Segmentation + object detection

“Машинное обучение”, лекция 5

Константин Архипенко

24 октября 2019 г.

ИСП РАН

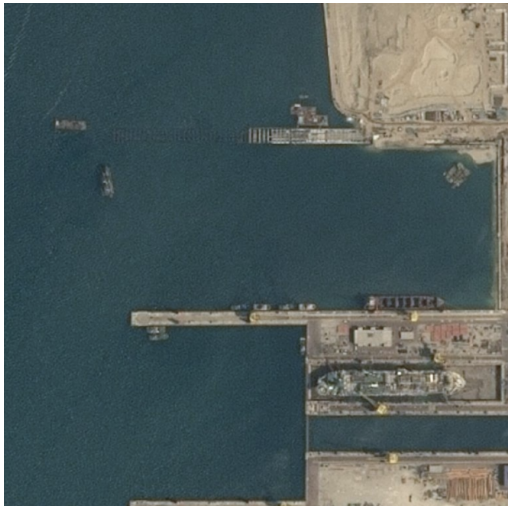
Semantic segmentation — классификация пикселей

Вопросы

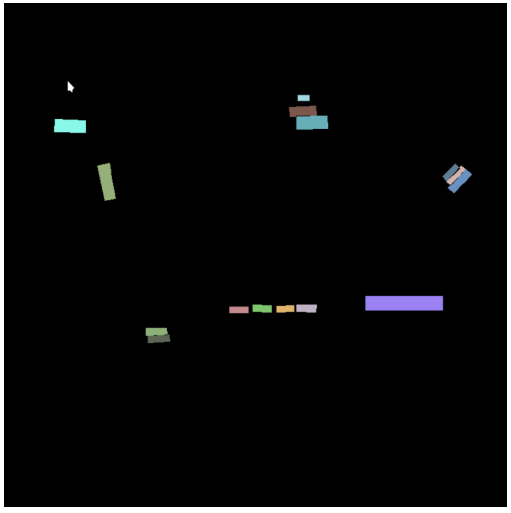
- Какую функцию потерь использовать?
- Как оценивать качество?

Оценка качества сегментации: pixel accuracy

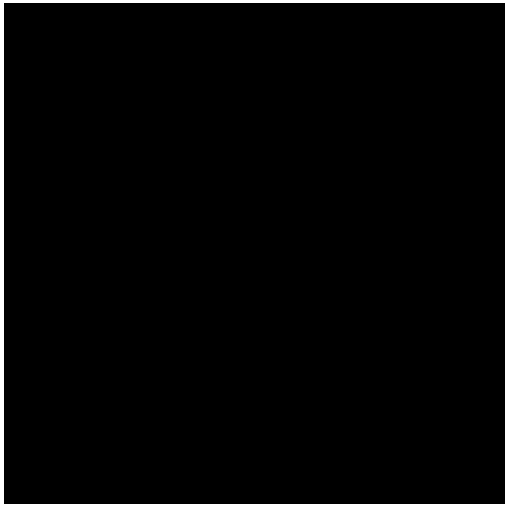
Вход



Ground truth



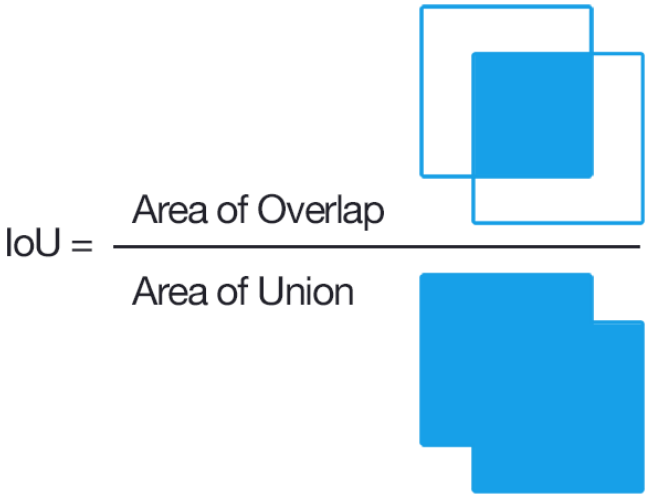
Предсказание



Получаем *accuracy* $= 0.95$

- Нужно что-то лучше

Mean IoU (intersection over union)



IoU

- Mean — среднее по всем классам
- Каким получится mean IoU для последнего примера? (там 2 класса — «фон» и «корабль»)

Функция потерь для сегментации

Кросс-энтропия vs IoU

- Первая более пригодна для градиентной оптимизации
- Но дисбаланс классов никуда не денется, поэтому взвешиваем потери (*только один* $y_c \neq 0$):

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_c \alpha_c \cdot y_c \log \hat{y}_c$$

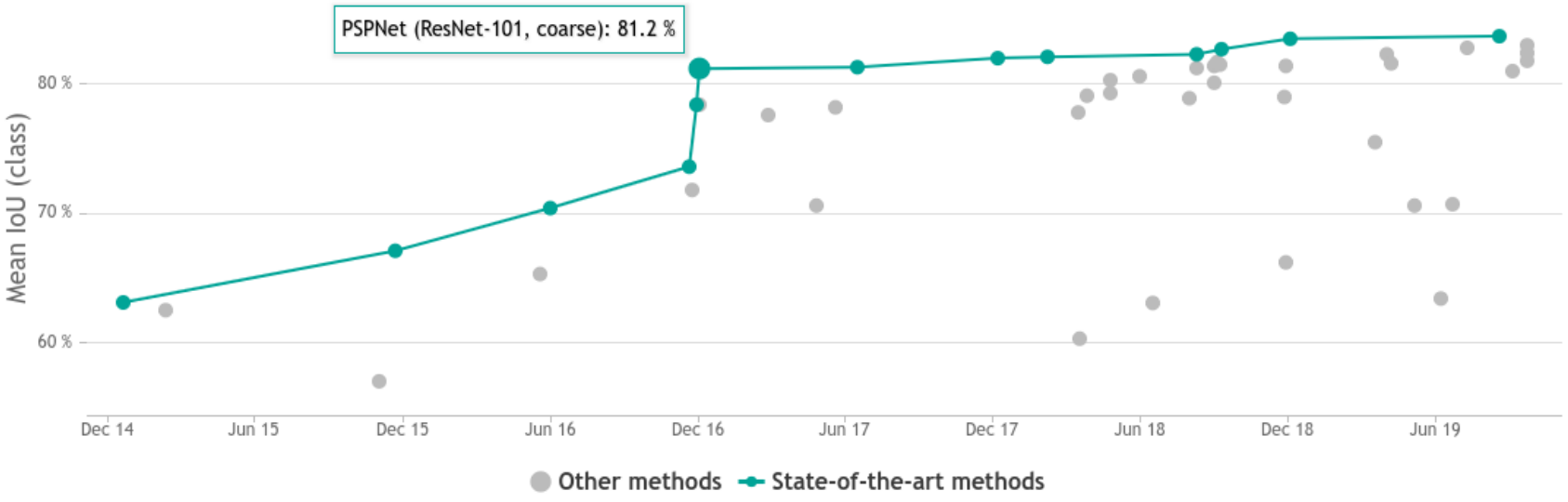
- **Focal loss** — фокусируемся на трудных примерах:

$$FL(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_c (1 - \hat{y}_c)^\gamma \alpha_c \cdot y_c \log \hat{y}_c$$

SOTA

paperswithcode.com/sota/semantic-segmentation-on-cityscapes

Semantic Segmentation on Cityscapes test

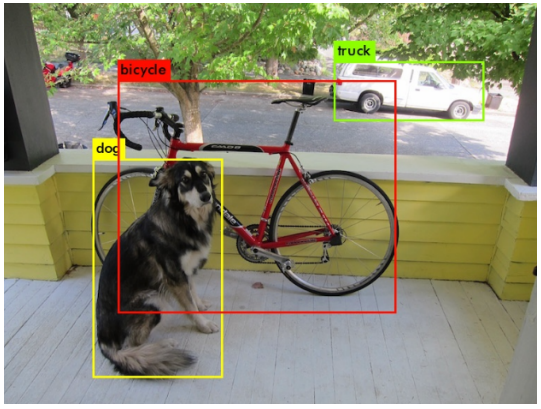


View: All methods

Edit

RANK	METHOD	MEAN IOU (CLASS)	EXTRA TRAINING DATA	PAPER TITLE	YEAR	PAPER	CODE
1	HRNetV2 + OCR (w/ ASP)	83.7%	✓	Deep High-Resolution Representation Learning for Visual Recognition	2019		
2	DeepLabV3Plus + SDCNetAug	83.5%	✓	Improving Semantic Segmentation via Video Propagation and Label Relaxation	2018		

Object detection

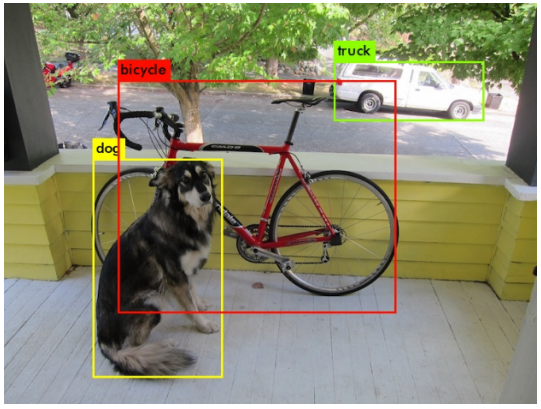


Оценка качества (наивная)

Для каждого класса *class*:

- *true positive (TP)* — предсказан bbox с классом *class* и *IoU* $\geq$ *threshold* с (одним из) ground truth
- *false positive (FP)* — предсказан bbox с классом *class*, не являющийся *TP*
- *false negative (FN)* — для ground truth bbox не нашлось предсказанного *TP*

Object detection

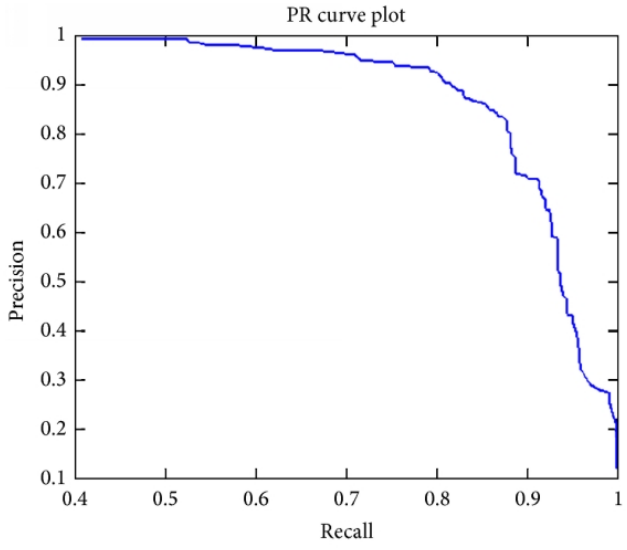


А если много bbox с хорошим IoU для одного и того же GT?

COCO dataset:

- Сортируем предсказанные bbox по *confidence* и смотрим на первые *maxDet*
- Для каждого предсказания: матчим его с GT с наибольшим IoU (при условии $\geq$ *threshold*)
- Unmatched предсказания попадают в *FP*, а GT — в *FN*

AP и mAP



Average precision

- Может ли PR кривая возрастать? (ответ: да)
- Усредняем для разных IoU *threshold*:

$$AP = \frac{1}{10} \sum_{threshold=0.5,0.55,\ldots,0.95} \frac{TP_{threshold}}{TP_{threshold} + FP_{threshold}}$$

AP и mAP

mAP

- Mean — среднее по всем классам
- В статьях и лидербордах букву m опускают

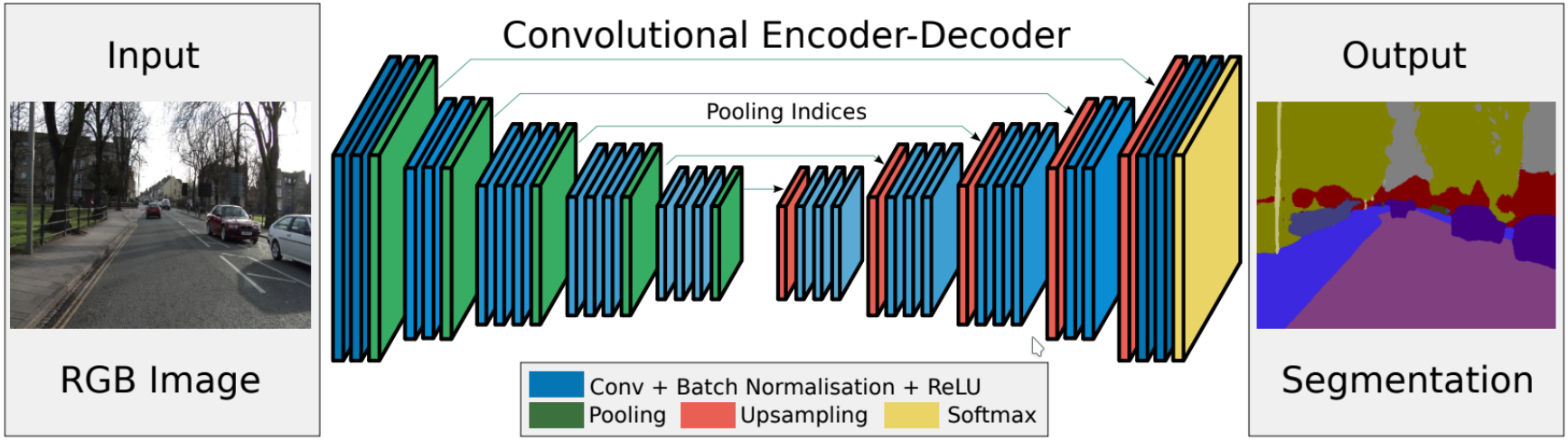
Похожие метрики

- AP_{50}, AP_{75}: точность для конкретного *threshold* (буква A здесь сбивает с толку — усреднения нет)
- AP_S, AP_M, AP_L: считаем только для маленьких/средних/больших объектов

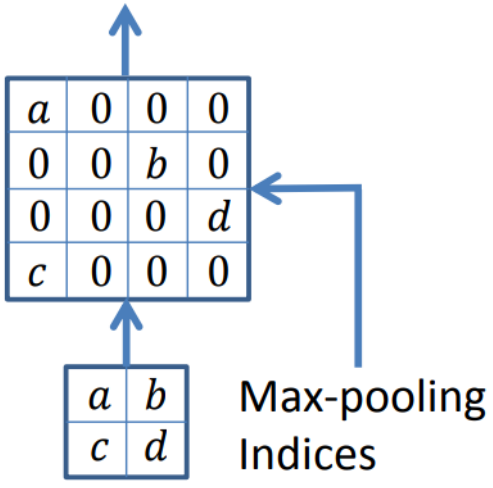
А какую взять функцию потерь для обучения?

- Свои для разных моделей, скоро рассмотрим
- Но начнем с сегментации

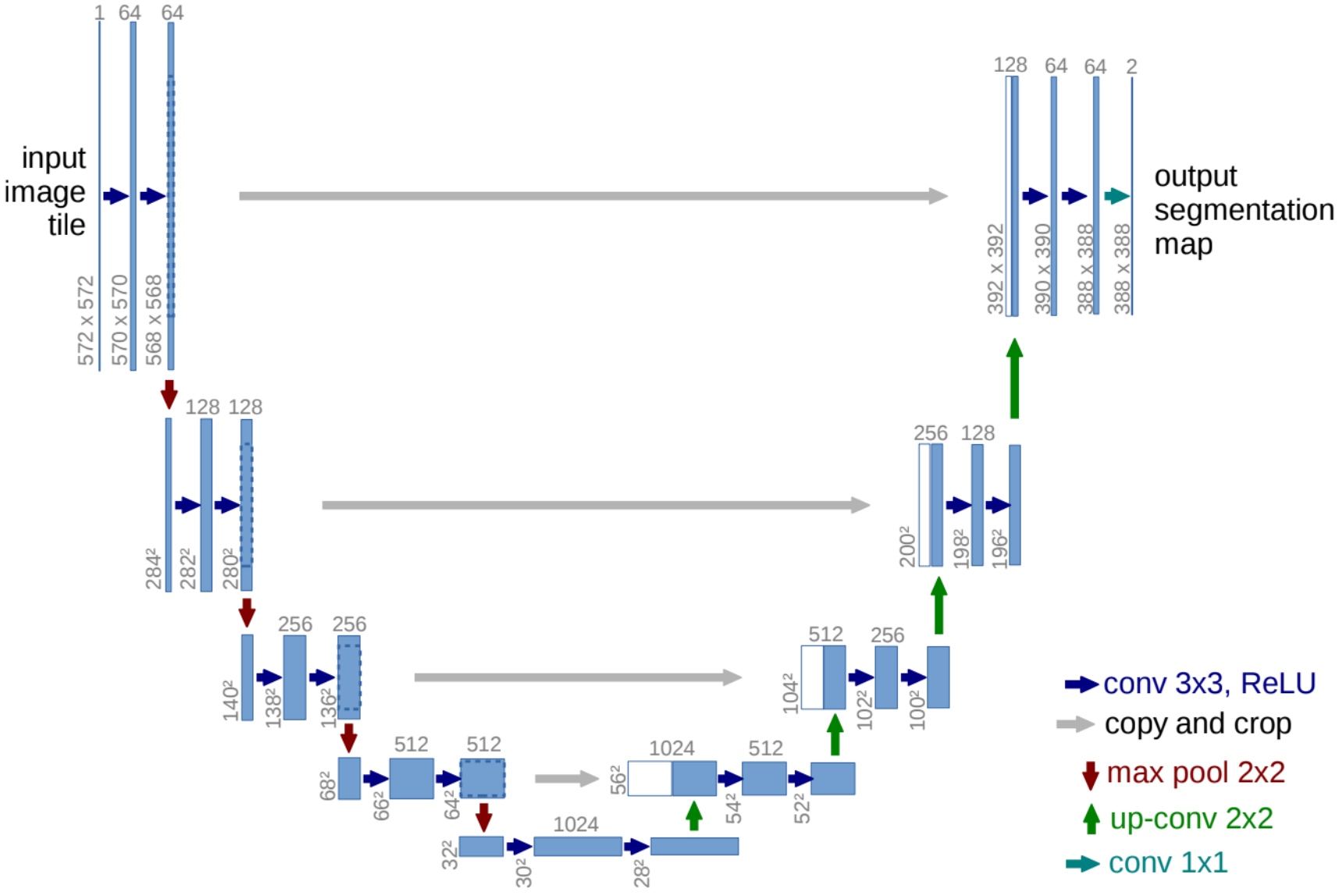
SegNet (Badrinarayanan et al., 2015)



- Энкодер — сверточная часть VGG16
- Нет dense слоев
- Unpooling: запоминаем индексы максимумов в pooling энкодера, переносим в декодер
- Просто, не очень качественно



U-Net (Ronneberger et al., 2015)

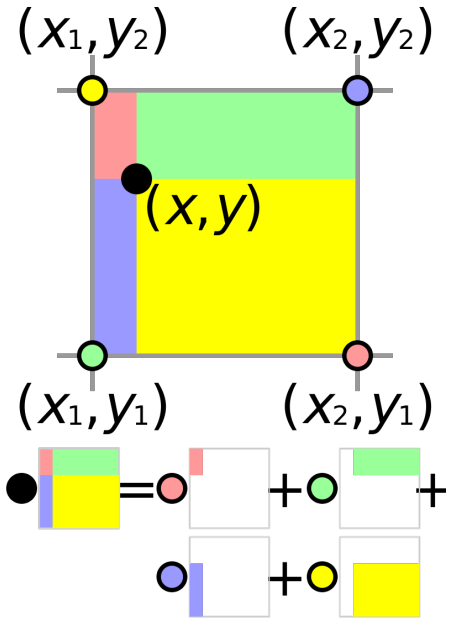


U-Net: upsampling

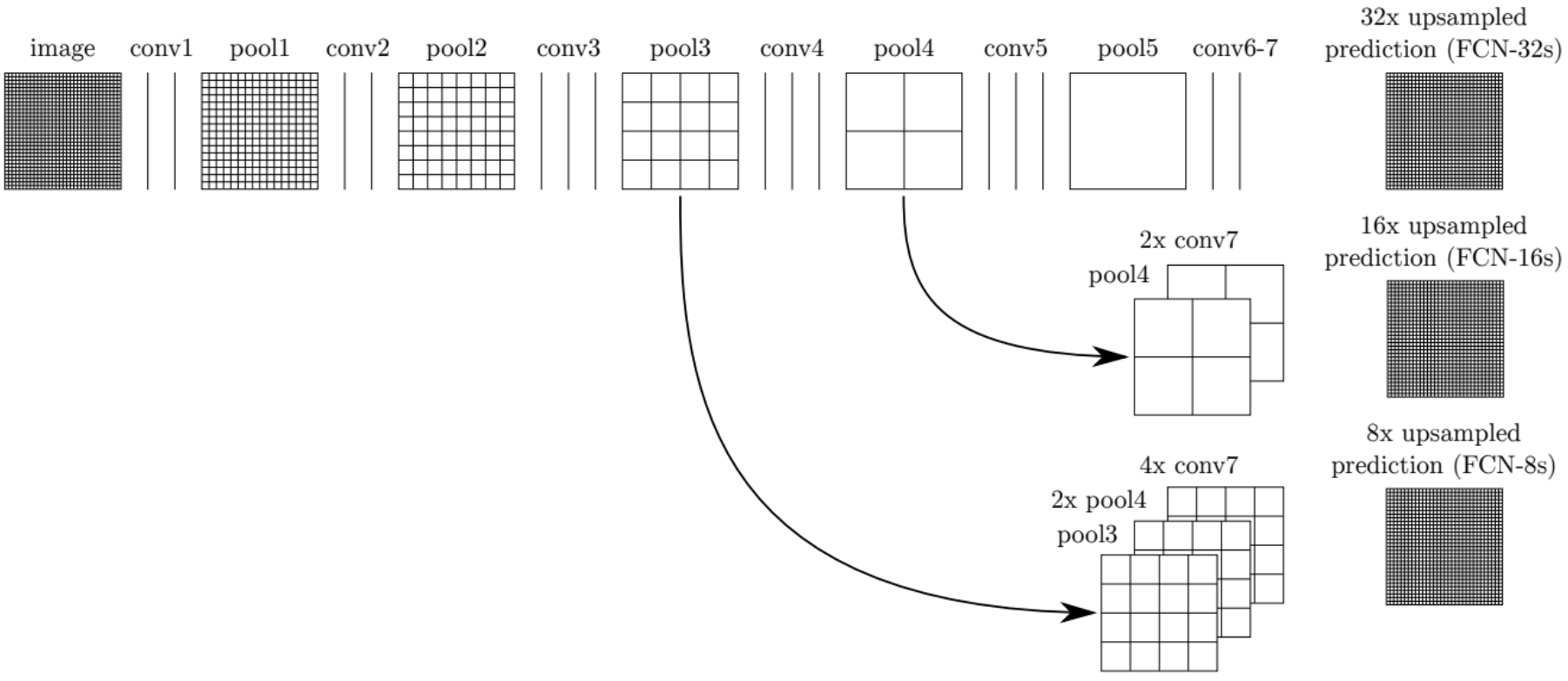
- up-conv — *bilinear upsampling* с последующей сверткой
- В декодере: конкатенация с feature maps энкодера
- Выход меньше входа $\Longrightarrow$ overlap-tile strategy

```
>>> input = torch.arange(1, 5, dtype=torch.float32).view(1, 1, 2, 2)
>>> input
tensor([[[[ 1.,  2.],
          [ 3.,  4.]]]])

>>> m = nn.UpsamplingBilinear2d(scale_factor=2)
>>> m(input)
tensor([[[[ 1.0000,  1.3333,  1.6667,  2.0000],
          [ 1.6667,  2.0000,  2.3333,  2.6667],
          [ 2.3333,  2.6667,  3.0000,  3.3333],
          [ 3.0000,  3.3333,  3.6667,  4.0000]]]])
```



FCN for Semantic Segmentation (Long et al., 2014)



- Как и SegNet, основана на VGG16 ($conv_1, \ldots, pool_5, fc_6, fc_7, fc_8$)
- В VGG16 все свертки 3×3 с same padding, выход $pool_5$ имеет размер $(512, 7, 7)$ (вход — $(3, 224, 224)$), и перед fc_6 превращается в вектор $(25088,)$

FCN

Fully convolutional

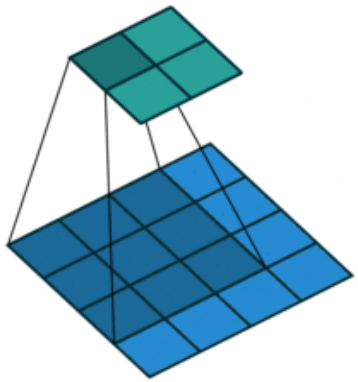
- Слой $Linear(inputSize, outputSize)$ можно представить как $Conv2d(inChannels = inputSize, outChannels = outputSize, kernelSize = 1)$ с valid padding

Модифицир уем VGG16

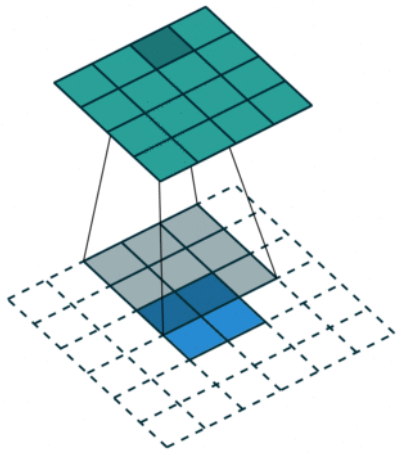
- Убираем flatten
- $fc_6 \to conv_6(512, 4096, 7)$
- $fc_7 \to conv_7(4096, 4096, 1)$
- $fc_8 \to conv_8(4096, numClasses, 1)$
- Можно обрабатывать картинки произвольного размера!

FCN for segmentation: обучаемый upsampling

Conv2d



ConvTranspose2d



- Выход $pool_5$ в 32 раза меньше картинки на входе
- Сделаем такой большой padding в $conv_1$ (100px), чтобы выход у $pool_5$ был $(512, 13, 13)$, тогда у $conv_6$ — $(4096, 7, 7)$
- Выход $conv_8$ размером $(numClasses, 7, 7)$ **увеличим в 32 раза** с помощью ConvTranspose2d — получим карту сегментации (FCN-32s)

FCN-32s

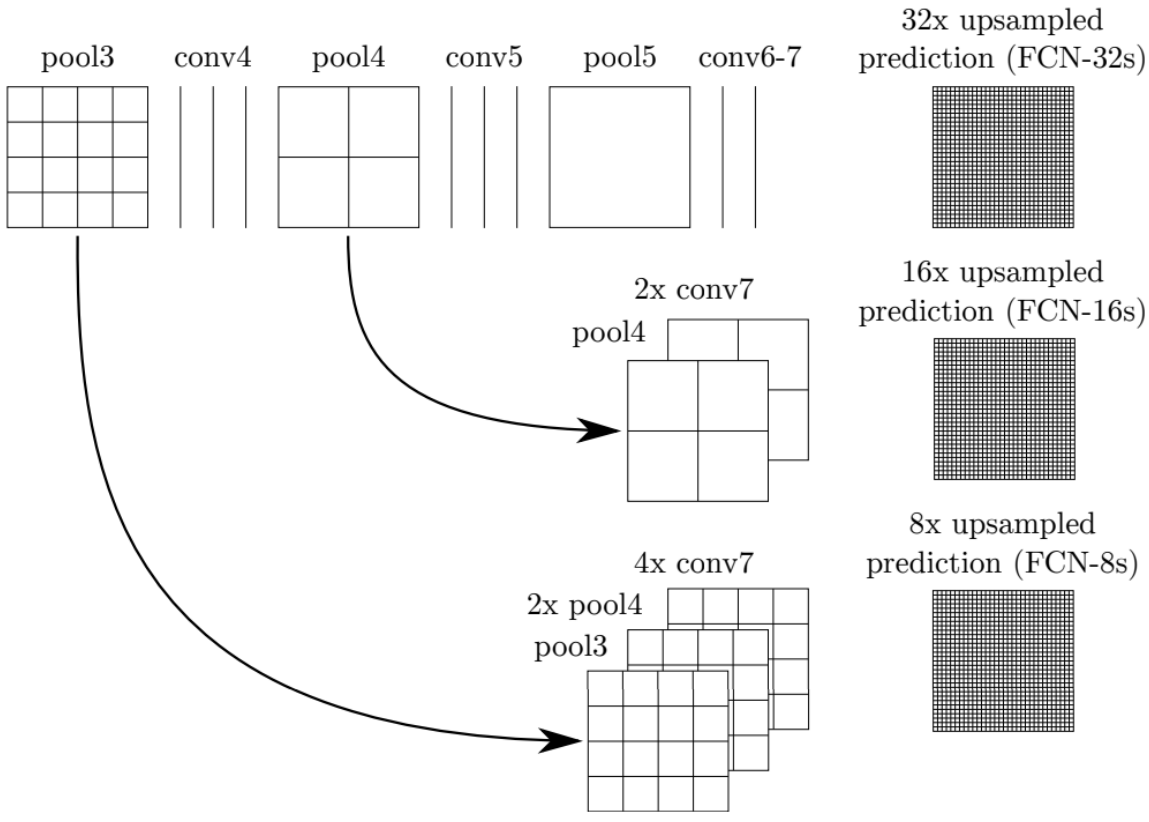
Увеличение с помощью ConvTranspose2d

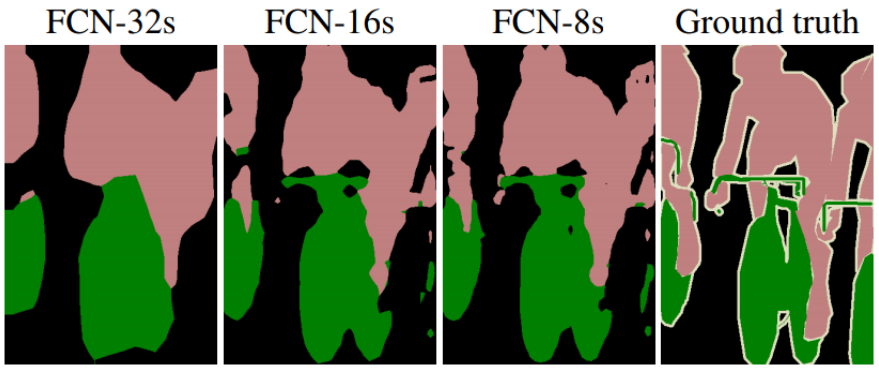
- Берем *stride*, равный кратности увеличения, а *kernelSize* — в 2 раза больше

Недостатки

- Слишком низкое разрешение feature map после $pool_5$
  $\Longrightarrow$ слишком грубая сегментация
- Решение — можно переиспользовать выходы промежуточных слоев ($pool_3, pool_4$), тоже их увеличивая транспонированной сверткой
  - Получим модели FCN-16s и FCN-8s, они более тяжеловесные и качественные

Варианты FCN





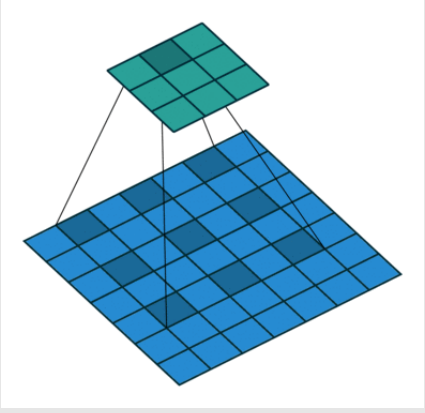
Dilated convolutions

Проблемы

- Ранние свертки видят мало контекста
- Поздние свертки видят низкое разрешение

Решение

- $rate = 2$:



- Используются в **DeepLab**v1...v3 (Chen et al.)

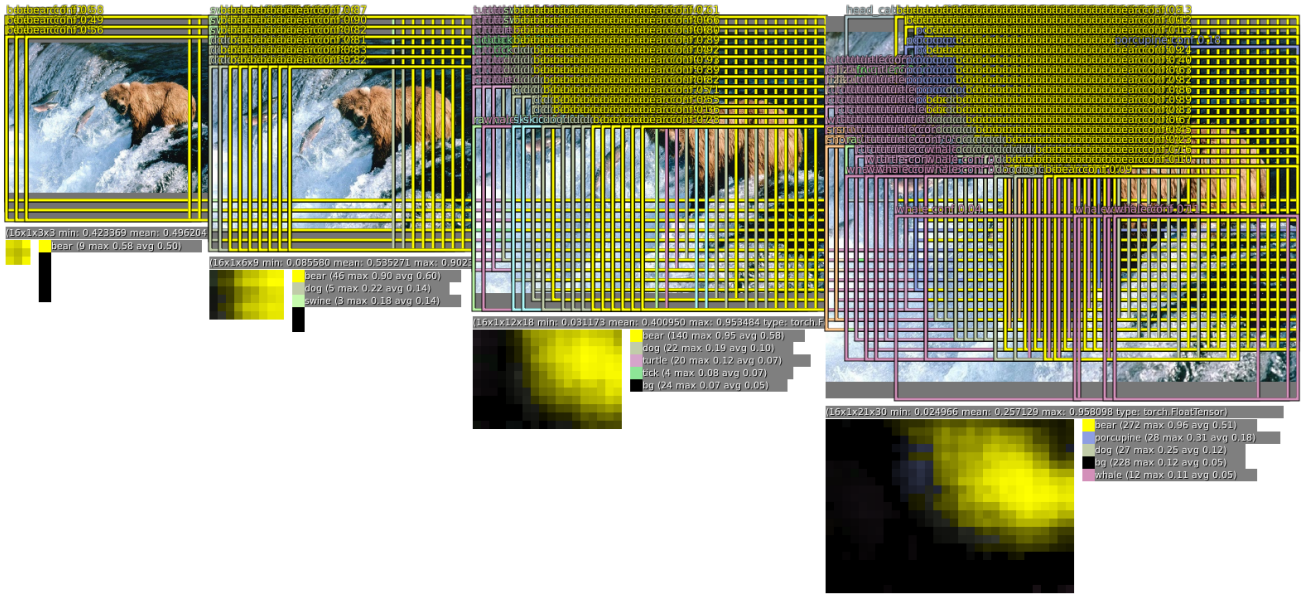
Другие SOTA: PSPNet, HRNetV2, ...

Вернемся к object detection

One-stage detection

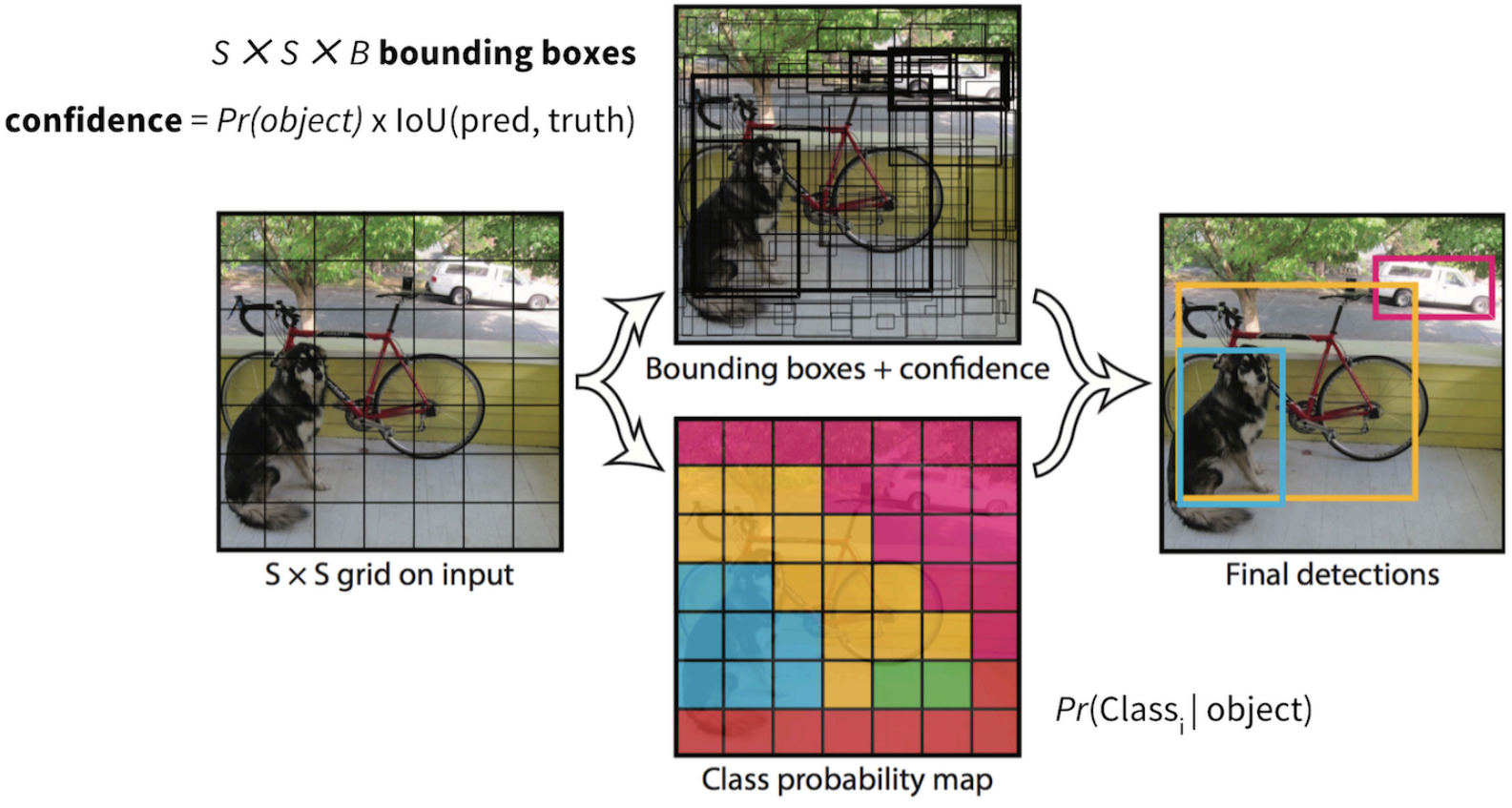
- Разбиваем картинку на фрагменты
  - Без перекрытия (сетка)
  - С перекрытием (скользящее окно с шагом *step*)
- Предполагаем, что фрагмент содержит *не более одного* объекта
- Подаем фрагмент:
  - классификатору — вероятности классов
  - регрессору — предсказывает bbox (или фиксированное количество bbox) и его *confidence*
- Примеры: OverFeat, YOLO, SSD, ...

OverFeat (Sermanet et al., 2013)



- Обучение на фрагментах 221×221
- Inference: бьем несколько версий картинки (multi-scale) на фрагменты, предсказываем для фрагмента класс и bbox
- Выполняем слияние предсказанных bbox по простому алгоритму

YOLO (Redmon et al., 2015)



- Как получить финальные предсказания?

Non-maximum suppression

Input : $\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, N_t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq$ *empty* **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
      end NMS
      $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$ Soft-NMS
    end
  end
  return $\mathcal{D}, \mathcal{S}$

Чего сегодня не было

Two-stage detection

- Первый этап — region proposal
- R-CNN, Fast[er] R-CNN, Mask R-CNN

Другие модели

- Продолжение про сегментацию (включая текущий SOTA)
- Другие задачи (pose estimation, text detection & recognition, видео, ...)

Лекция 6

- Quantization, ускорение инференса