

# Задача сегментации

Даниил Лысухин, ML Team Lead @ Ozon



# План

- Задача сегментации и ее виды
- Semantic Segmentation
- Instance Segmentation: Mask R-CNN
- BatchNorm и проблемы памяти
- Итоги

# Сегментация



# Локализация объектов

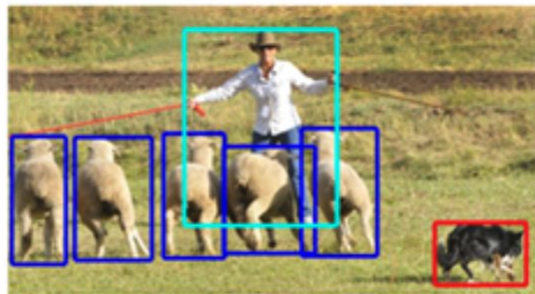
- На прошлых лекциях рассматривали задачу **детектирования объектов** (object detection)
  - Класс объектов
  - Положение объектов (через bounding boxes)
- Это - частный случай **задачи локализации**

# Локализация объектов

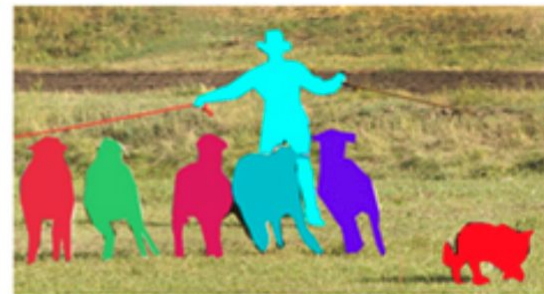
- Другой частный случай локализации - **сегментация**
- В сегментации положение объекта задается **попиксельной маской**



(a) classification

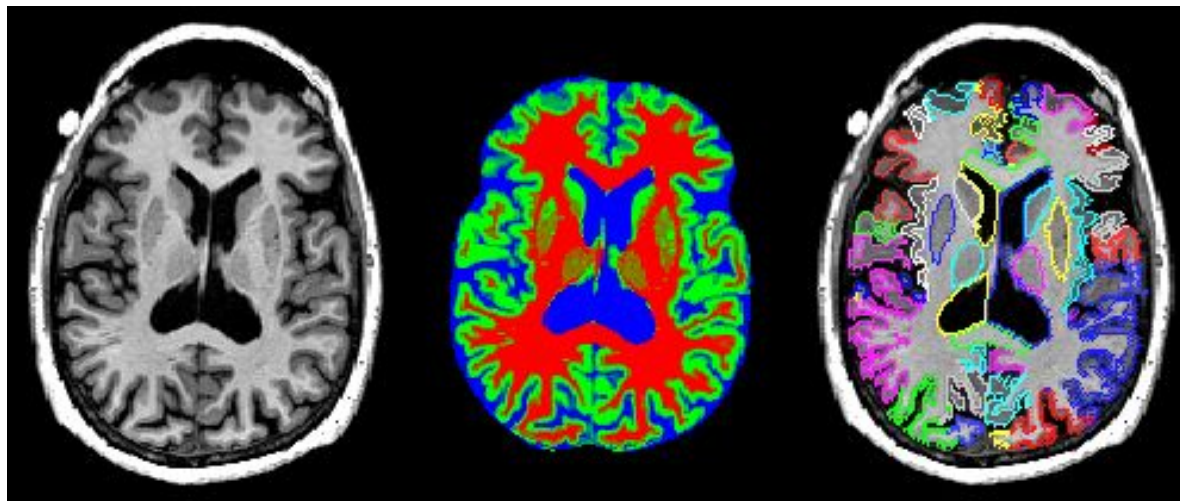


(b) detection



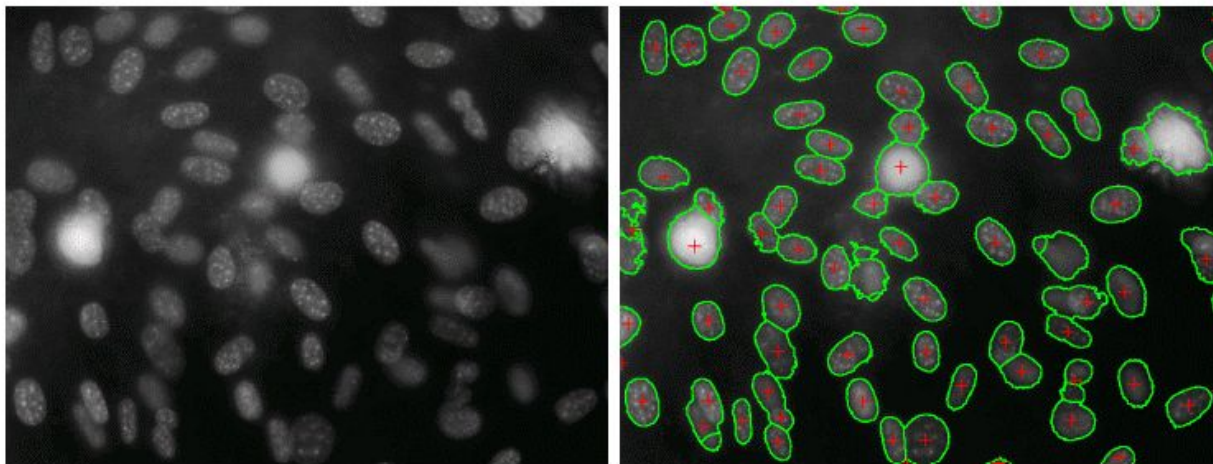
(c) segmentation

# Сегментация - примеры



Томографические  
снимки

# Сегментация - примеры



Снимки с  
микроскопа

# Сегментация - примеры



Спутниковые снимки /  
аэрофото



## Сегментация - примеры



Сегментация "фона"

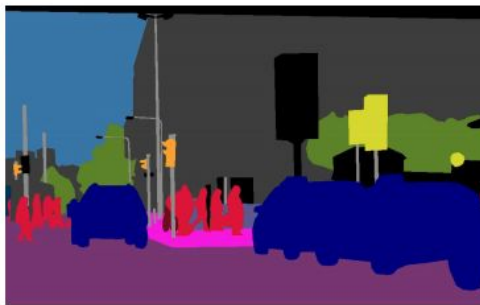
## Сегментация - виды

- Сегментация позволяет более точно **определять форму** интересующих объектов
- Есть несколько видов сегментации с **разной полнотой извлекаемой информации**:
  - Semantic Segmentation
  - Instance Segmentation
  - Panoptic Segmentation

# Сегментация - виды



(a) image



(b) semantic segmentation



(c) instance segmentation

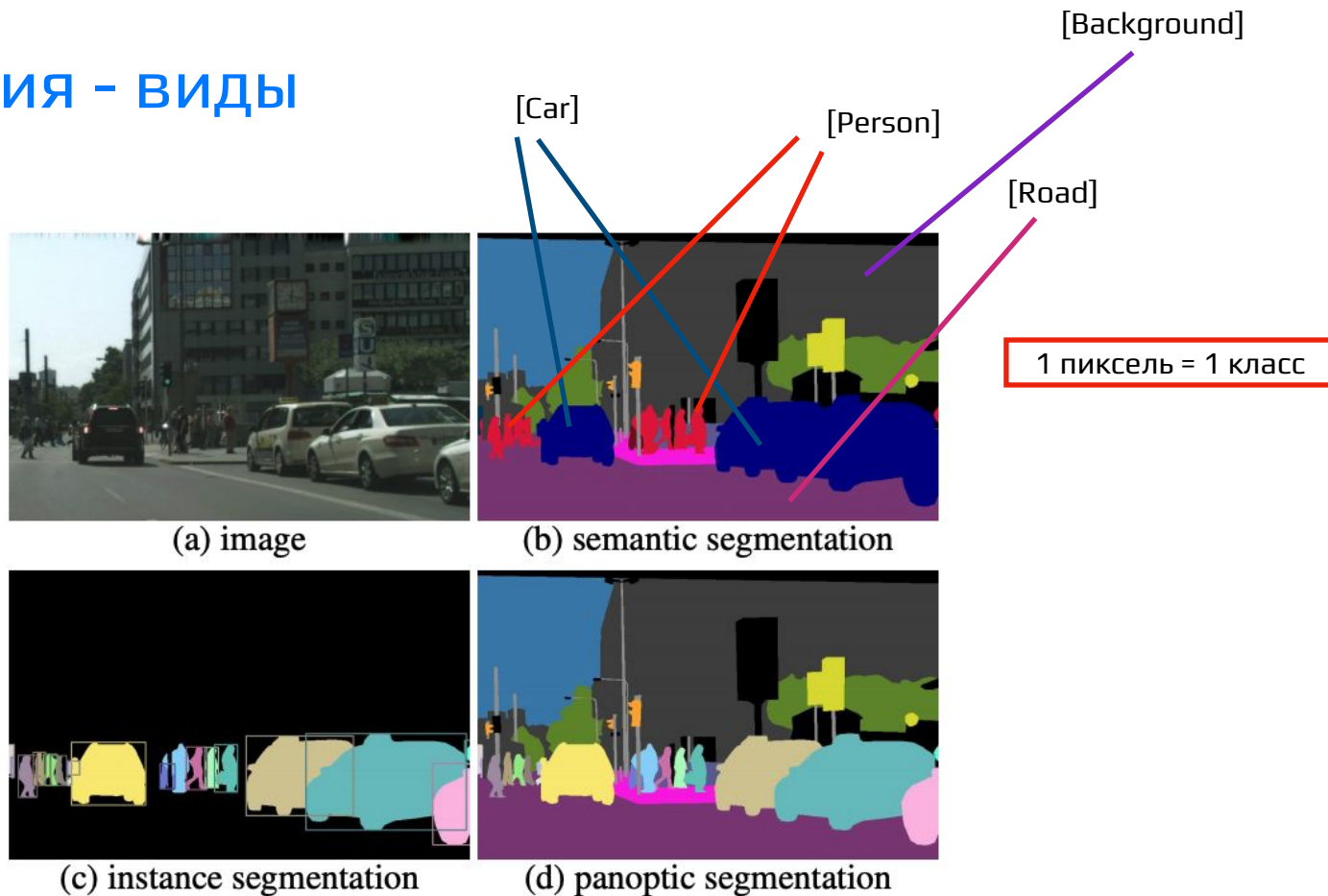


(d) panoptic segmentation

## Сегментация - виды

- **Семантическая (semantic)** - каждому пикселю (кроме фона) ставится в соответствие номер класса

# Сегментация - виды



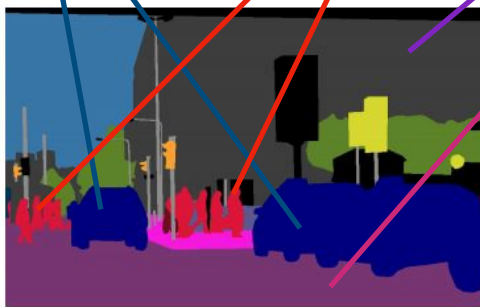
## Сегментация - виды

- Семантическая (semantic) - каждому пикселю (кроме фона) ставится в соответствие номер класса
- **Объектная (instance)** - разделяются маски разных экземпляров одного класса

# Сегментация - виды



(a) image



(b) semantic segmentation

[Background]

[Car]

[Person]

[Road]

1 пиксель = 1 класс



(c) instance segmentation



(d) panoptic segmentation

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс  
+ ID экземпляра

[Car, #ID1]

## Сегментация - виды

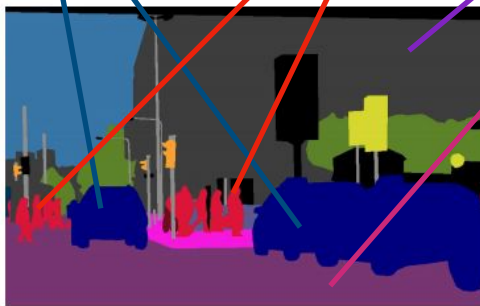
- Семантическая (semantic) - каждому пикселю (кроме фона) ставится в соответствие номер класса
- Объектная (instance) - разделяются маски разных экземпляров одного класса
- **Panoptic**
  - для классов типа “небо”, “дорожное полотно” и т.д. (stuff) только класс,
  - для остальных (things) - отдельная маска



# Сегментация - виды



(a) image



(b) semantic segmentation

[Background]

[Car]

[Person]

[Road]

1 пиксель = 1 класс



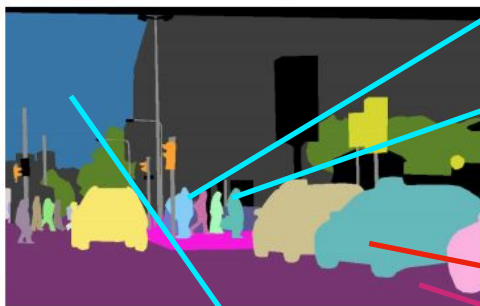
(c) instance segmentation

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс  
+ ID экземпляра

[Car, #ID1]



(d) panoptic segmentation

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс  
(+ ID экземпляра  
для "вещей")

[Car, #ID1]

[Road]

[Sky]

## Сегментация - виды

- Семантическая (semantic) - каждому пикселю (кроме фона) ставится в соответствие номер класса
- Объектная (instance) - разделяются маски разных экземпляров одного класса
- Panoptic
  - для классов типа “небо”, “дорожное полотно” и т.д. (stuff) только класс,
  - для остальных (things) - отдельная маска

# Семантическая сегментация



# Семантическая сегментация

- Семантическая сегментация  $\sim$  попиксельная классификация
  - На входе в модель - изображение (RGB, RGBD, ...), облако точек, ...
  - На выходе - набор масок разных классов
- Рассмотрим бинарный случай - классы “фон” и “объект”

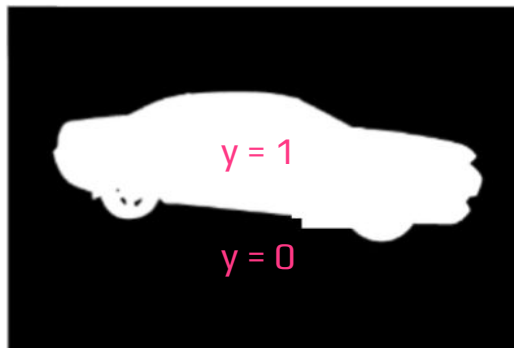
# Семантическая сегментация

RGB



# Семантическая сегментация

RGB

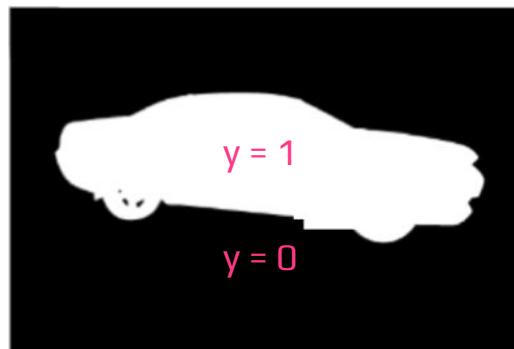


Ground-truth

маска

# Семантическая сегментация

RGB



Ground-truth

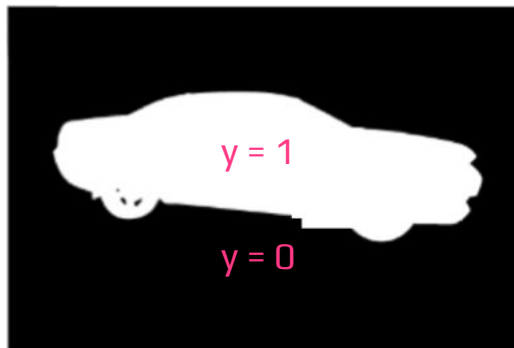
маска

# Семантическая сегментация

RGB



Модель



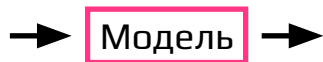
Ground-truth

маска

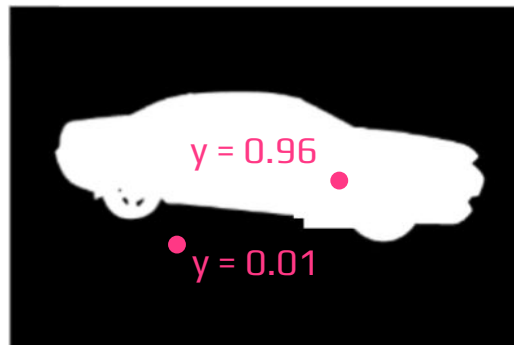


# Семантическая сегментация

RGB

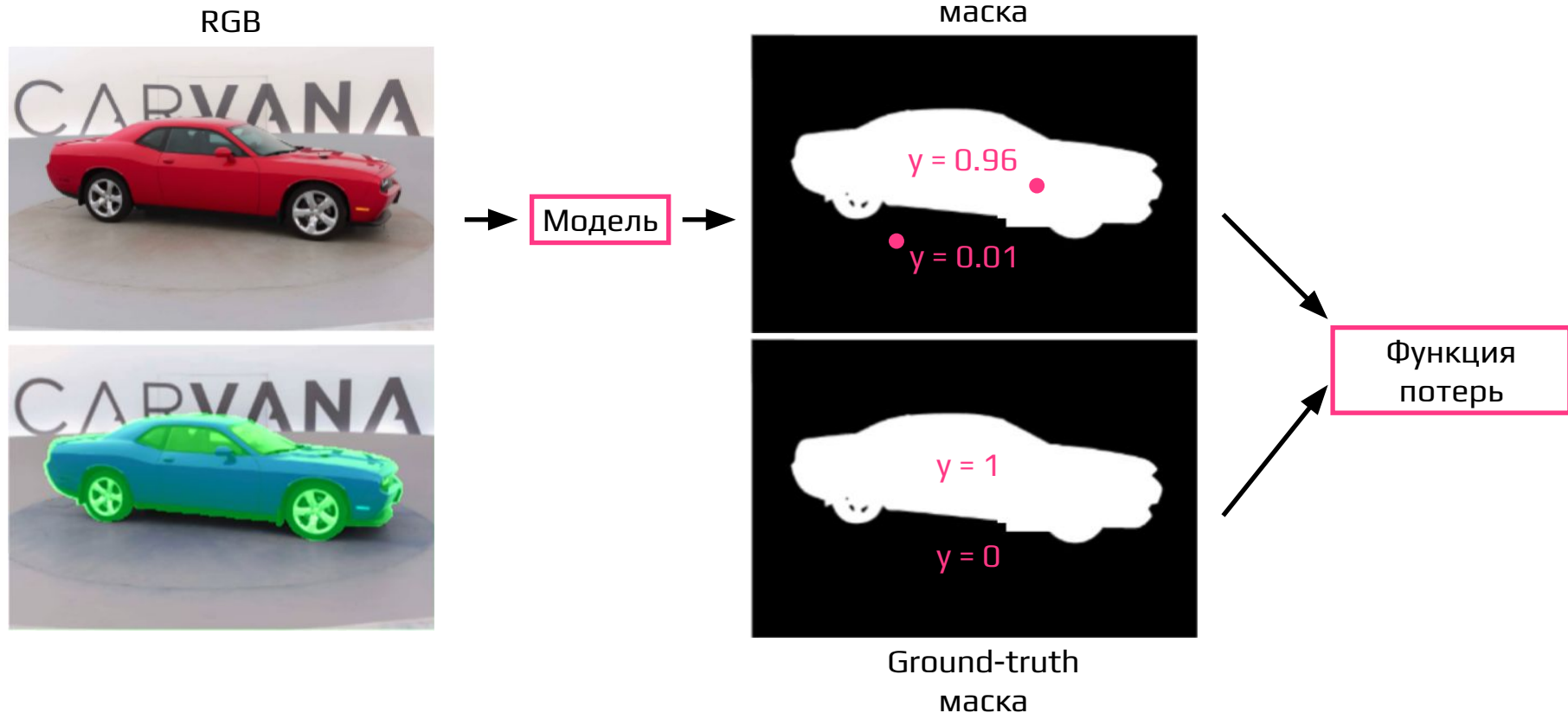


Предсказанная  
маска



Ground-truth  
маска

# Семантическая сегментация



# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?

# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
  - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
  - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- Считаем в каждом пикселе, усредняем по площади

# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?

- Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

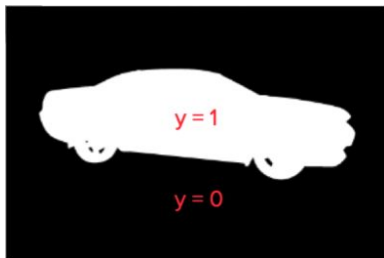
- Считаем в каждом пикселе, усредняем по площади
  - Какие проблемы?

# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
  - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- Считаем в каждом пикселе, усредняем по площади
  - Какие проблемы?

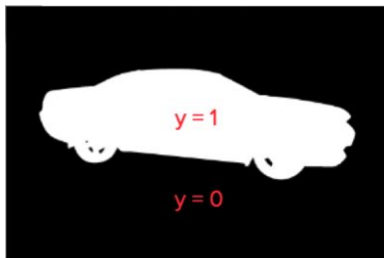


# Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
  - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

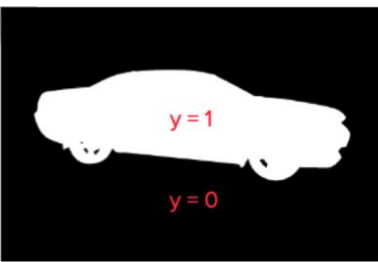
- Считаем в каждом пикселе, усредняем по площади
  - Какие проблемы?





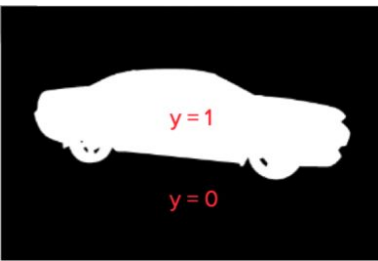
# Семантическая сегментация - дисбаланс классов

- Аналогично задаче детектирования, может быть сильный дисбаланс классов
- Возможные решения:



# Семантическая сегментация - дисбаланс классов

- Аналогично задаче детектирования, может быть сильный дисбаланс классов
- Возможные решения:
  - Веса для балансировки вклада разных пикселей
  - Focal Loss
  - Другие функции (см. дальше)



# Семантическая сегментация - архитектура

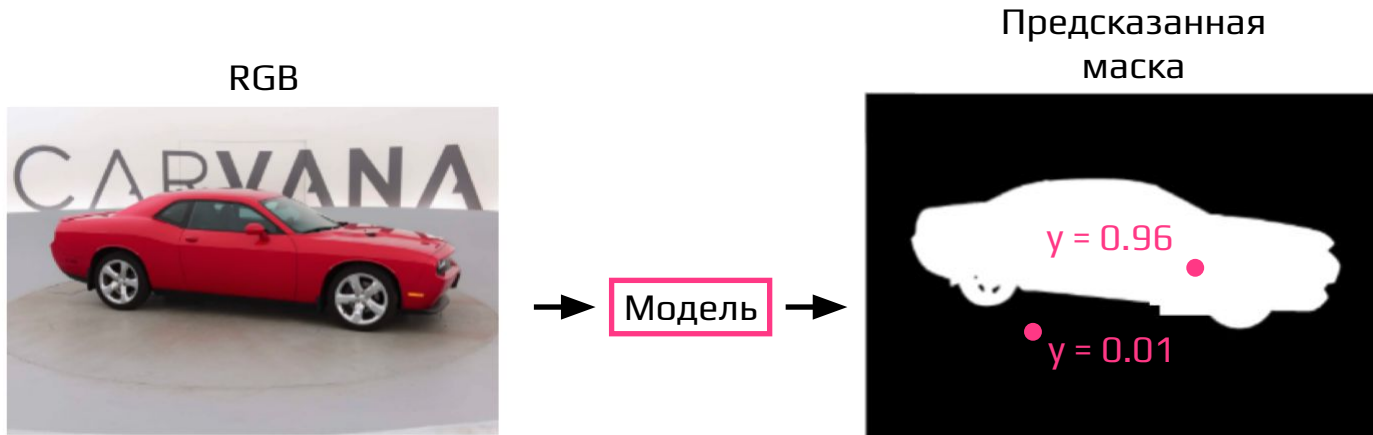
- Какой может быть архитектура модели?

# Семантическая сегментация - архитектура

- Какой может быть архитектура модели?
- "Интерфейс" модели:
  - На входе - изображение,  $H \times W \times C$
  - На выходе -  $K$  масок,  $H \times W \times K$

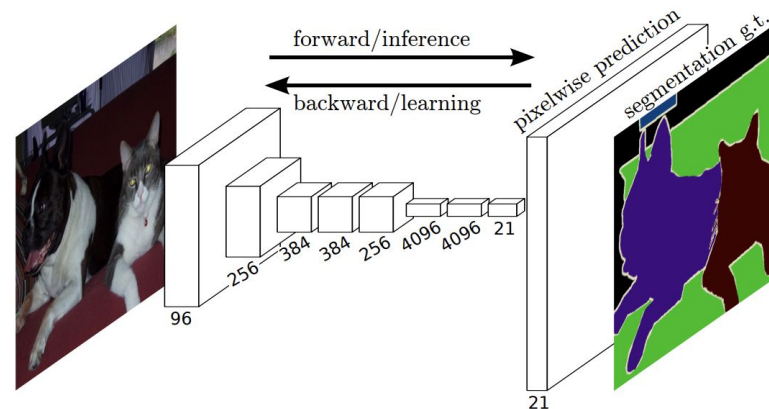
# Семантическая сегментация - архитектуры

- Какой может быть архитектура модели?
- "Интерфейс" модели:
  - На входе - изображение,  $H \times W \times C$
  - На выходе -  $K$  масок,  $H \times W \times K$



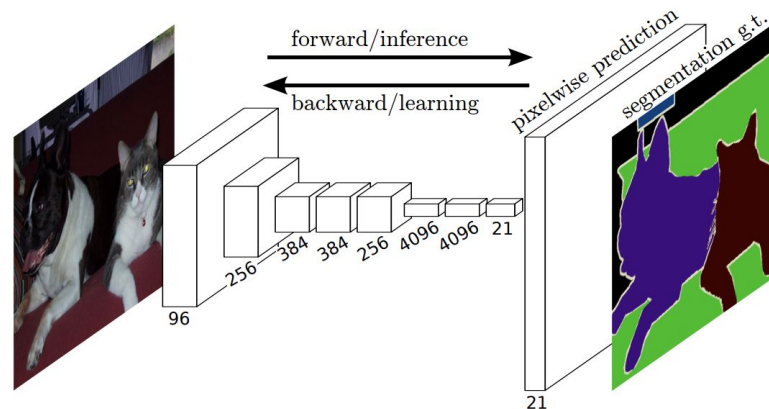
# Fully-Convolutional Network (FCN) (2014)

- Fully convolutional networks for Semantic Segmentation (2014)
  - Backbone (VGG)
  - Upsampling для приведения к исходному размеру



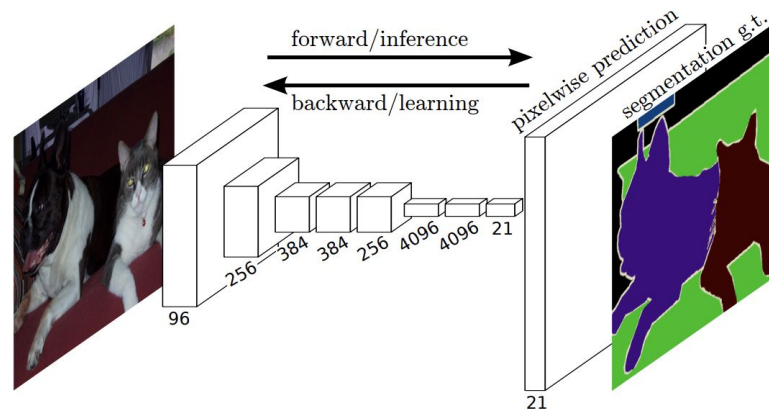
# Fully-Convolutional Network (FCN) (2014)

- Fully convolutional networks for Semantic Segmentation (2014)
  - Backbone (VGG)
  - Upsampling для приведения к исходному размеру
- Какие тут проблемы?



# Fully-Convolutional Network (FCN) (2014)

- Fully convolutional networks for Semantic Segmentation (2014)
  - Backbone (VGG)
  - Upsampling для приведения к исходному размеру
- Какие тут проблемы?
  - Узкое бутылочное горлышко
  - Резкое увеличение H/W





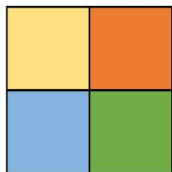
# Interlude

- Как увеличить размер (H, W) карты активаций?
  - Интерполяция значений (Upsampling)
  - Транспонированная свертка (Transposed Conv)

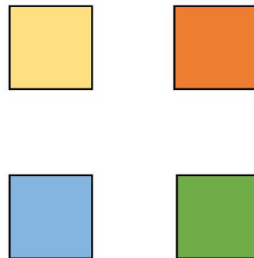
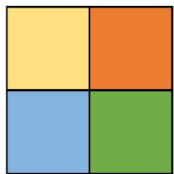
# Interlude

- Как увеличить размер (H, W) карты активаций?
  - **Интерполяция значений (Upsampling)**
  - Транспонированная свертка (Transposed Conv)

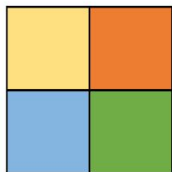
# Upsampling



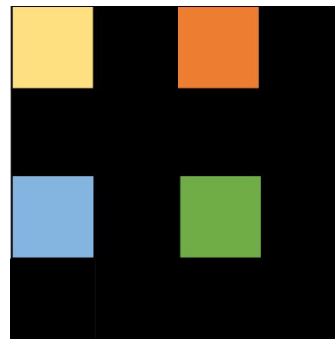
# Upsampling



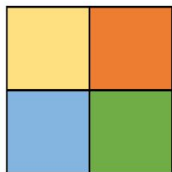
# Upsampling



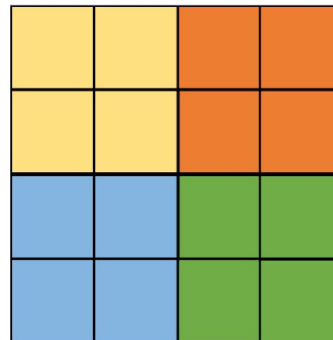
Без интерполяции



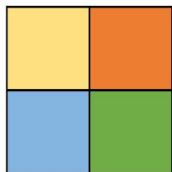
# Upsampling



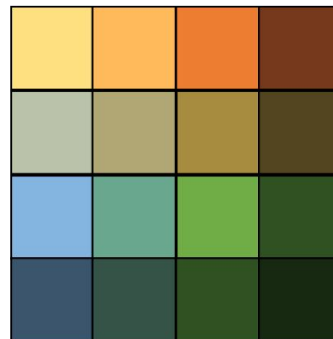
Интерполяция методом  
ближайшего соседа



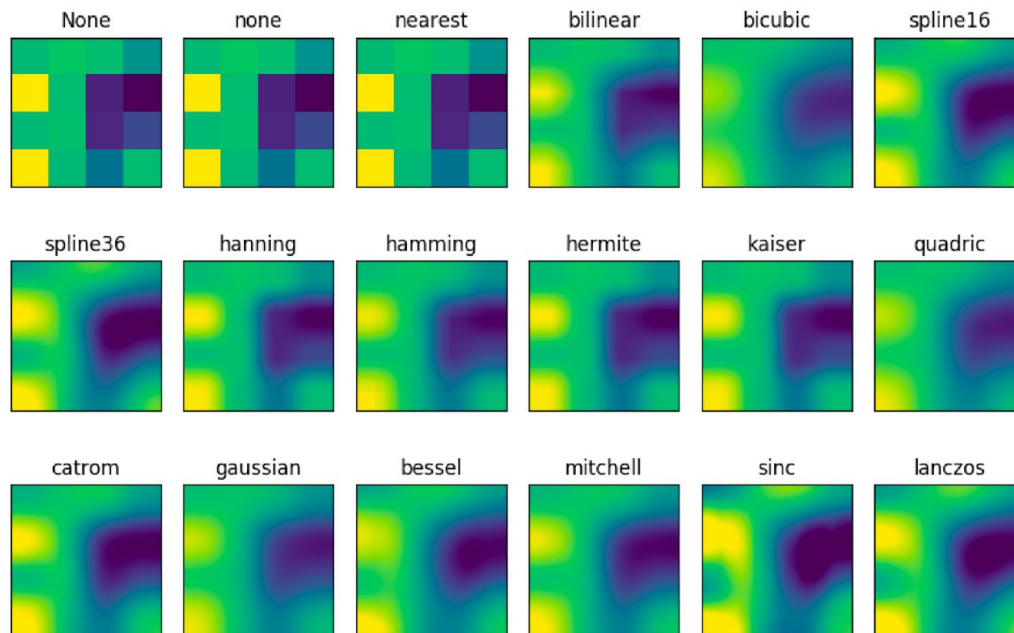
# Upsampling



(би-)Линейная  
интерполяция



# Upsampling

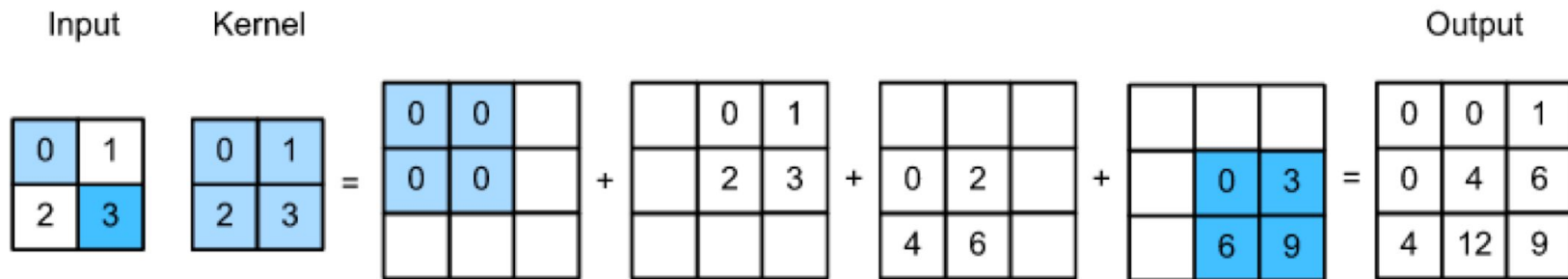




# Interlude

- Как увеличить размер (H, W) карты активаций?
  - Интерполяция значений (Upsampling)
  - **Транспонированная свертка (Transposed Conv)**

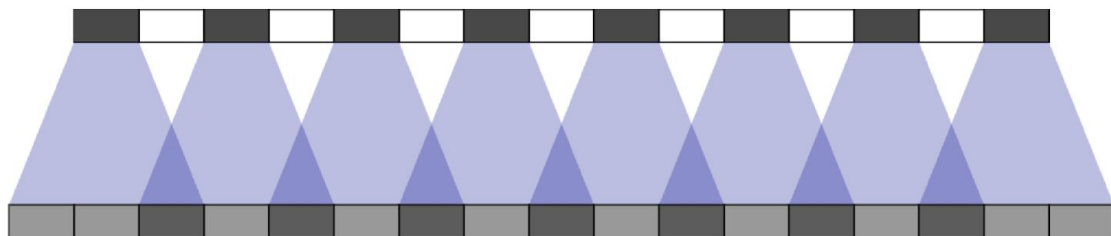
# Transposed Conv



- Вместо "сворачивания" с ядром (как в обычном ConvLayer) происходит "разворачивание" входного сигнала
- Значения входного сигнала выступают весами перед ядром транспонированной свертки

# Transposed Conv

- В некоторых случаях может появляться "шахматный" паттерн в выходном сигнале
- [Deconvolution and Checkerboard Artifacts](#)

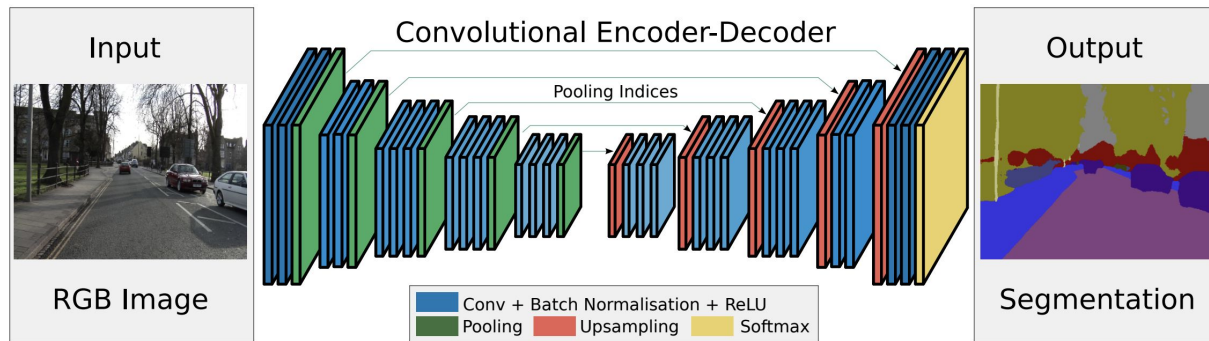


stride = 2

size = 3

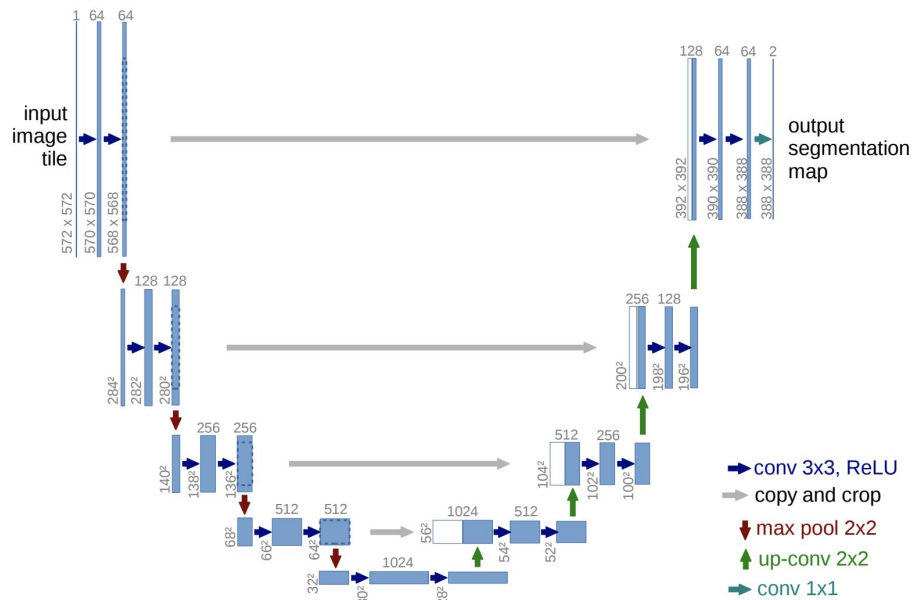
# SegNet (2015)

- SegNet: A Deep Convolutional Encoder-Decoder Architecture
  - Симметричная архитектура вида Encoder-Decoder
  - Постепенный Upsampling



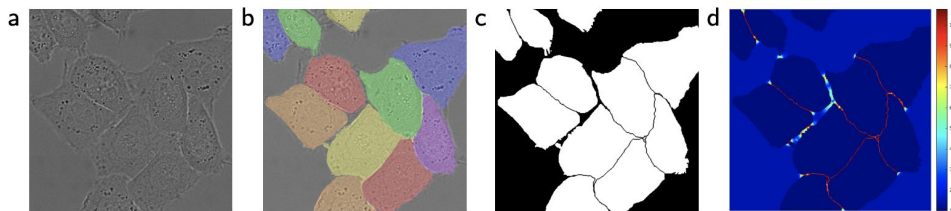
# UNet (2015)

- U-Net: CNNs for Biomedical Image Segmentation
  - Добавили горизонтальные связи к Encoder-Decoder

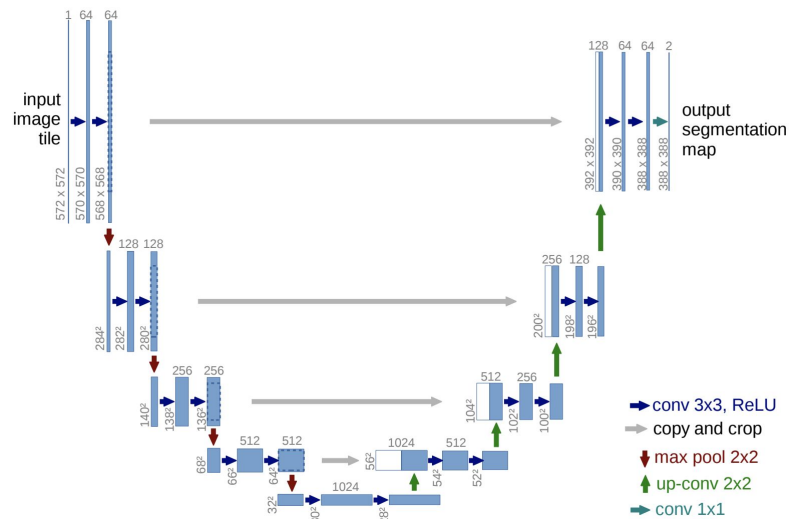


# UNet (2015)

- Сильное улучшение сегментации на границах объектов
- Всего лишь 30 изображений 512x512 для обучения!



**Fig. 3.** HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels.

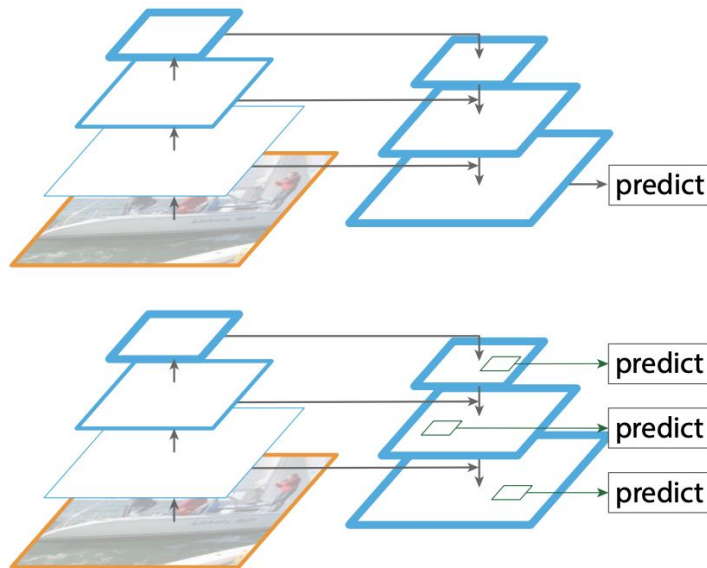


## UNet-like

- Из конкретной архитектуры для сегментации UNet давно превратился в "подход" для задач image-to-image:
  - Сегментация (subj)
  - Колоризация (предсказание цветных каналов для grayscale-входа)
  - InPainting ("закрашивание" пустот)
  - ...
- UNet-like сеть =
  - encoder (resnet, efficientnet, ...) +
  - decoder

# Feature Pyramid Networks (FPN) (2017)

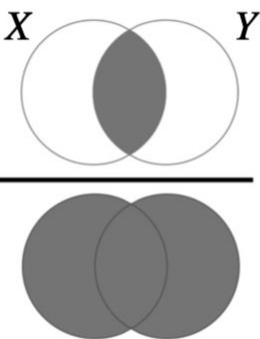
- [Feature Pyramid Networks for Object Detection \(2017\)](#)
- Та самая пирамида признаков, что использовали в RetinaNet
- Идея подойдет и для улучшения сегментации





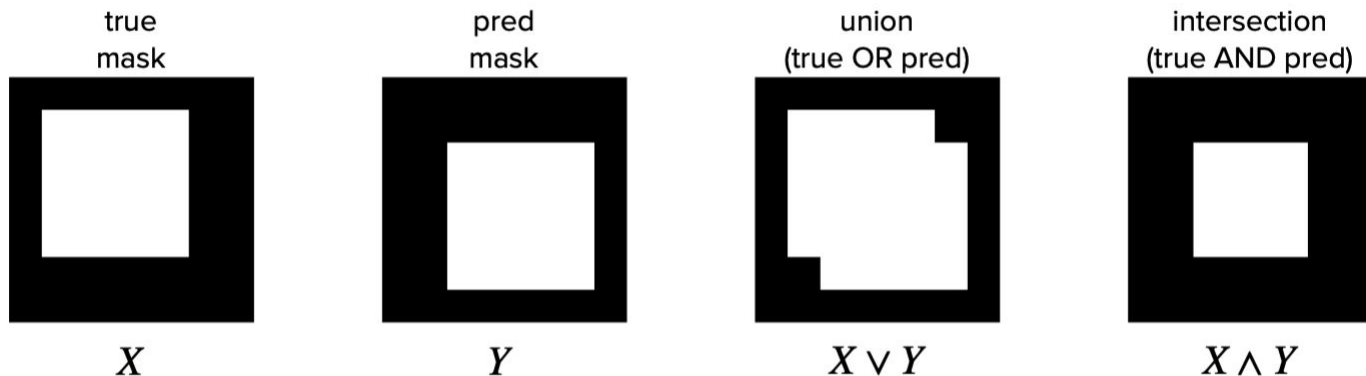
## Beyond BCE

- Применение кросс-энтропии может сломаться о дисбаланс классов (и не только в сегментации)
- Вспомним, что в детекторах объектов говорили про понятие **Intersection-over-Union (IoU)** (синоним - Jaccard Index):


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$
$$\text{Jaccard}(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

## Jaccard Index для сегментации

- По аналогии определим Jaccard Index для пары сегментационных масок:



$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

## Jaccard Index для сегментации

- Напрямую оптимизировать Jaccard Index нельзя (почему?)
- Но можно аппроксимировать его, например:

$$J_{seg} = \frac{\sum_{x,y} M_{gt}(x,y) * M_{pred}(x,y)}{\sum_{x,y} M_{gt}(x,y) + M_{pred}(x,y) - M_{gt}(x,y) * M_{pred}(x,y)}$$

- Получить из этого лосс можно, например, так:

$$Loss_J = 1 - \log(J_{seg})$$

- Часто комбинируют с BCE:

$$Loss = \alpha * Loss_{BCE} + (1 - \alpha) * Loss_J$$

# Segmentation Losses

- Jaccard Loss vs Dice Loss - постоянная [путаница](#)
- [A survey of loss functions for semantic segmentation \(2020\)](#)

## Semantic Segmentation: итоги

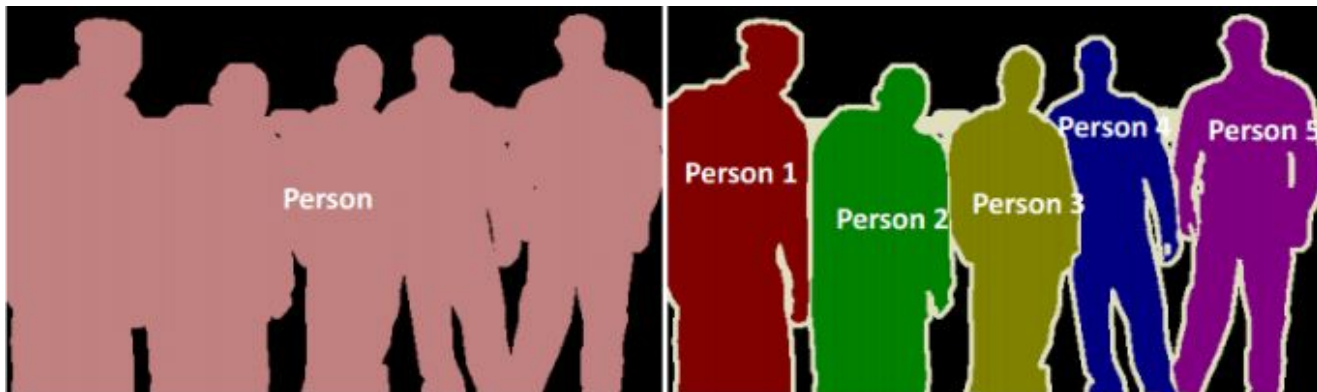
- Используется, когда достаточно разбить пиксели по классам
- Модели получают на вход изображения и отдают маски с вероятностями для каждого класса (U-Net)
- Для обучения используют попиксельный BCE/Focal + Jaccard/Dice

# Instance Segmentation



# Instance Segmentation

- Если объекты не “касаются” и не перекрывают друг друга, то разделить их маски не составит труда и после semantic segmentation
- ... Но так бывает не всегда



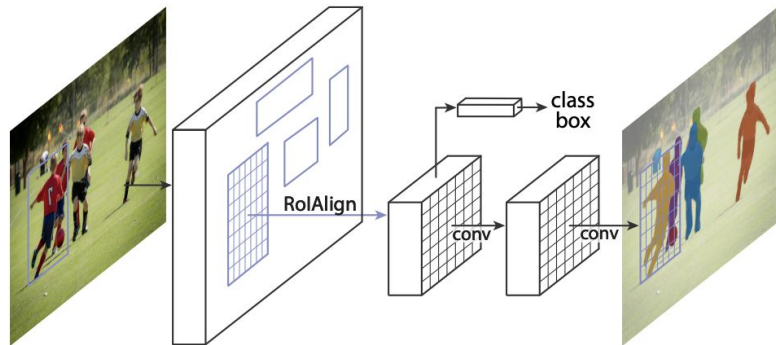
# Instance Segmentation

- Идея: встроить в детектор объектов семантическую сегментацию для бокса вокруг каждого объекта



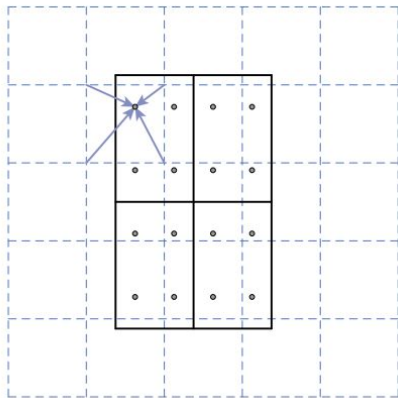
## Mask R-CNN (2017)

- [Mask R-CNN \(2017\)](#)
- В основе - Faster R-CNN
- Дополнительная ветвь для предсказания бинарной маски во всех proposals
  - Маски не зависят от классов (т.е. сегментация на 1 класс)
- Вместо RoIPool — RoIAlign



## Mask R-CNN (2017) - RoIAlign

- В Mask R-CNN используется операция RoIAlign: вместо грубого округления границ и пулинга значений используется интерполяция значений по сетке



**Figure 3. RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with  $2 \times 2$  bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

# Mask R-CNN - примеры



Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask AP* of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

# Mask R-CNN

- ЕСТЬ В [PyTorch](#)

```
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
```

# Проблемы с памятью



# Проблемы с памятью

- Модели для сегментации (и не только) могут быть прожорливы по памяти по сравнению с классификацией
  - Больше карт активаций (encoder + decoder)
  - Больше HW
- Увеличивается потребление памяти на 1 пример -> уменьшается размер батча
  - Чем это грозит?

# Проблемы с памятью

- Малый размер батча ->
  - Шумные градиенты
  - Проблемы с BatchNorm
  - ...

# Проблемы с памятью

- Малый размер батча ->
  - **Шумные градиенты**
  - Проблемы с BatchNorm
  - ...



# Шумные градиенты для малых батчей

- [Gradient Accumulation](#)
- Накапливание градиентов по нескольким батчам перед обновлением весов

# Проблемы с памятью

- Малый размер батча ->
  - Шумные градиенты
  - **Проблемы с BatchNorm**
  - ...

# Малый батч vs BatchNorm

- BN вычисляет статистики по батчу
- Попался выброс в малом батче - беда
- Как быть?
  - Не использовать BN (-> [InstanceNorm](#), ...)
  - Использовать много GPU + [SynchronizedBN](#)
  - Если делаете fine-tuning: [заморозить](#) слои BN
  - Оптимизировать вычисления для BN

# Малый батч vs BatchNorm

- BN вычисляет статистики по батчу
- Попался выброс в малом батче - беда
- Как быть?
  - Не использовать BN (-> [InstanceNorm](#), ...)
  - Использовать много GPU + [SynchronizedBN](#)
  - Если делаете fine-tuning: [заморозить](#) слои BN
  - **Оптимизировать вычисления для BN**

# Recap: BatchNorm

- BatchNorm

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

**Algorithm 1:** Batch Normalizing Transform, applied to activation  $x$  over a mini-batch.

# Recap: BatchNorm

- BatchNorm

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;  
Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

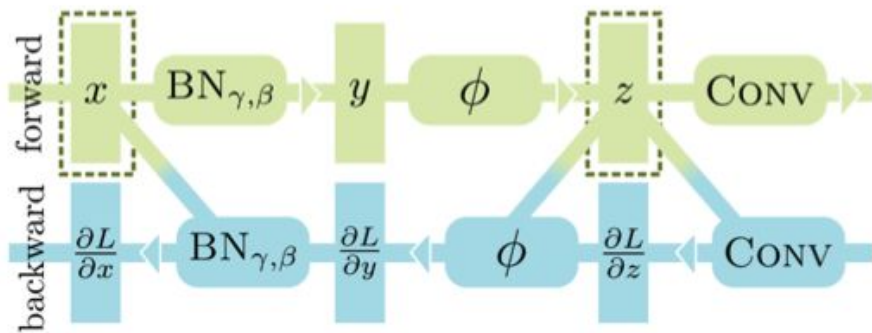
**Algorithm 1:** Batch Normalizing Transform, applied to activation  $x$  over a mini-batch.

- Для обновления параметров слоя требуется хранить тензор  $\mathbf{x}$  после forward pass в буфере
- При backward pass значение  $\mathbf{x}$  извлекается из буфера
- Есть подходы, избавляющие от этой необходимости

# Inplace-ABN (2018)

- [In-Place Activated BN for Memory-Optimized Training of DNNs](#)
- Рассмотрим связку **BN + Activation + Conv**

## Pre-1: Vanilla BN

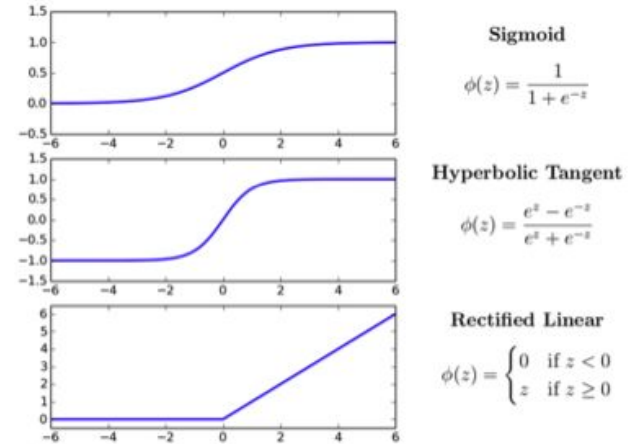
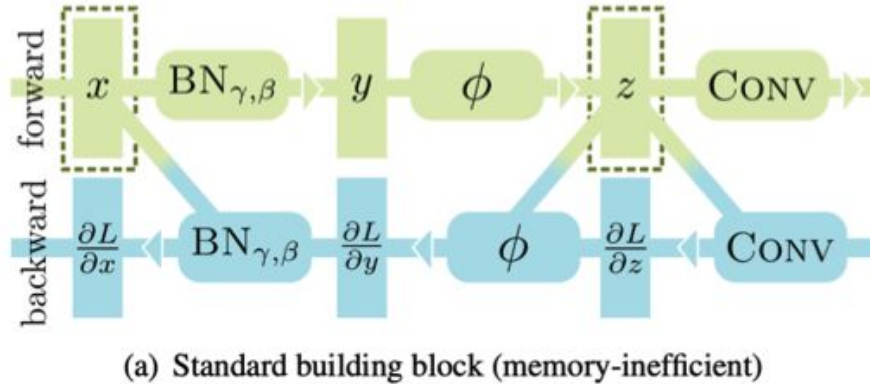


(a) Standard building block (memory-inefficient)

- Для backward pass необходимо хранить только промежуточные тензоры  $\mathbf{x}$ ,  $\mathbf{z}$

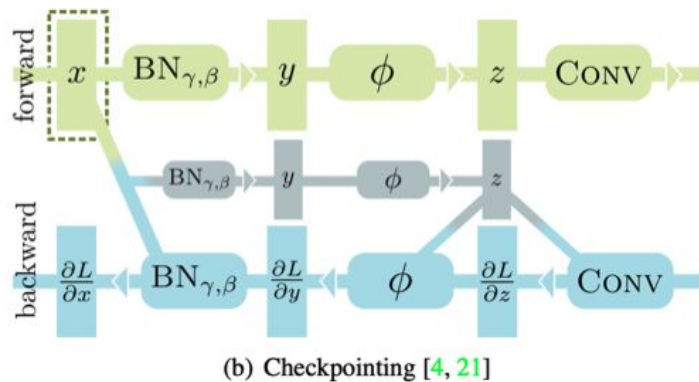


# Pre-1: Vanilla BN



- Для backward pass необходимо хранить только промежуточные тензоры **x**, **z**
  - Для обратимых активаций вроде **Sigmoid/Tanh y** восстанавливается по **z**
  - Для **ReLU** по **z** нельзя восстановить **y**, но легко восстановить градиент **dz/dy**

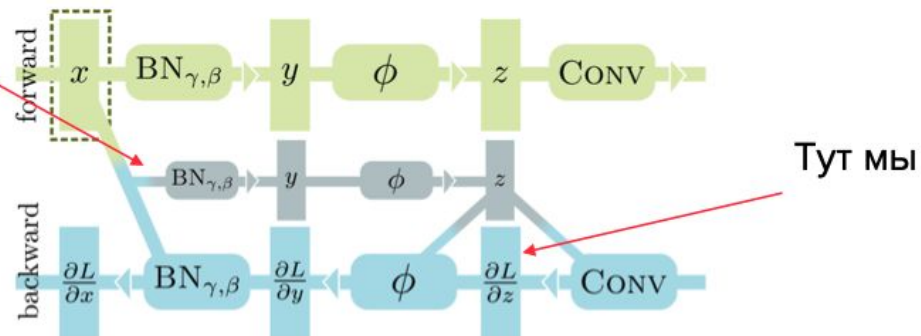
## Pre-2: Vanilla BN + checkpoints



- Если после forward pass буферизовать только  $x$ , то по сохраненным статистикам **BN** можно вычислить заново  $y$  и затем  $z$
- Меньше памяти, больше вычислений
- Нелокальность вычислений
- [Pytorch Checkpointing Doc](#)

## Pre-2: Vanilla BN + checkpoints

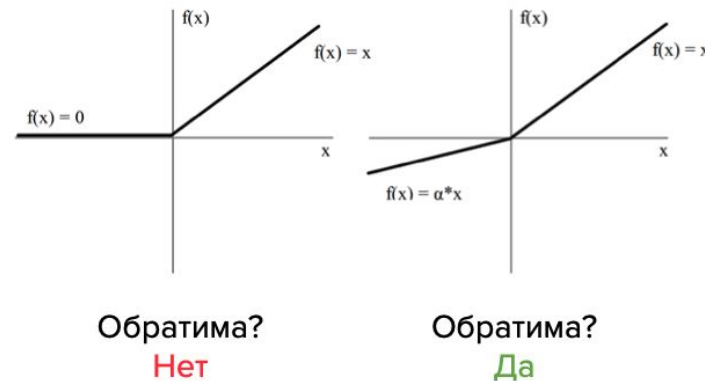
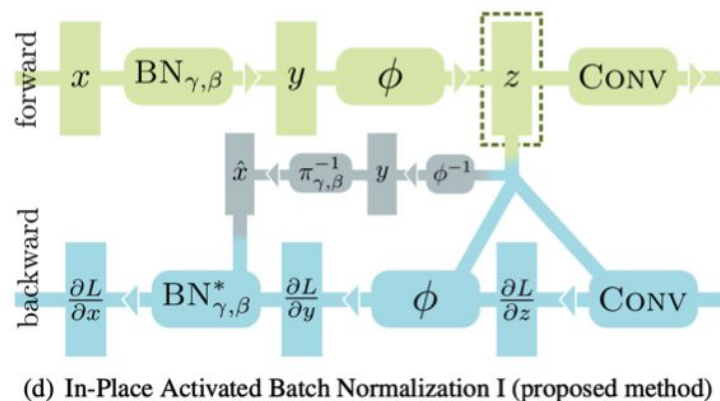
А тут необходимые  
нам вычисления



(b) Checkpointing [4, 21]

- Если после forward pass буферизовать только **x**, то по сохраненным статистикам **BN** можно вычислить заново **y** и затем **z**
- Меньше памяти, больше вычислений
- **Нелокальность вычислений**
- [Pytorch Checkpointing Doc](#)

# Inplace-Activated BN



- Будем использовать только обратимые активации
- Тогда можно при forward pass **хранить только  $z$**
- **Вычисления теперь локальны**

# Inplace-Activated BN

- Заявленная авторами экономия памяти GPU: до 50%
- Можно [комбинировать с SyncBN](#)

# Итоги



# Итоги

- Сегментация нужна для чёткого выделения границ объектов
  - Разница между Semantic и Instance - в разделении объектов одного класса
  - U-Net как хороший старт в Semantic Segmentation
  - Для Instance Segmentation — Mask R-CNN
- Для моделей локализации могут требоваться хитрые оптимизации по памяти
  - Чекпоинты
  - Inplace-ABN

## Вопросы на самопроверку

- Пусть стоит задача семантической сегментации на  $C$  классов
  - Какой будет размерность выходного тензора?
  - Как будет выглядеть лосс в этом случае?
- Почему Focal Loss может помочь с проблемой дисбаланса классов?
- Почему малый размер батча может плохо повлиять на модель, включающую слои BN?