Y} \cap Y|}{|(B_r(Y) \cap \hat{Y}) \cup (B_r(Y) \cap Y)|}$$

- недостатки: несимметрична, не смотрит ошибки границы  $\hat{Y}$  за пределами границ  $B_r(Y)$
- Boundary IoU: симметрична, смотрим пересечение прогноза на границе прогноза с фактом на границе факта.

$$\text{Boundary IoU} = \frac{|(B_r(\hat{Y}) \cap \hat{Y}) \cap (B_r(Y) \cap Y)|}{|(B_r(\hat{Y}) \cap \hat{Y}) \cup (B_r(Y) \cap Y)|}$$

<sup>6</sup><https://arxiv.org/pdf/2103.16562.pdf>

# Boundary IoU



## Меры качества: $C$ классов<sup>7</sup>

- Матрица ошибок (confusion matrix)  $M_{ij} = \#\{y = i \& \hat{y} = j\}$

$$G_i = \sum_j M_{ij} - \# \text{пикселей класса } i \text{ (ground truth)}$$

$$P_j = \sum_i M_{ij} - \# \text{прогнозов класса } j \text{ (predicted)}$$

- Overall pixel accuracy (OP) - микроусреднение на классах:

$$\frac{\sum_{i=1}^C M_{ii}}{\sum_{i=1}^C G_i} - \text{доминируется частыми классами}$$

- Per-Class (PC) accuracy - макроусреднение на классах:

$$\frac{1}{C} \sum_{i=1}^C \frac{M_{ii}}{G_i} - \text{выгоднее чаще предс-ть редкие классы}$$

---

<sup>7</sup><http://www.bmva.org/bmvc/2013/Papers/paper0032/paper0032.pdf>

# Меры качества: $C$ классов

- Индекс Жаккарда (Jaccard index) - макроусредненный intersection over union

$$\frac{1}{C} \sum_{i=1}^C \frac{M_{ii}}{G_i + P_i - M_{ii}} = \frac{1}{C} \sum_{i=1}^C \frac{|\hat{y} = i \text{ and } y = i|}{|\hat{y} = i \text{ or } y = i|}$$

- Macro-averaged Dice:

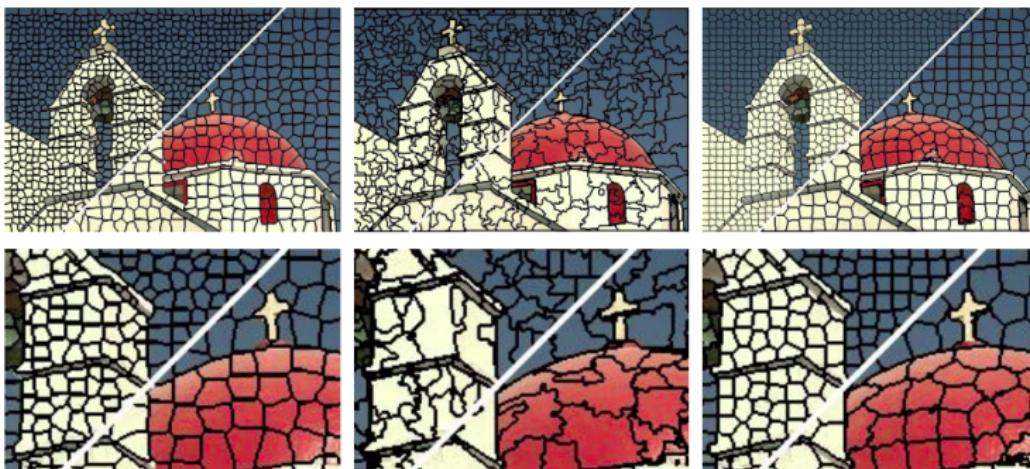
$$\frac{1}{C} \sum_{i=1}^C \frac{2M_{ii}}{G_i + P_i} = \frac{1}{C} \sum_{i=1}^C \frac{2 |\hat{y} = i \text{ and } y = i|}{|y = i| + |\hat{y} = i|}$$

## 1 Введение

- Постановка задачи
- Меры качества
- **Подходы решения задачи**
- Повышение пространственного разрешения

## SLIC<sup>8</sup>: сегментация, основанная на суперпиксели

- Алгоритм SLIC позволяет разбить изображение на суперпиксели (близкие блоки с примерно похожими цветами), кластеризуя в  $(x, y, color)$  пространстве.



<sup>8</sup>[https://www.iro.umontreal.ca/~mignotte/IFT6150/Articles/SLIC\\_Superpixels.pdf](https://www.iro.umontreal.ca/~mignotte/IFT6150/Articles/SLIC_Superpixels.pdf)

# Алгоритм SLIC

1. Изображение переводится в цветовое пространство CIELAB.
  - в котором Евклидово расстояние  $\approx$  воспринимаемой цветовой разнице.
2. Инициализируются центры  $K$  кластеров по равномерной сетке  $\{(x_0^k, y_0^k)\}_{k=1}^K$ .
  - если  $N = \#\text{пикселей}$ , то  $N/K = \#\text{пикселей в кластере}$ , сторона  $S \sim \sqrt{N/K}$ .
3. Центры смещаются, чтобы обеспечить минимум перепада цветов вдоль вертикальной и горизонтальной оси в окрестности 3x3:

$$\|I(x+1, y) - I(x-1, y)\|_2^2 + \|I(x, y-1) - I(x, y+1)\|_2^2$$

$$\rightarrow \min_{x, y \in \Omega(x_0^k, y_0^k)}$$

## Алгоритм SLIC

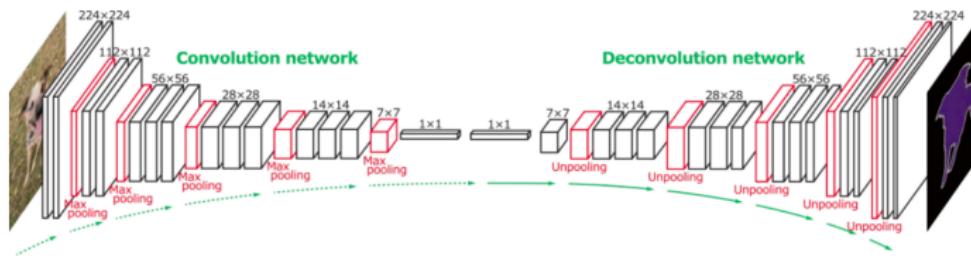
### 4. В цикле до сходимости:

- ① для каждого центроида производится распределение окружающих его пикселей между центроидами в  $(l, a, b, x, y)$  в окрестности  $(\pm 2S, \pm 2S)$
- ② обновляются расположения центроидов кластеров

### 5. Постобработка: если обнаружены несвязные области, отнесенные к одному центроиду, они присоединяются к ближайшему соседнему кластеру.

# Базовая нейросетевая архитектура сегментации

- Поскольку  $\hat{Y} \in \mathbb{R}^{H \times W}$ , используется следующая архитектура:



- Кодировщик: свёртки извлекают все более сложные признаки
  - разрешение  $\downarrow$  за счёт свёрток и пулингов
  - можно инициализировать первыми слоями CNN
    - потом можно донастроить под сегментацию
- Декодировщик постепенно  $\uparrow$  разрешение.

## Базовая нейросетевая архитектура сегментации

- Это полносвёрточная архитектура (fully convolutional architecture)
- Архитектура применима к изображениям произвольных размеров с любым соотношением сторон
  - но веса свёрток обучаются на (и далее ожидают) некоторое привычное разрешение
- Нужен способ  $\uparrow$  пространственное разрешение.

## 1 Введение

- Постановка задачи
- Меры качества
- Подходы решения задачи
- Повышение пространственного разрешения

# Повышение пространственного разрешения (upsampling)

↑ пространственного разрешения (upsampling): расширение входа, затем обычная свертка.

- заполнение нулями (bed of nails):

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & b & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- перемасштабирование ближайшим соседом:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow \begin{pmatrix} a & a & b & b \\ a & a & b & b \\ c & c & d & d \\ c & c & d & d \end{pmatrix}$$

- более точно: билинейная/бикубическая интерполяция

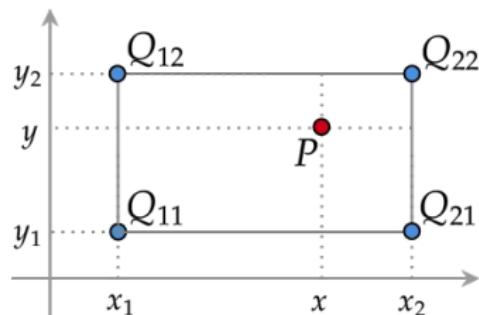
# Билинейная интерполяция

$$P(y_1) = (1 - \alpha) Q_{11} + \alpha Q_{21}, \quad P(y_2) = (1 - \alpha) Q_{12} + \alpha Q_{22}$$

$$\alpha = \frac{x - x_1}{x_2 - x_1}$$

$$P = (1 - \beta) P(y_1) + \beta P(y_2)$$

$$\beta = \frac{y - y_1}{y_2 - y_1}$$

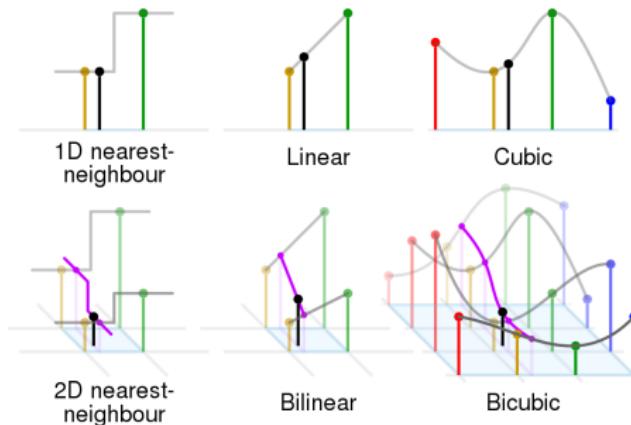


Bilinear

# Другие виды интерполяции

бикубическая интерполяция:  $p(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$

$p(x_i, y_j)$ -известны, производные на углах соседних квадратов равны



# Транспонированная свёртка

Обычная свёртка:  $H \times W \rightarrow h \times w$  и представима в виде матричного произведения:

$$\begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \circledast \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha A + \beta B + \gamma D + \delta E & \alpha B + \beta C + \gamma E + \delta F \\ \alpha D + \beta E + \gamma G + \delta H & \alpha E + \beta F + \gamma H + \delta I \end{pmatrix}$$

$$= \text{reshape} \left\{ \underbrace{\begin{pmatrix} \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta & 0 \\ 0 & 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta \end{pmatrix}}_P \right\} \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{pmatrix}$$

# Транспонированная свёртка

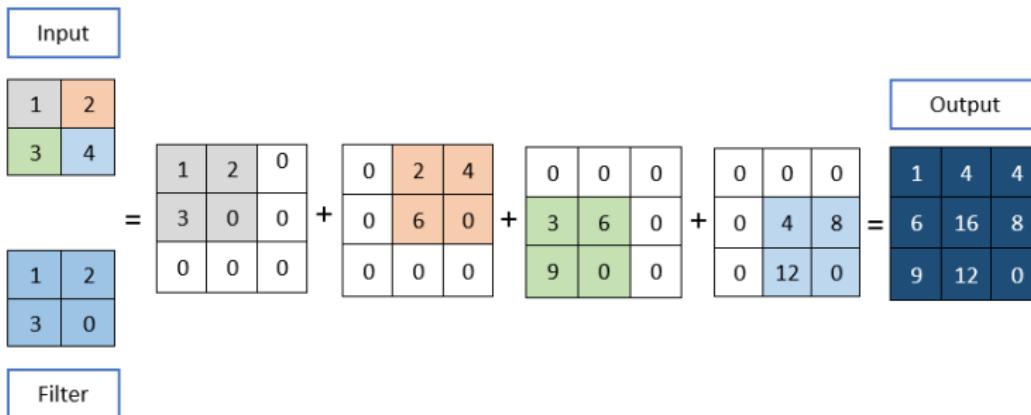
Транспонированная свёртка:  $h \times w \rightarrow H \times W$ ,  
результат-домножением на  $P^T$ :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \circledast^T \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} a\alpha & a\beta + b\alpha & b\beta \\ a\gamma + c\alpha & a\delta + b\gamma + c\beta + d\alpha & b\delta + d\beta \\ c\gamma & c\delta + d\gamma & d\delta \end{pmatrix}$$

$$= \text{reshape} \left\{ \underbrace{\begin{pmatrix} \alpha & 0 & 0 & 0 \\ \beta & \alpha & 0 & 0 \\ 0 & \beta & 0 & 0 \\ \gamma & 0 & \alpha & 0 \\ \delta & \gamma & \beta & \alpha \\ 0 & \delta & 0 & \beta \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & \delta & \gamma \\ 0 & 0 & 0 & \delta \end{pmatrix}}_{P^T} \right\} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

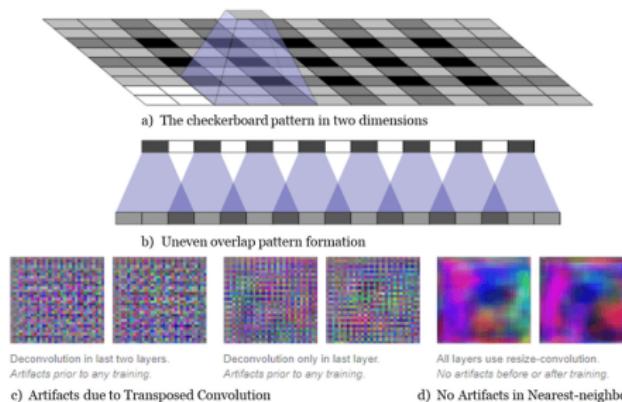
# Интуиция

Транспонированная свертка суммирует "штампы" фильтра с весами входов:



# Недостаток

Приводит к артефактам "шахматной доски" (checkerboard artifacts):



Есть способы борьбы с этим:  $\text{stride} = \text{kernel size}$ , применять несколько раз с перекрытием и смещением, нормировать на число перекрытий, свёртка после транспонированной свёртки.

# Содержание

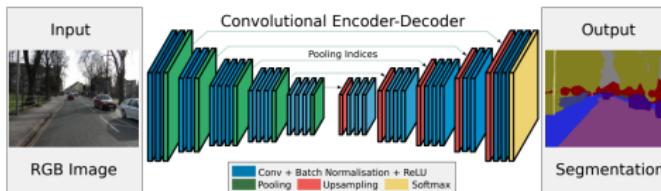
## 1 Введение

## 2 Нейросетевые архитектуры

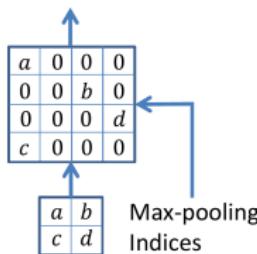
- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

## 3 Instance и panoptic сегментация

# SegNet<sup>9</sup>



- Проблема: при пулинге теряем пространственную информацию.
- Решение: в декодировщике - max unpooling
  - в котором локация максимума сохраняется



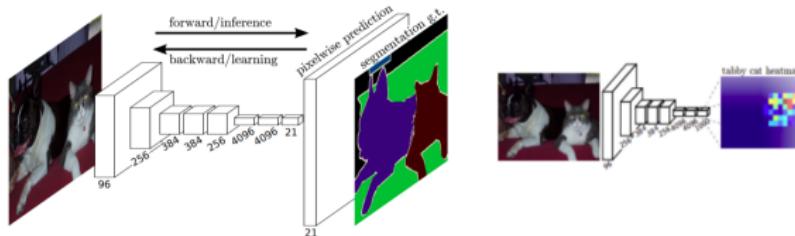
<sup>9</sup><https://arxiv.org/pdf/1511.00561.pdf>

## 2 Нейросетевые архитектуры

- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

# Семейство архитектур FCN<sup>10</sup>

- Семейство архитектур FCN (fully convolutional networks)
- Для кодировщика использовались первые слои VGG, затем кодировщик+декодировщик дообучались под задачу.
- малоразмерное промежуточное представление => грубые неточные границы объектов



<sup>10</sup><https://arxiv.org/pdf/1411.4038.pdf>

# Архитектура FCN-8s

- Прибавление расширенных (upsampling) предыдущих слоев к текущему позволяет совместить:
  - низкоразмерную высокогородневую информацию
  - высокоразмерную низкогородневую информацию

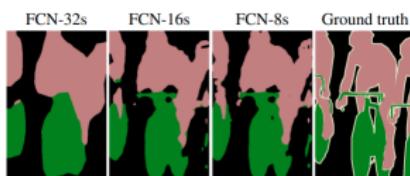
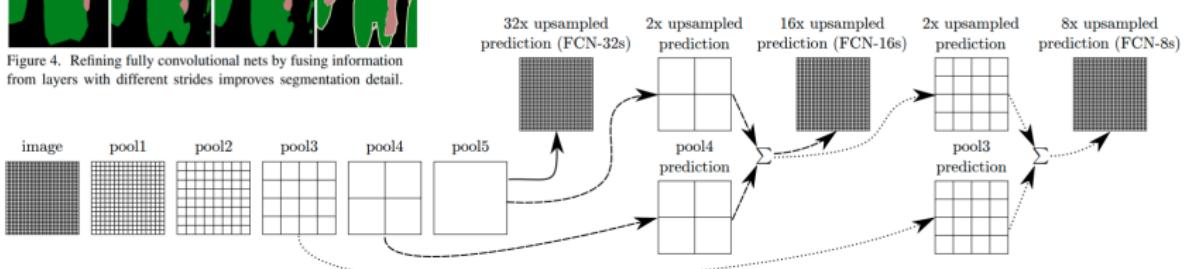


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail.

	pixel acc.
FCN-32s-fixed	83.0
FCN-32s	89.1
FCN-16s	90.0
FCN-8s	<b>90.3</b>

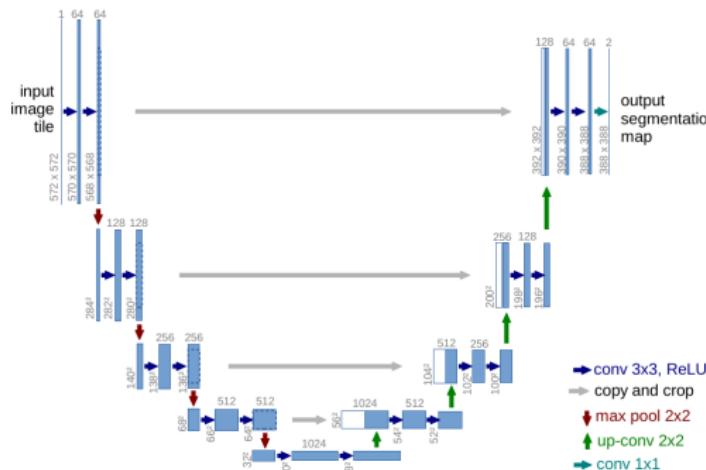
архитектура пулингов и прогнозов  
(свертки не показаны):



## 2 Нейросетевые архитектуры

- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

# Архитектура U-net<sup>11</sup>



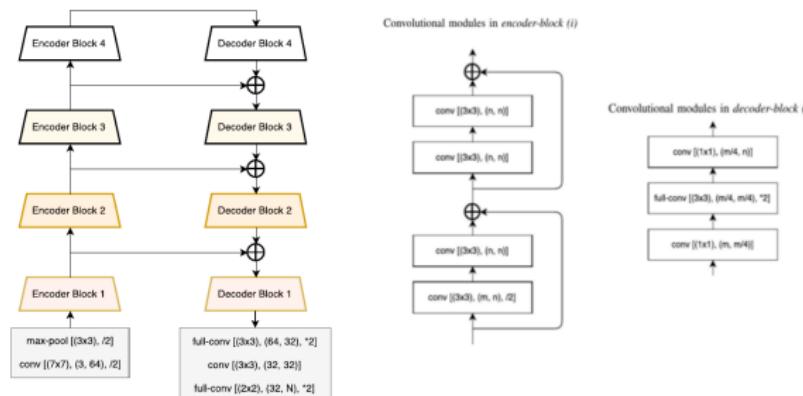
- Горизонтальные числа: С; вертикальные числа: HxW.
- up-conv - перемасштабирование & свертка.
- Серые стрелки: комбинируем низко/высокоуровневые признаки (конкатенация вдоль каналов)

<sup>11</sup><https://arxiv.org/pdf/1505.04597.pdf>

# LinkNet<sup>12</sup>

Как U-net, но агрегирует информацию через сумму представлений, а не конкатенацию.

- В кодировщике: ResNet блоки. В декодировщике:  
1x1 свертка  $\downarrow \#$  параметров, свертка 3x3, 1x1 свертка  $\uparrow \#$  параметров.

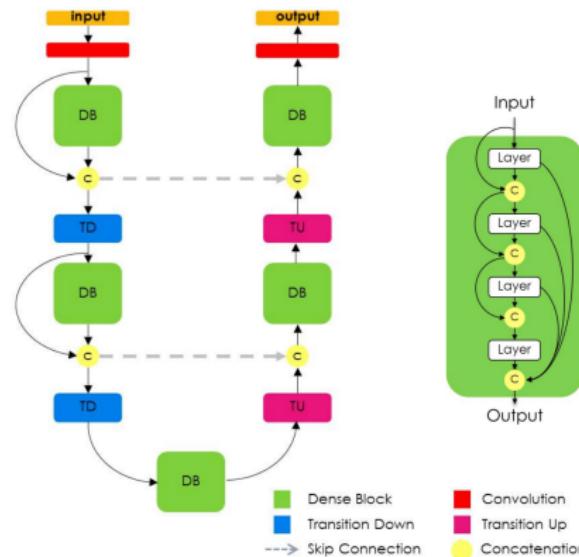


<sup>12</sup><https://arxiv.org/pdf/1707.03718.pdf>

# One Hundred Layers Tiramisu<sup>13</sup>

One Hundred Layers Tiramisu - U-net, состоящая из dense блоков:

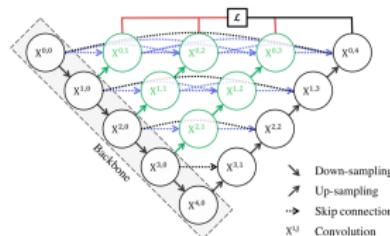
Архитектура и dense block.



<sup>13</sup><https://arxiv.org/pdf/1611.09326.pdf>

## 2 Нейросетевые архитектуры

- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

U-net++<sup>14</sup>

- Черные блоки: ~U-net.
- Идея: приведение в соответствие высокоуровневых и низкоуровневых признаков (зелёные блоки).
- Информация снизу конкатенируется после upsampling.
- Каждый ярус - dense блок (легче настройка, можно ↓ #каналов, т.к. информация не забывается)
- $X^{0,1}, X^{0,2}, X^{0,3}, X^{0,4}$  все выдают прогноз  $Y$  (ансамбль).

<sup>14</sup><https://arxiv.org/pdf/1807.10165.pdf>

# U-net++

- Итоговый прогноз при обучении и применении:

- принцип называется deep supervision

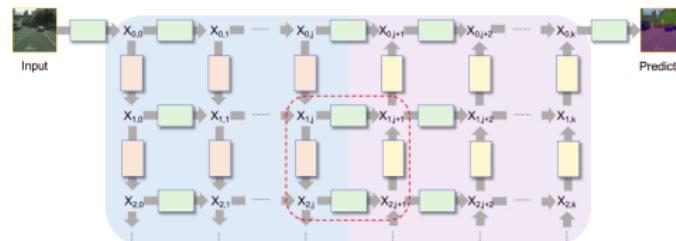
$$\hat{Y} = \frac{1}{4} (X^{0,1} + X^{0,2} + X^{0,3} + X^{0,4})$$

- Настройка: log-likelihood+dice -> max

$$\mathcal{L}(Y, \hat{Y}) = -\frac{1}{N} \sum_{b=1}^N \left( \frac{1}{2} \cdot Y_b \cdot \log \hat{Y}_b + \frac{2 \cdot Y_b \cdot \hat{Y}_b}{Y_b + \hat{Y}_b} \right)$$

# GridNet<sup>15</sup>

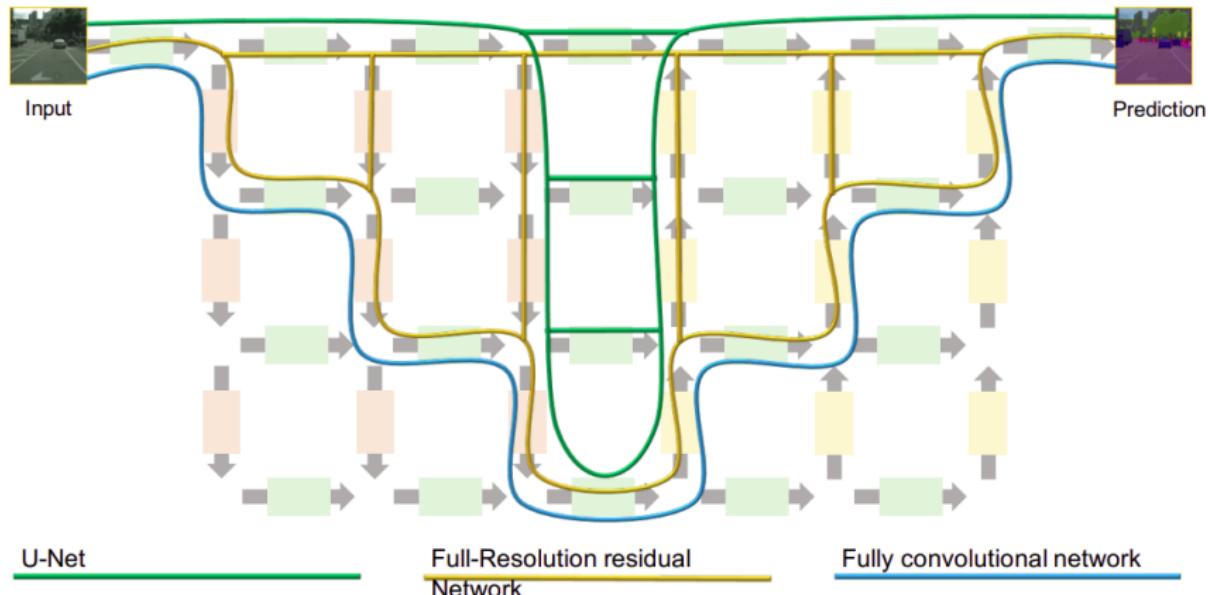
- Отличия GridNet от Unet++:
  - обучение и прогнозы - только по самому правому верхнему элементу.
  - $X_{0,0}, X_{0,1}, \dots, X_{0,k}$  - все агрегируют как низкоуровневые, так и высокоуровневые (одного порядка) признаки.
- Выходы и входы элементов агрегируются суммированием.
- Обучаем ансамбль, т.к. выход определяется информацией, полученной по разным путям от входа к выходу.



- Модель не показала улучшение качества.

<sup>15</sup><https://arxiv.org/pdf/1707.07958.pdf>

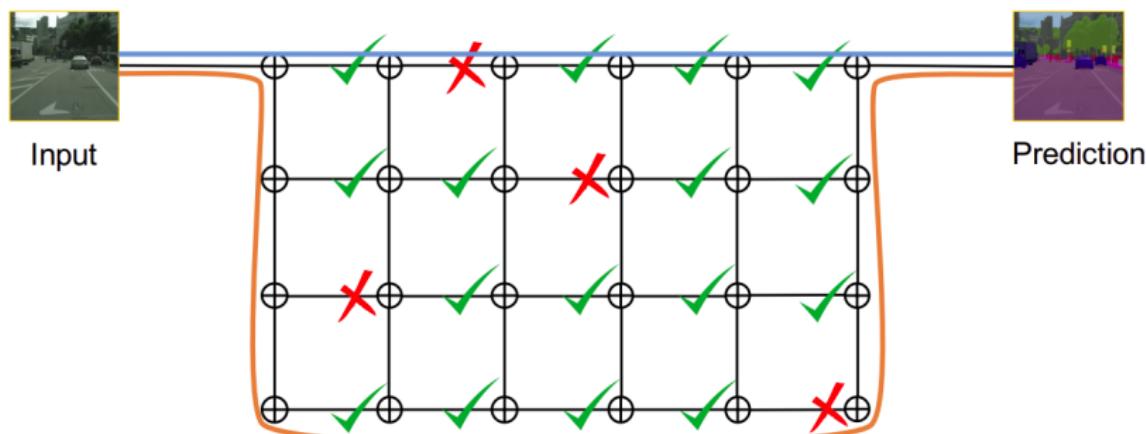
# GridNet обобщает другие архитектуры



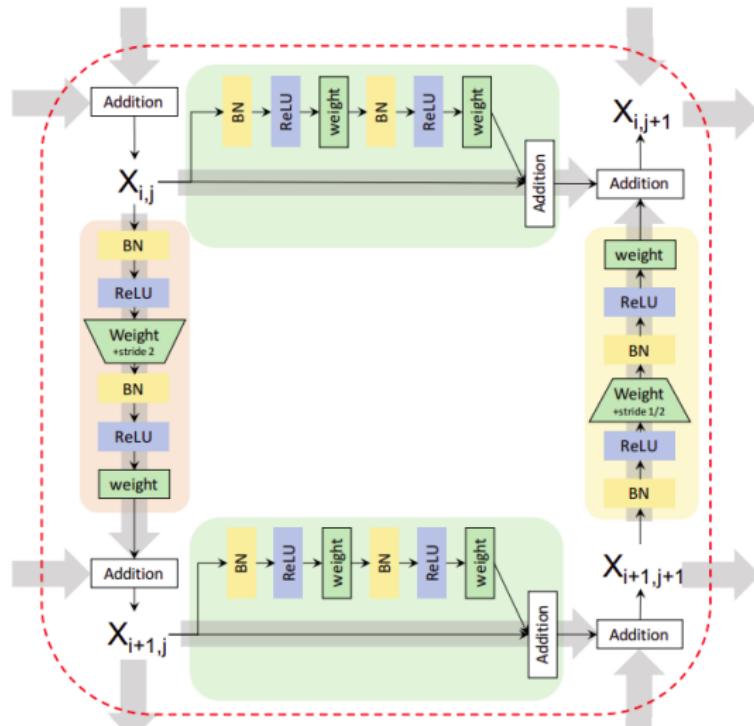
# DropOut в GridNet

DropOut - отбрасывается поднабор горизонтальных связей.

- модель учится использовать информацию с разных ярусов



# GridNet - каждый элемент

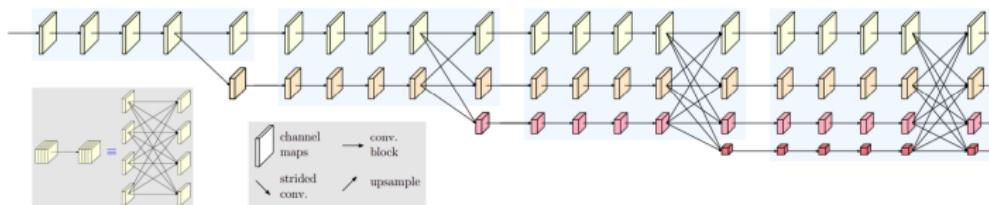


# High-Resolution Representations

Модель High-Resolution Representations<sup>16</sup> ↑ качество.

Идея: вместо объединения только 2 представлений (низко и высокоуровневого) параллельно поддерживаются представления в нескольких разрешениях

- обмен информацией - конкатенация выходов разного разрешения (downsampling: strided conv, upsampling: bilinear)
- сеть сама выбирает, как и когда использовать низко и высокоуровневую информацию.



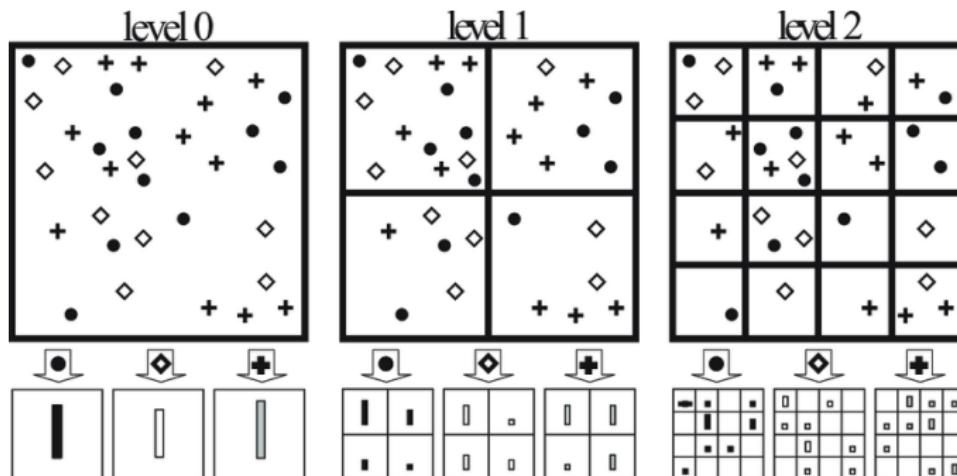
<sup>16</sup><https://arxiv.org/pdf/1904.04514.pdf>

## 2 Нейросетевые архитектуры

- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

# PSPNet<sup>17</sup>

- По сравнению с FCN PSPNet учитывает глобальный контекст с помощью SpatialPyraidePooling.



<sup>17</sup><https://arxiv.org/pdf/1612.01105.pdf>

# PSPNet

В примере FCN классифицирует лодку машиной, а PSPNet-лодкой, т.к. в окрестности воды.



(a) Image



(b) Ground Truth

sky
tree
grass
earth
plant
car
boat



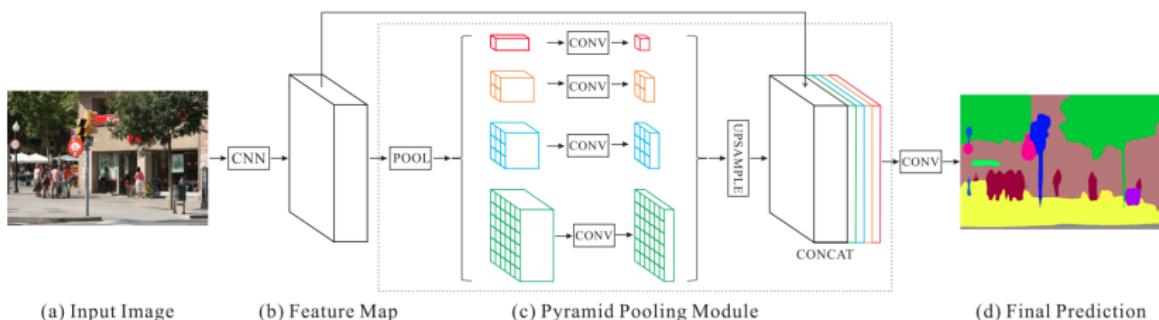
(c) FCN



(d) PSPNet

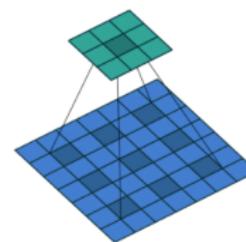
# PSPNet

Кодировщик: предобученное начало ResNet. Пирамидальный пулинг для агрегации общей информации, затем расширение (upsampling) для совместимости с высокоразмерным представлением.

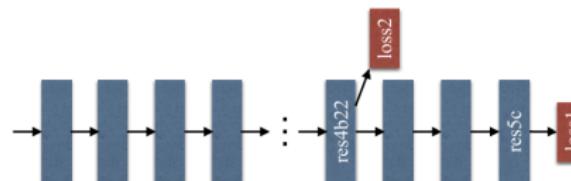


# PSPNet

- CNN кодировщик - ResNet со свёртками с dilation:



- Во время обучения используется дополнительная ф-ция потерь в середине (принцип deep supervision):



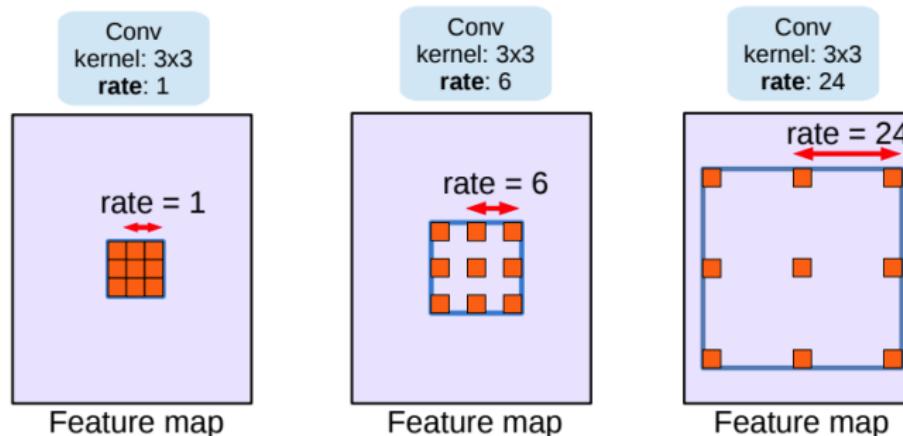
## 2 Нейросетевые архитектуры

- Архитектура FCN
- Архитектура U-net
- Архитектурные расширения U-net
- Учёт глобальной информации
- Использование dilated свёрток

## DeepLab V3 (2017)<sup>18</sup>

Используются dilated convolutions (называемые Atrous convolutions).

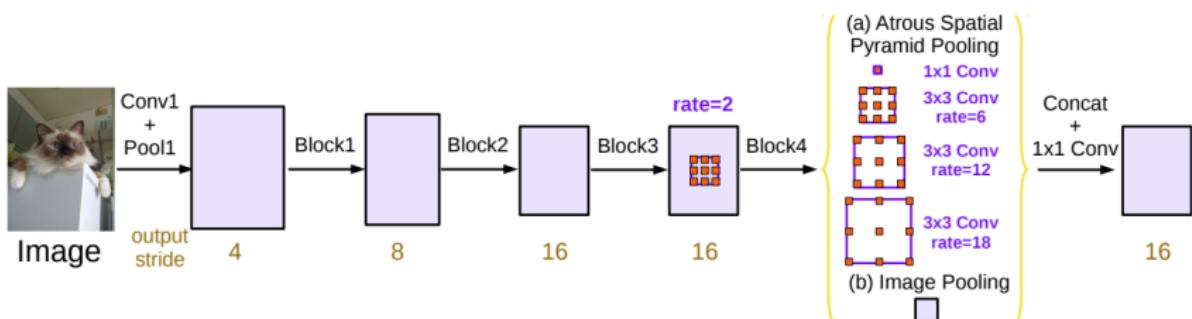
- позволяют собирать информацию по более широкой окрестности при том же #вычислений и #параметров.



<sup>18</sup><https://arxiv.org/pdf/1706.05587v3.pdf>

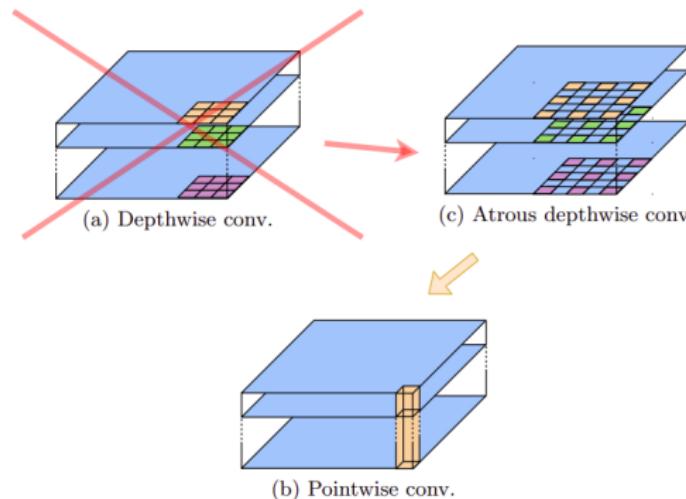
# DeepLab V3: блок объединения результатов

- Конкatenируются рез-ты dilated conv с разным dilation
  - +GlobalAvgPooling->conv 1x1->растягивается по размеру feature map
- Потом применяется conv 1x1 ( $\downarrow$  размерности)



# DeepLab V3+ (2018)<sup>19</sup>

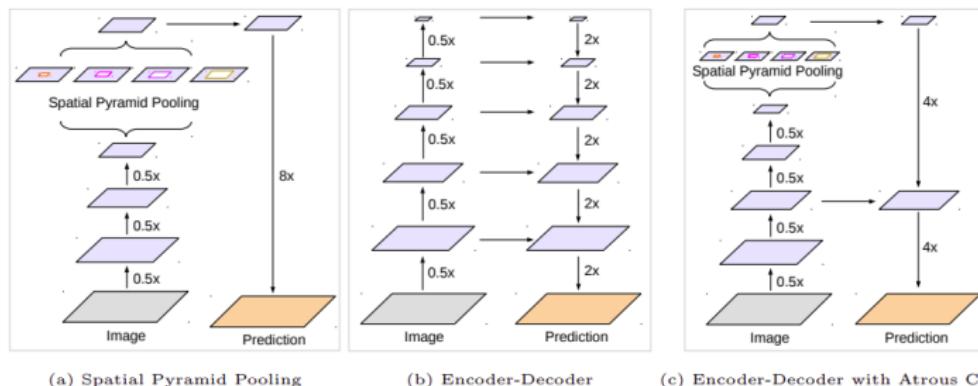
Использует depthwise separable conv (depthwise, потом conv 1x1), но для ↑ области видимости использованы dilated depthwise separable свёртки:



<sup>19</sup><https://arxiv.org/pdf/1802.02611v3.pdf>

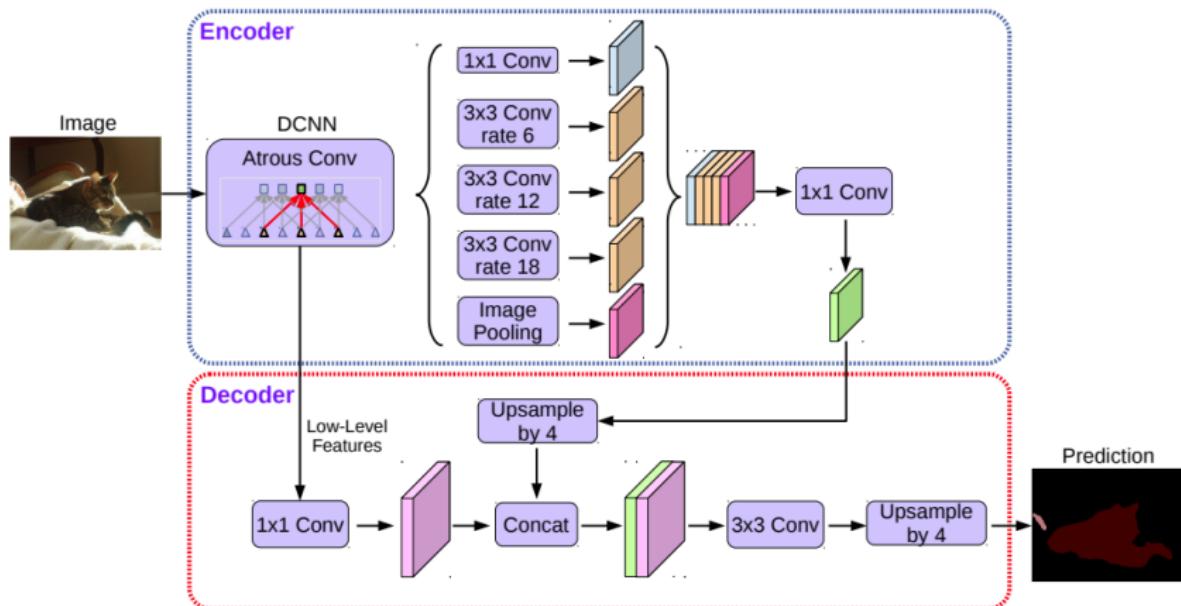
# DeepLab V3+: дополнительные связи

Для  $\uparrow$  точности комбинируются низкоуровневые и высокуюровневые признаки в кодировщике и декодировщике:



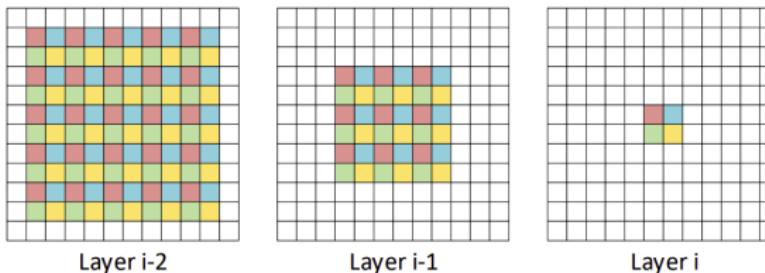
**Fig. 1.** We improve DeepLabv3, which employs the spatial pyramid pooling module (a), with the encoder-decoder structure (b). The proposed model, DeepLabv3+, contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows us to extract features at an arbitrary resolution by applying atrous convolution.

# DeepLab V3+



# Особенности работы с dilated свёртками

Выход dilated свёрток зависит от разных пикселей:



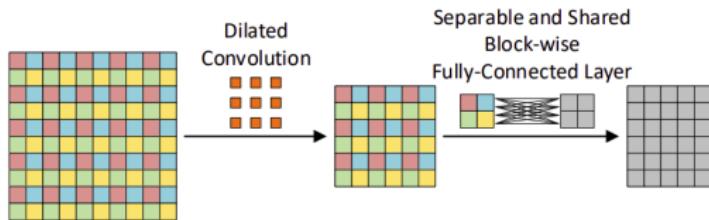
- Получаем пространственную нестабильность прогнозов, т.к. они определяются разными блоками пикселей.
- Smoothed Dilated Convolutions for Improved Dense Prediction<sup>20</sup>: предлагается 2 подхода для сглаживания выходов.
  - подходит  $\uparrow$  точность сегментации DeepLab V2.

<sup>20</sup><https://arxiv.org/pdf/1808.08931.pdf>

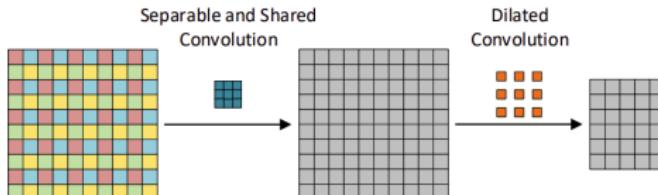
# Smoothed Dilated Convolutions<sup>21</sup>

Предлагаются 2 подхода (иллюстрация для dilation=2):

- ① патч 2x2 → патч 2x2, используя полно связный слой:  
скользим им с одинаковыми весами по каждой карте  
признаков в отдельности со stride=2:



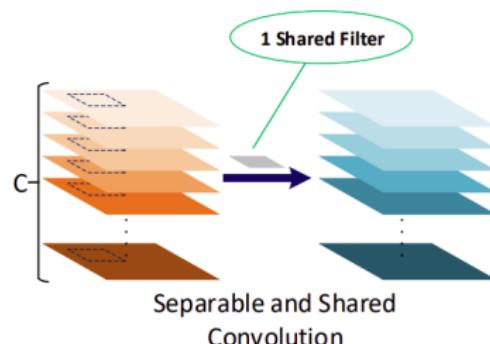
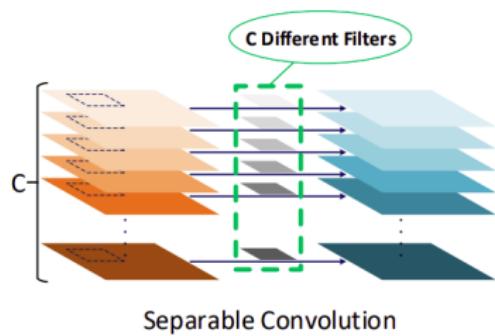
- ② либо применить separable & shared conv перед dilated conv:



<sup>21</sup><https://arxiv.org/pdf/1808.08931.pdf>

# Smoothed Dilated Convolutions

- Т.е. 2й подход: **separable & shared conv** - свёртка, действующая с одинаковыми весами на каждый канал в отдельности:

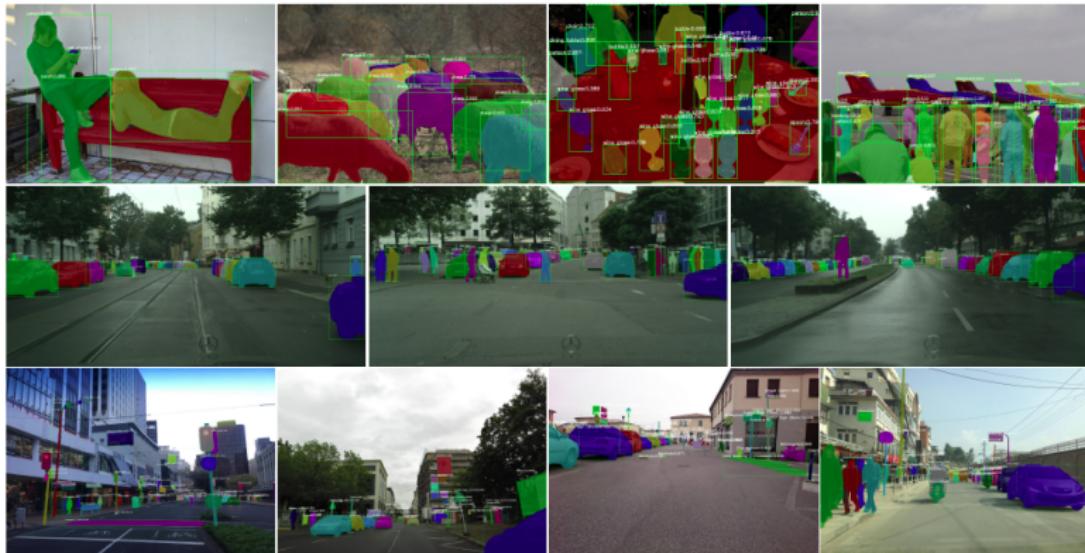


# Содержание

- 1 Введение
- 2 Нейросетевые архитектуры
- 3 Instance и panoptic сегментация

# Path Aggregation Network for Instance Segmentation<sup>22</sup>

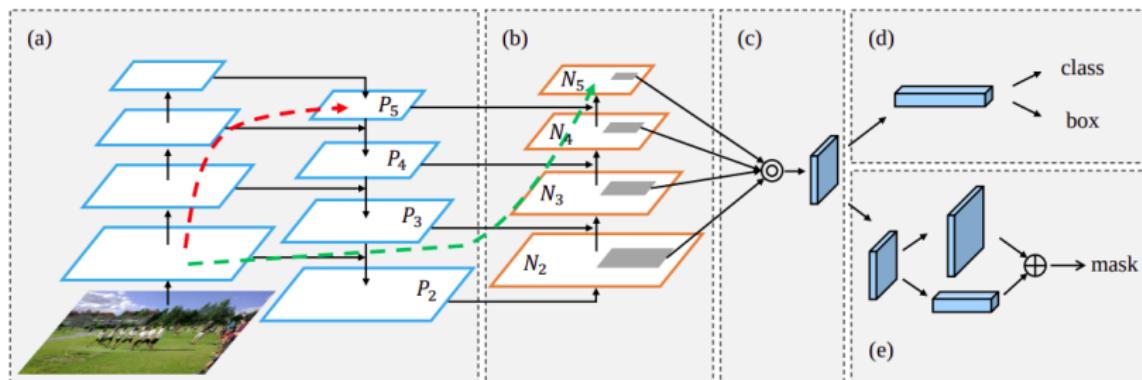
Path Aggregation Network (PANet) реализует instance-сегментацию.



<sup>22</sup><https://arxiv.org/pdf/1803.01534.pdf>

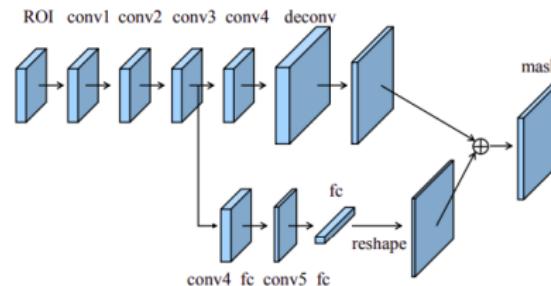
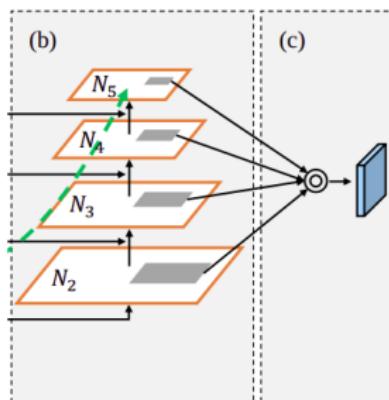
# Path Aggregation Network for Instance Segmentation

Для каждого обнаруженного объекта предсказывается рамка, класс и маска выделения.



# Path Aggregation Network for Instance Segmentation

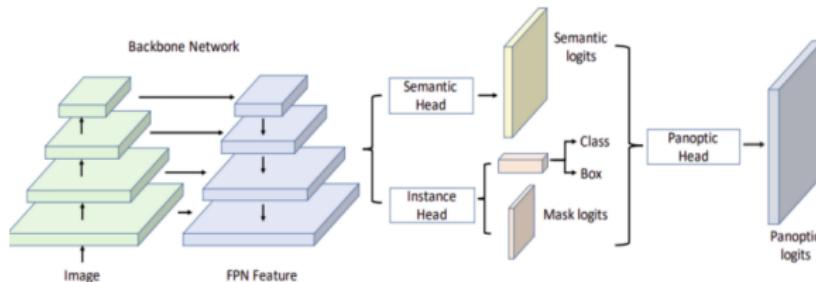
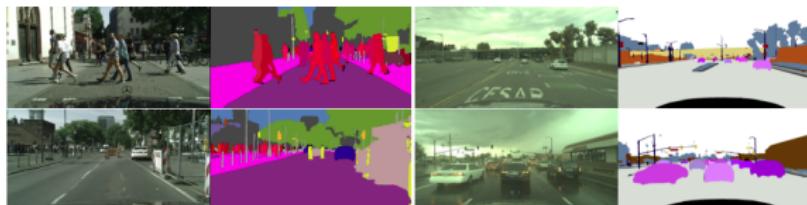
- Adaptive feature pooling (c) использует max-pooling для каждой (x,y) позиции для каждого канала по разным представлениям (b).
- Маска выделения (mask prediction) предсказывается агрегацией выходов
  - свёрточного слоя (учитываем локальную информацию)
  - полносвязного слоя (учитываем позицию)



Mask prediction branch with fully-connected fusion.

# USPNet<sup>23</sup>

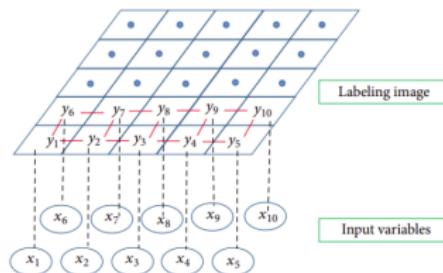
- USPNet реализует panoptic segmentation.
- Блок Panoptic head: агрегация семантической и instance сегментации:



<sup>23</sup><https://arxiv.org/pdf/1901.03784.pdf>

## Условные случайные поля<sup>25</sup>

Условные случайные поля (conditional random fields<sup>24</sup>): -  
надстройка стандартных моделей, учитывающая взаимосвязи  
между соседними прогнозами  $Y_{i,j}$  и  $Y_{u,v}$ .



<sup>24</sup> Введение в условные случайные поля.

<sup>25</sup> <https://downloads.hindawi.com/journals/mpe/2016/3846125.pdf>

## Условные случайные поля

- Минимизируем энергию несоответствия входов и выходов, а также выходов между собой (energy minimization):

$$E(\mathbf{X}, \mathbf{Y}) = \sum_{i,j} D(X_{i,j}, Y_{i,j}) + \sum_{(i,j)} \sum_{(u,v) \in \mathcal{N}(i,j)} V(Y_{i,j}, Y_{u,v}) \rightarrow \min_{\mathbf{Y}}$$

- $(i,j), (u, v)$ -пространственные позиции,  $\mathcal{N}(i,j)$  - окрестность позиций, связанных с  $(i,j)$
- $D(X_{i,j}, Y_{i,j})$  - связь  $X_{i,j}$  (окрестность пикселей вокруг  $(i,j)$ ) и  $Y_{i,j}$  (модель классификации).
- $V(Y_{i,j}, Y_{u,v})$  - связь соседних меток.
- Минимизация энергии эквивалентна максимизации правдоподобия:

$$p(\mathbf{Y}|\mathbf{X}) \propto e^{-E(\mathbf{X}, \mathbf{Y})} \rightarrow \max_{\mathbf{Y}}$$

## Заключение

- Семантическая сегментация реализуется полносвёрточными архитектурами.
- Ключевая проблема: сохранение & учёт информации
  - с высокоуровневых признаков (семантика)
  - низкоуровневых признаков ( $\uparrow$  точности выделения границ)
- Решение проблемы:
  - суммирование/конкатенация промежут. признаков
  - добавление канала GlobalPooling / SpatialPyramidPooling
  - dilated свёртки  $\uparrow$  область видимости
- Метрики качества: accuracy, IoU, dice
  - их можно не только отслеживать, но и оптимизировать по ним
- Instance-сегментация: свой класс для каждого объекта.
- Panoptic-сегментация: Instance-сегментация + сегментация фона.