Машинное обучение (Machine Learning)

Сегментация изображений

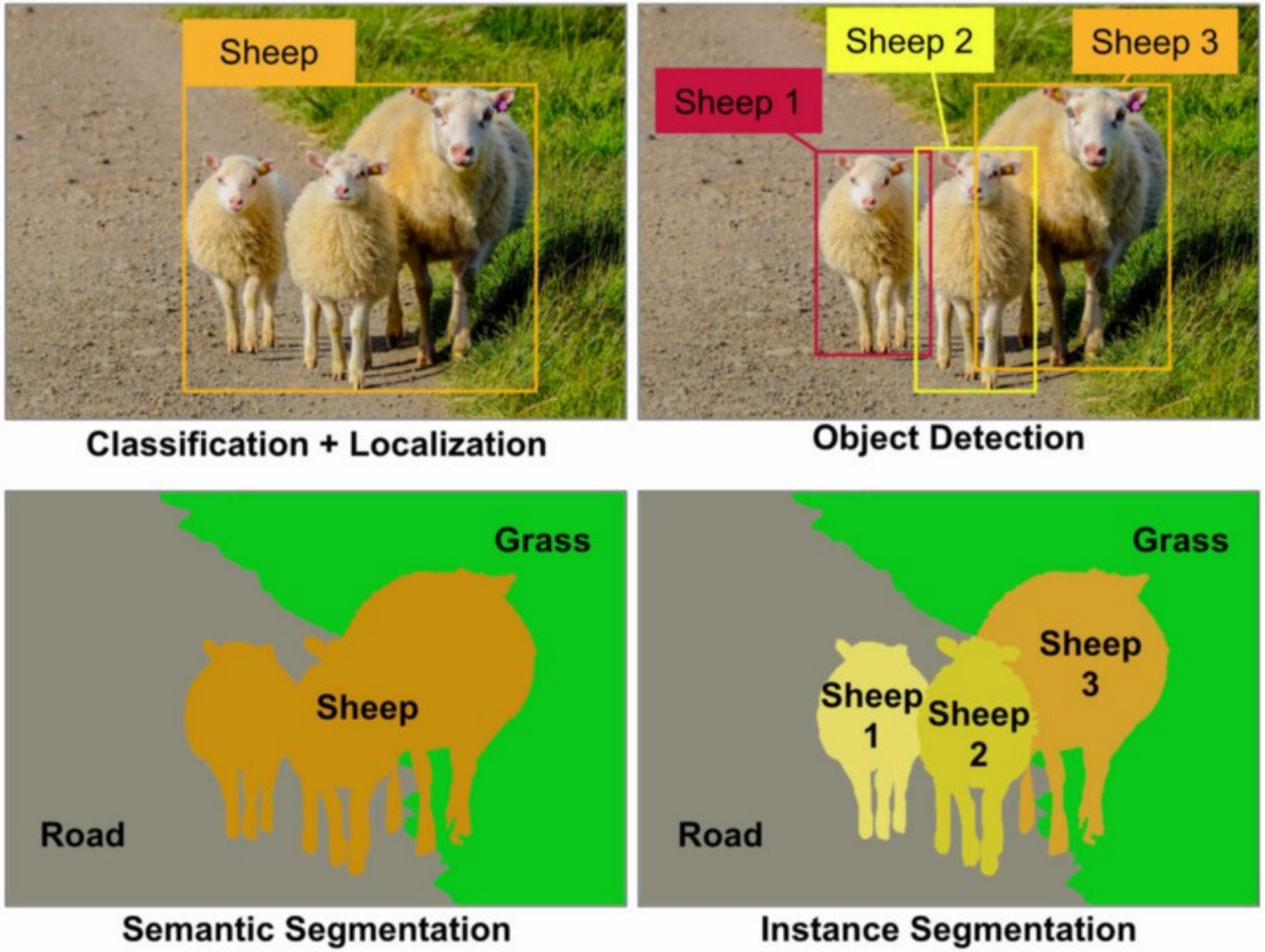
Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого

Типы сегментации

- Semantic segmentation
- Classification and localization
- Object detection
- Instance segmentation

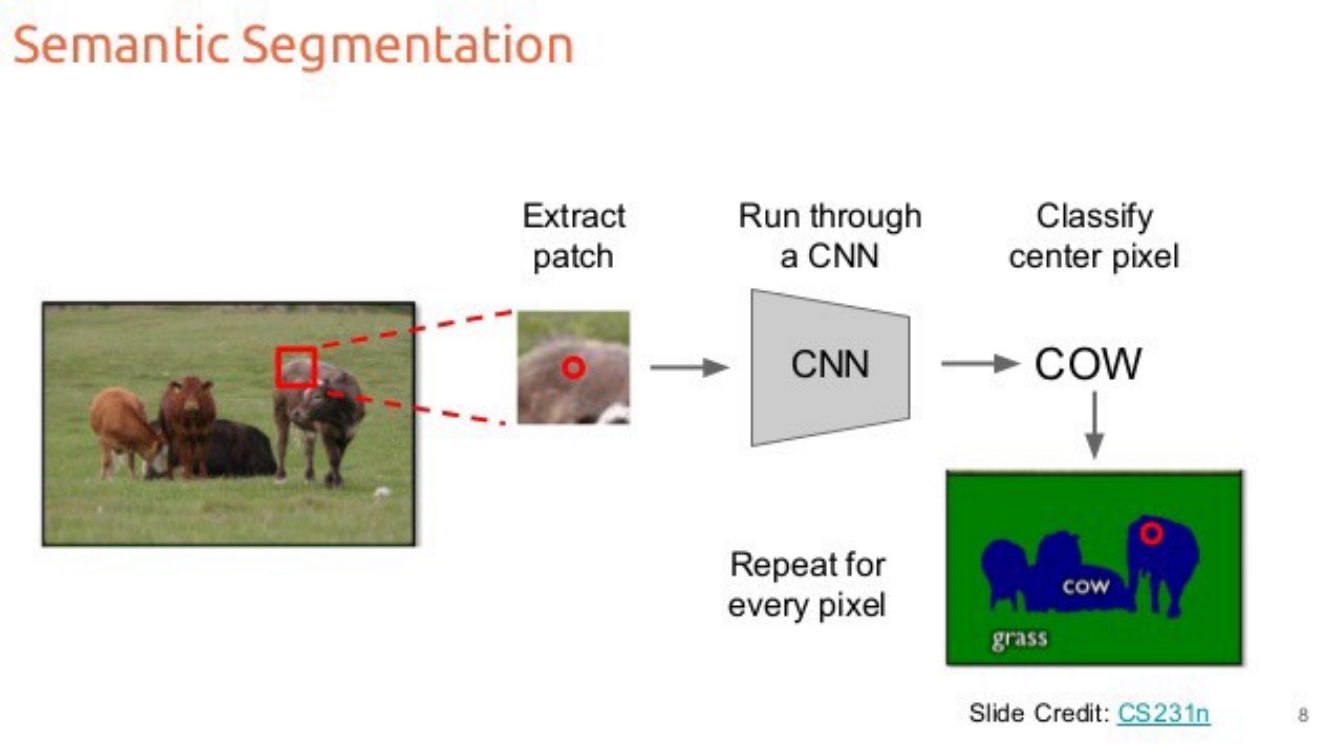
Типы сегментации



Semantic segmentation (1)

- Есть изображение и выход - классификация каждого отдельного пикселя
- Например, все пиксели с изображением овец будут отнесены к одной категории, как и пиксели с травой и дорогой
- Один из возможных подходов к решению - рассматривать как задачу классификации со скользящим окном
- Исходное изображение разбивается на несколько фрагментов одинакового размера
- Каждый фрагмент — в CNN, чтобы классифицировать

Semantic segmentation (2)



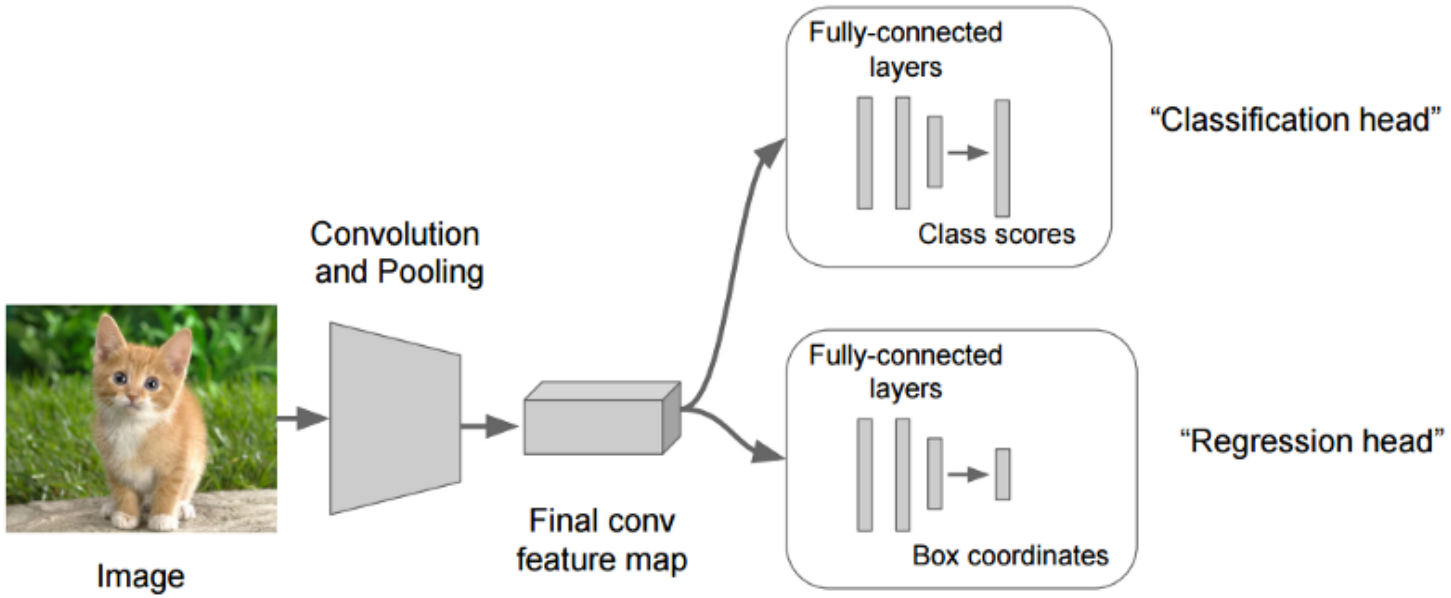
Классификация и локализация (Classification and localization)

- Классификация ставит метку, но иногда, интересует расположение этого объекта на изображении.
- Можно нарисовать ограничивающую рамку вокруг объекта на изображении, чтобы локализовать его
- Как?

Классификация и локализация (1)

- Сначала можно "скормить"входное изображение какой-нибудь гигантской ConvNet, которая выдаст веса по каждой категории
- Теперь есть еще один полносвязный слой, который предсказывает координаты ограничивающей рамки для объекта (координаты центра x, y, а также высоту и ширину) из карты объектов, созданной более ранними слоями
- Таким образом, сетка выдаст два выхода: один соответствует классу изображения, а другой — ограничивающей рамке
- Для обучения сети две функции потерь: потери (энтропия) для классификации и потери L1/L2 для предсказаний ограничительной рамки

Классификация и локализация (2)



Object Detection (1)

- Идея обнаружения объектов: начинаем с некоторого фиксир. набора интересующих нас категорий
- Каждый раз, когда любая из этих категорий появляется на входном изображении, мы рисуем ограничивающую рамку вокруг этого изображения вместе с предсказанием метки класса
- Отличие от классиф-ии и локал-ии в том, что в detection классифицируют и рисуют рамку только вокруг **одного** объекта
- В класс. и локализ. не известно заранее, сколько объектов ожидать на изображении.

Object Detection - Region Proposals Based Algorithms (RPBA)

- RPBA даст много рамок, в которых может присутствовать объект. На выходе возможен шум в виде рамок, в которых нет объектов. Но объект будет выбран в качестве рамки-кандидата.
- Чтобы сделать все рамки-кандидаты одинакового размера, нам нужно деформировать их до некоторого фиксированного размера квадрата, который можно в конечном итоге передать в сеть.
- Затем применяют большую ConvNet к каждой рамке-кандидату для классификации.

Object Detection - Region Proposals Based Algorithms (RPBA)

- В этом вся идея R-CNN. Для снижения сложности используются быстрые R-CNN:
  - сначала получить карту объектов с высоким разрешением, передав входное изображение через ConvNet,
  - затем наложить эти Region Proposals на карту объектов вместо фактического изображения.
- Это позволяет повторно использовать вычисления свертки по всему изображению, когда много обрезков.

Object Detection - YOLO (You only look once)

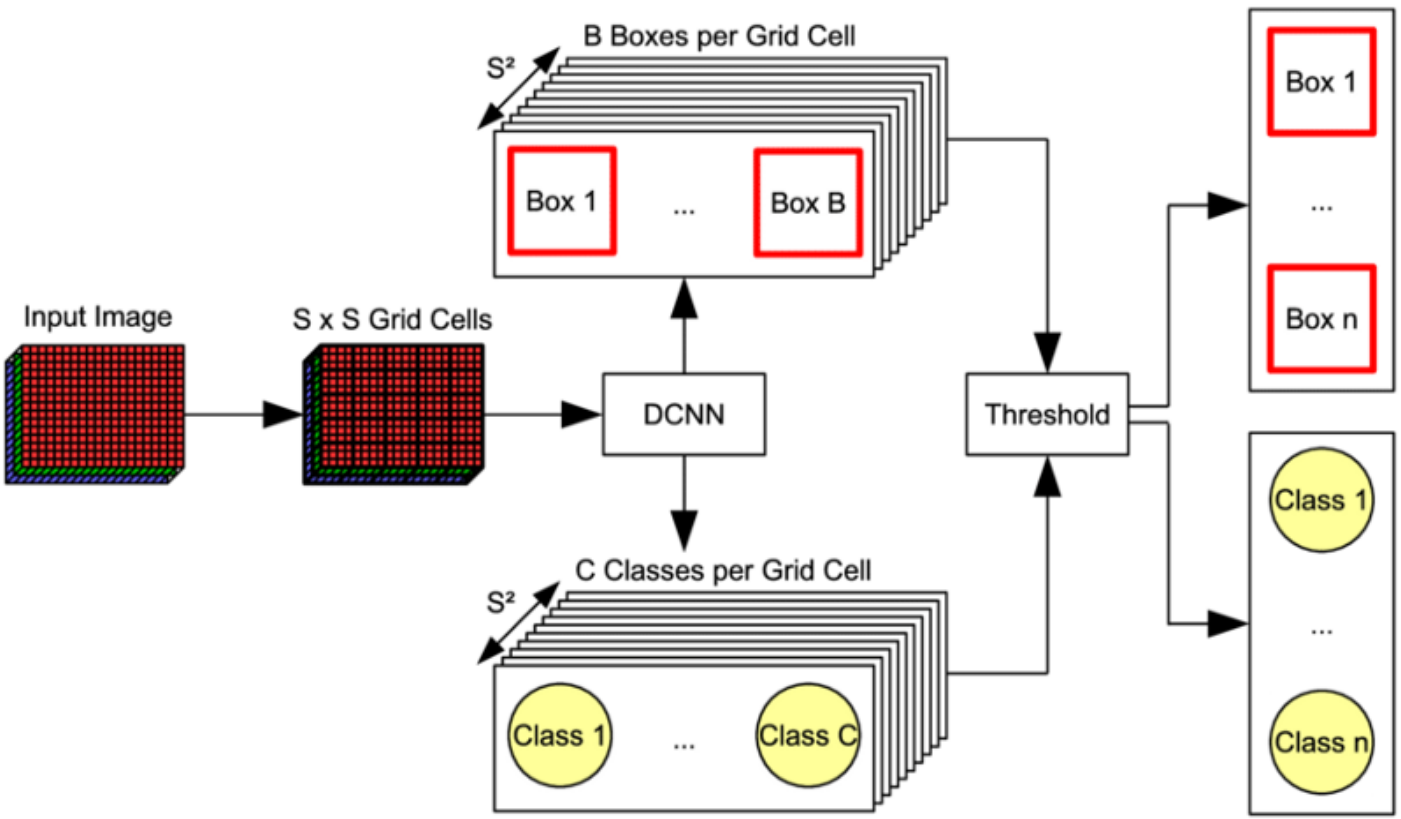
- YOLO (You only look once)
- Принцип работы YOLO подразумевает ввод сразу всего изображения, которое проходит через сверточную нейронную сеть только один раз. Именно поэтому он называется “You only look once”.
- Идея YOLO: вместо независимой обработки предложенных областей делать все предсказания одновременно, переформулируя их как единую задачу регрессии, прямо от пикселей изображения до координат ограничивающей рамки и вероятностей классов.

Object Detection - YOLO (You only look once)

- Сначала делим все входное изображение на сетку $S \times S$.
- Каждая ячейка сетки предсказывает C вероятностей условного класса (Pr(Class|Object)) вместе с B ограничивающими рамками (x, y, w, h), каждая из которых имеет показатель достоверности.
- Координаты (x, y) представляют центр прямоугольника относительно границ ячейки сетки, тогда как ширина w и высота h предсказываются относительно всего изображения.
- Вероятности зависят от ячейки сетки, содержащей объект. Прогнозируется только один набор вероятностей класса на ячейку сетки, независимо от количества рамок B.

Object Detection - YOLO

YOLO (You only look once)



Object Detection - YOLO (You only look once)

- Показатели достоверности отражают, насколько модель уверена в том, что рамка содержит объект. Если в рамке нет объектов, то показатель достоверности должен быть равен нулю. С другой стороны, показатель достоверности должен быть таким же, как пересечение по объединению (IOU) между предсказанной рамкой и меткой класса.
- Confidence score = Pr(Object) * IOU

Object Detection - YOLO (You only look once)

- Во время тестирования условные вероятности класса умножаются на предсказанные достоверности отдельных рамок, что дает оценки достоверности для каждого класса для каждой рамки. Эти оценки кодируют как вероятность того, что этот класс появится в рамке, так и то, насколько хорошо предсказанная рамка соответствует объекту.
- Pr(Class | Object) * (Pr(Object) * IOU) = Pr(Class) * IOU

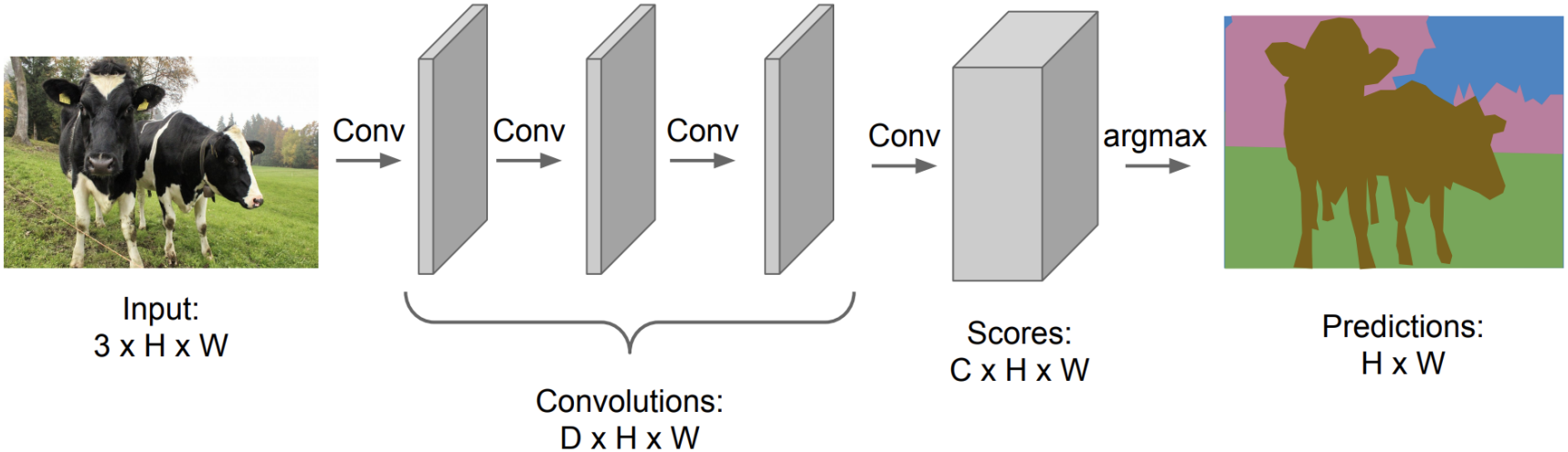
Semantic Segmentation, классификация на уровне пикселов

- Просто соединить набор сверточных слоев, где захватываются локальные элементы в изображениях.
- CNN может кодировать изображение как компактное представление его содержимого
- Отображение между входным изобр. и соответствующим выходом сегментации через иерархическое представление

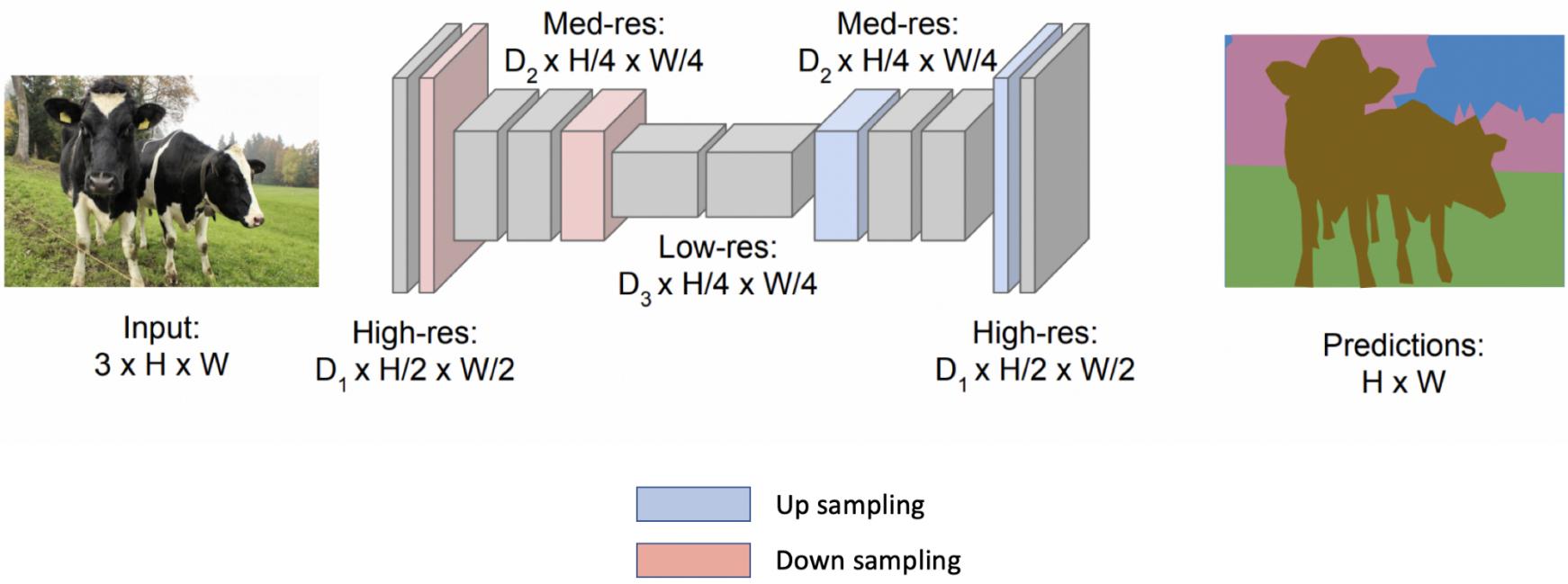
Patch-by-patch scanning - Основная идея

- Метка класса определяется для каждого пиксела: задача рассматривается как задача “попикселной” классификации
- Окно (patch) с центром в каждом пикселе подается на вход сверточной сети для генерации его класса
- Точность сегментации повышается с увеличением размера окна, так как оно охватывает больше контекстной информации
- Главный недостаток - большое перекрытие соседних окон и большой объем избыточных вычислений

Semantic Segmentation, классификация на уровне пикселов

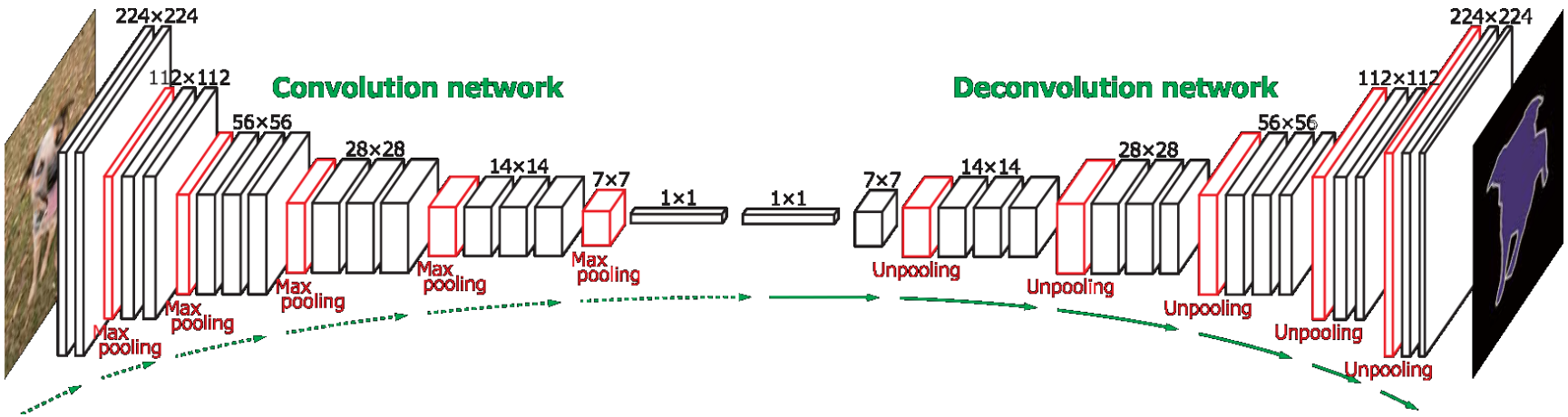


Semantic Segmentation, классификация на уровне пикселов



Deconvolution network (Deconvnet)

H. Noh S. Hong B. Han, Learning Deconvolution Network for Semantic Segmentation



Deconvolution network

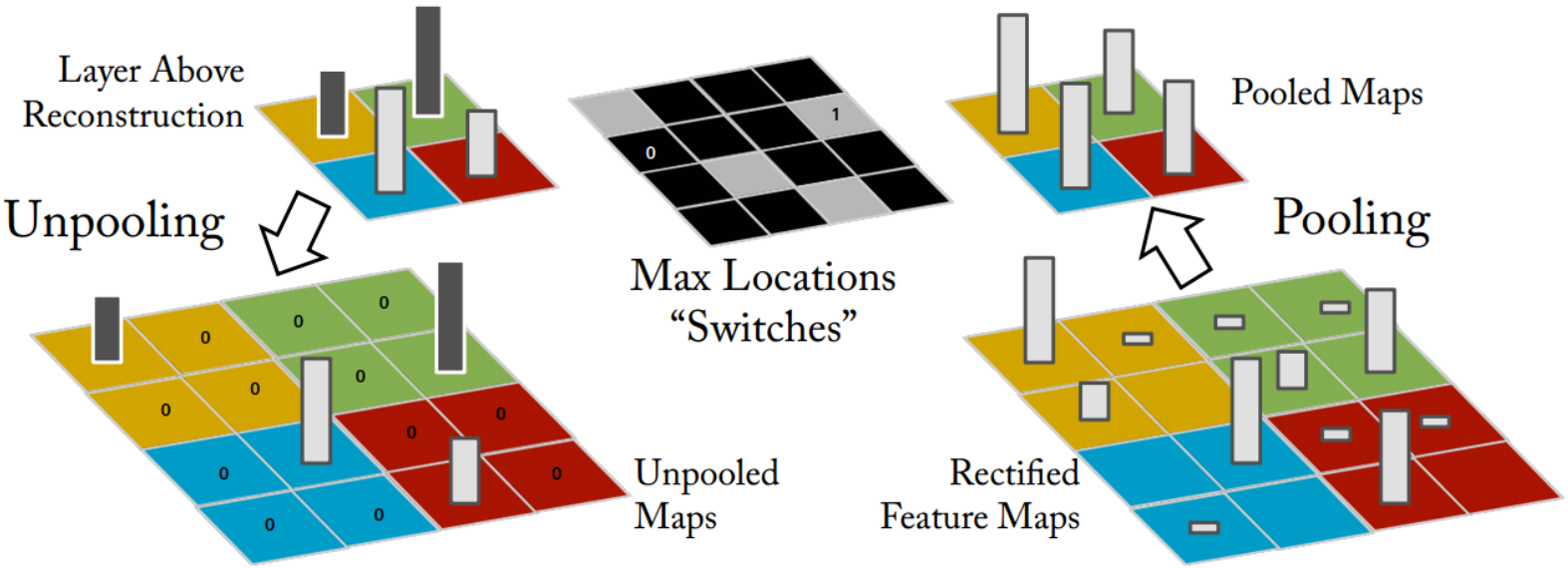
Реализует две основные процедуры

1. Unpooling
2. Deconvolution

Повышающая дискретизация (unpooling)

Повышающую дискретизацию (unpooling, upsampling) можно рассматривать как операцию, обратную понижающей дискретизации, или субдискретизации (pooling, downsampling).

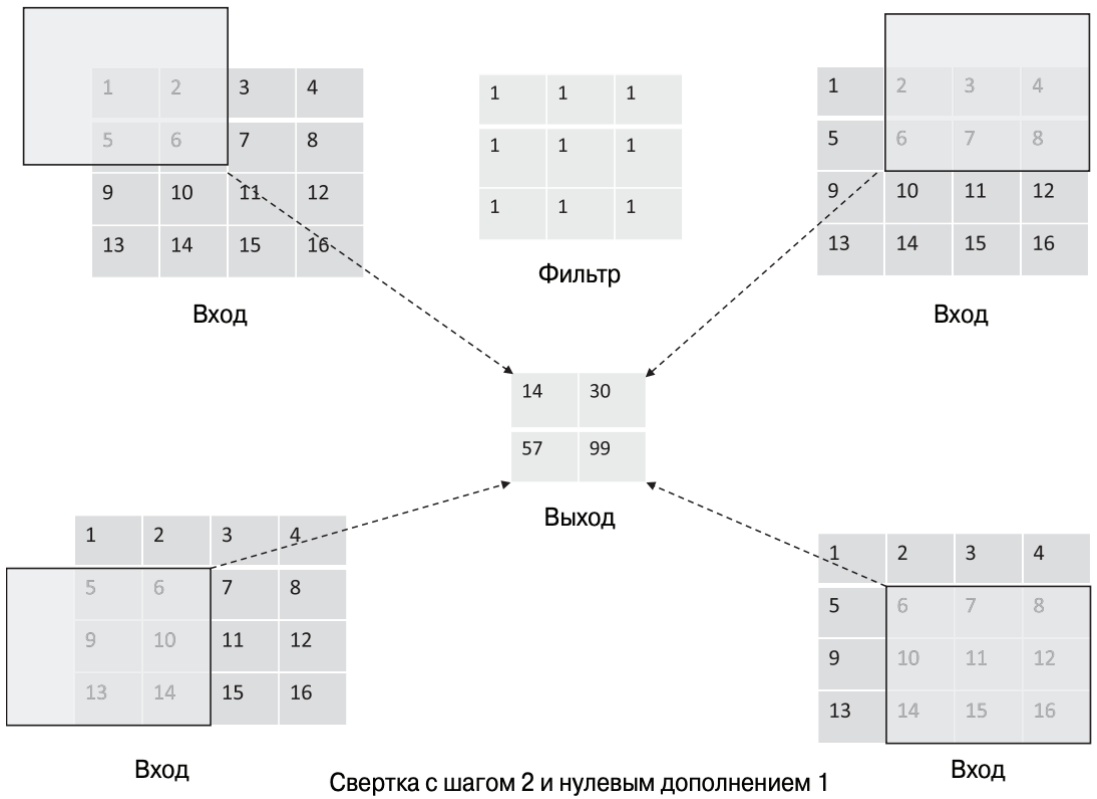
Unpooling (upsampling)



Deconvolution

- Deconvolution слой layers уплотняет рареженный слой, полученный в unpooling, с ипользованием операций свертки
- В отличие от сверточных слоев, которые соединяют множественные входные значения активации нейронов внутри окна фильтра в одно значение, обратная операция свртки ассоциирует одиночное входное значение активации с множественным выходом

Convolution



Свертка с шагом 2 и нулевым дополнением 1

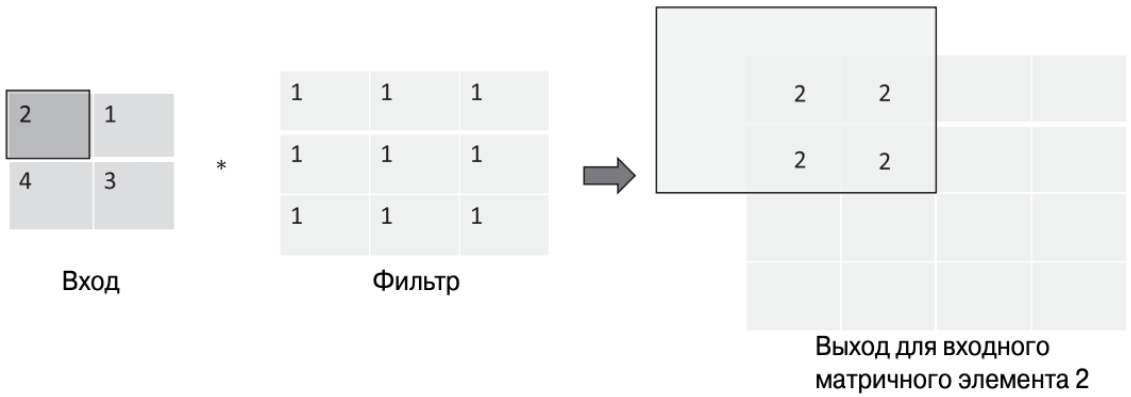
Deconvolution

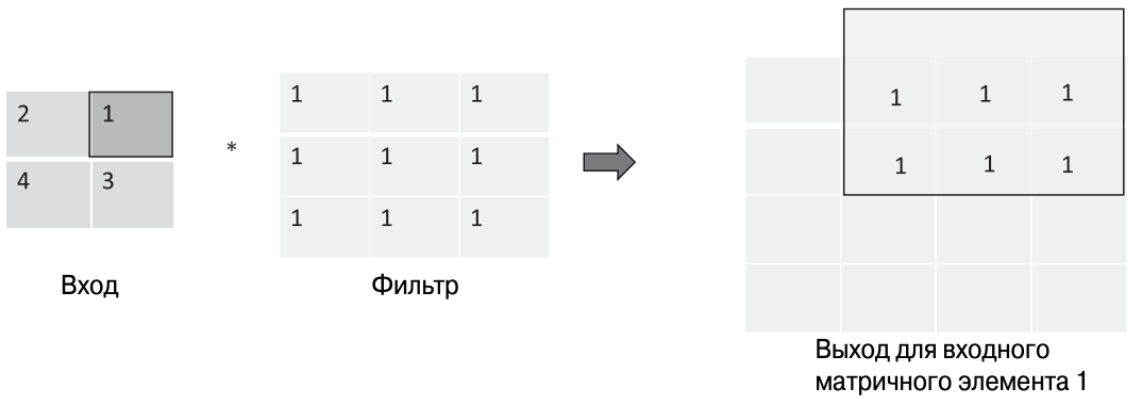
- Воздействие операции транспонированной свертки с использованием размера фильтра или ядра, равным 3x3 на вход размером 2x2: результат - выход размером 4x4.
- В отличие от обычной свертки, когда мы вычисляем поточечное произведение фильтра и части входа, в случае транспонированной свертки значения фильтра при каждом расположении взвешиваются входным значением из той позиции, в которой находится фильтр, и эти взвешенные значения фильтра заполняют соответствующие расположения в выходе.

Deconvolution

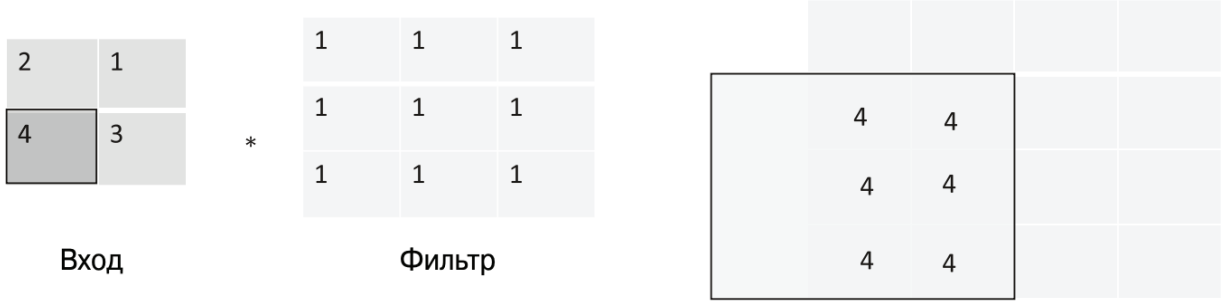
- Выходы для последовательных входных значений вдоль того же пространственного измерения размещаются с зазором, определяемым величиной шага транспонированной свертки.
- Эти действия выполняются по отношению ко всем входным значениям.
- Наконец, выходы, соответствующие каждому из входных значений, складываются для получения конечного выхода

Deconvolution 1

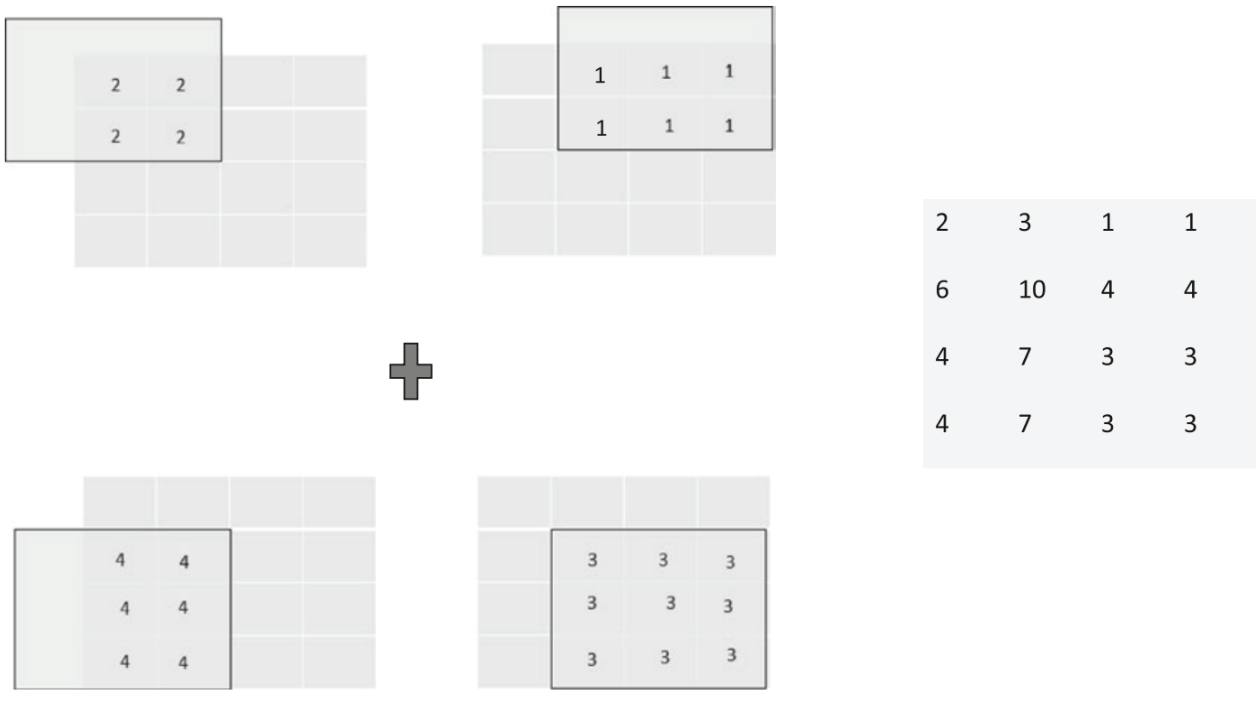




Deconvolution 2

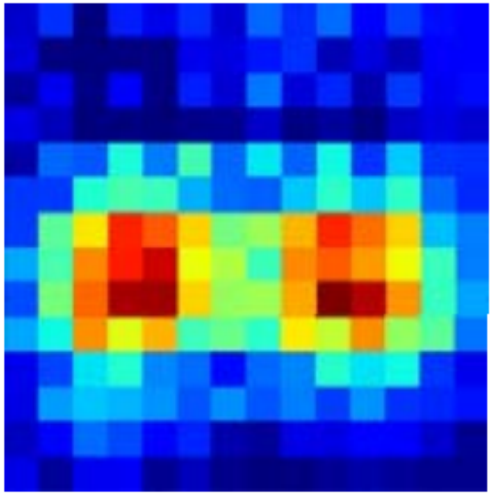


Deconvolution 3

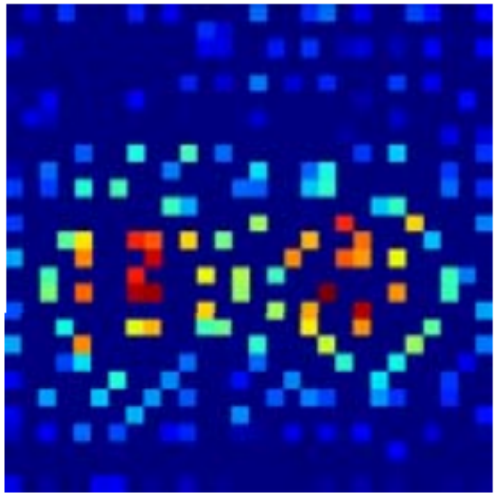


2	3	1	1
6	10	4	4
4	7	3	3
4	7	3	3

Deconvolution and unpooling



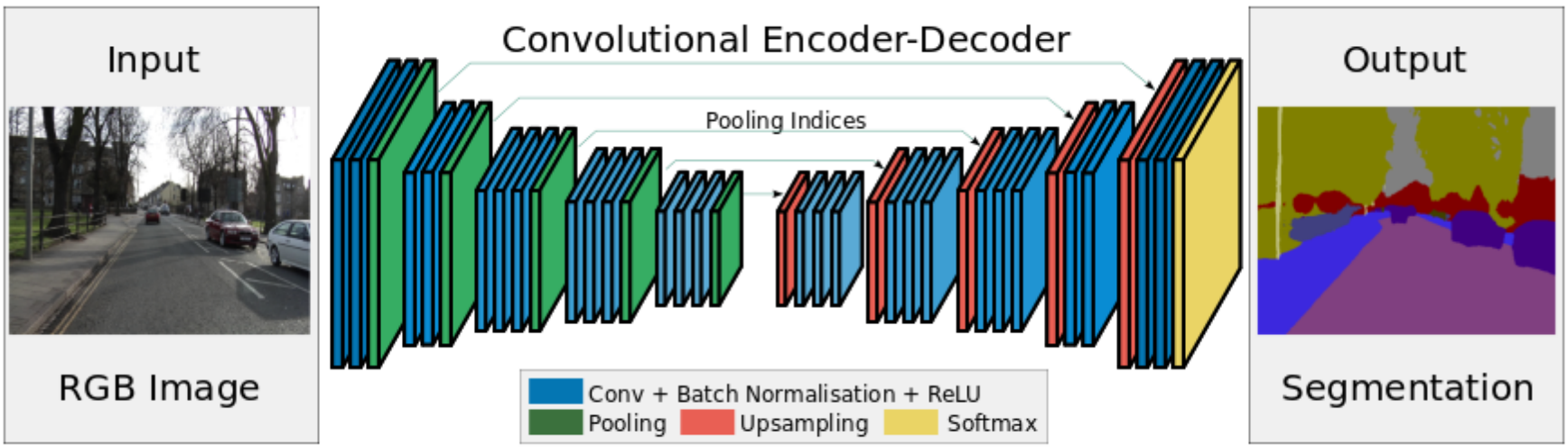
Deconv: 14x14



Unpool: 28x28

SegNet

V. Badrinarayanan, A. Kendall and R. Cipolla "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation."arXiv preprint arXiv:1511.00561, 2015



Особенность SegNet

- Сеть состоит из :
  - последовательности слоев нелинейного преобразования (кодер) и
  - соответствующего множества декодеров, за которыми следует “попикселный” классификатор
- **Ключевой элемент** SegNet - декодер использует индексы, полученные на этапах max-pooling в кодере, для реализации upsampling
- **Индексы** - местоположения признаков с максимальными значениями в каждом окне pooling (запоминаются для каждой карты признаков в кодере)
- Это позволяет избежать обучения проц. upsampling
- Сеть в целом обучается, используя градиентный спуск

Semantic Segmentation, классификация на уровне пикселов

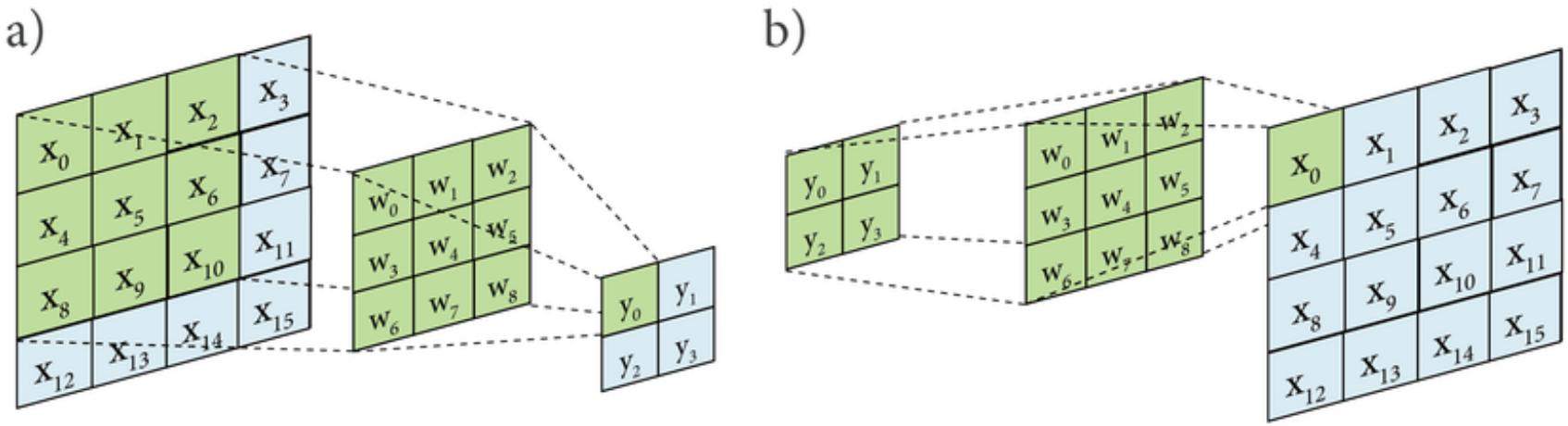
Семантическая сегментация состоит из трех шагов:

1. Выделение всех классов, присутствующих на изображении.
2. Сегментация, которая предоставляет не только классы, но и информацию относительно пространственного расположения этих классов.
3. Создаются метки для каждого пикселя. Эти метки указывают на принадлежность определенного пикселя классу.

Semantic Segmentation, классификация на уровне пикселов

- Сжатие изображения посредством использования пулинга и сверток
- Эта конфигурация кодера для задач классификации, так как она заботилась только о содержании изображения, а не о его местоположении
- Однако для задачи сегментации необходимо иметь маску с полным разрешением для пиксельного прогнозирования

Semantic Segmentation, классификация на уровне пикселов

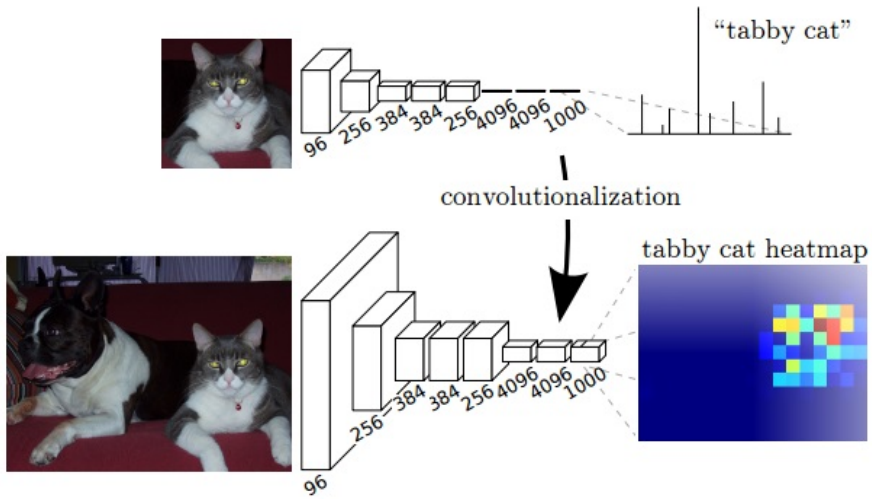


Fully Convolutional Network (FCN) - Полносверточная сеть

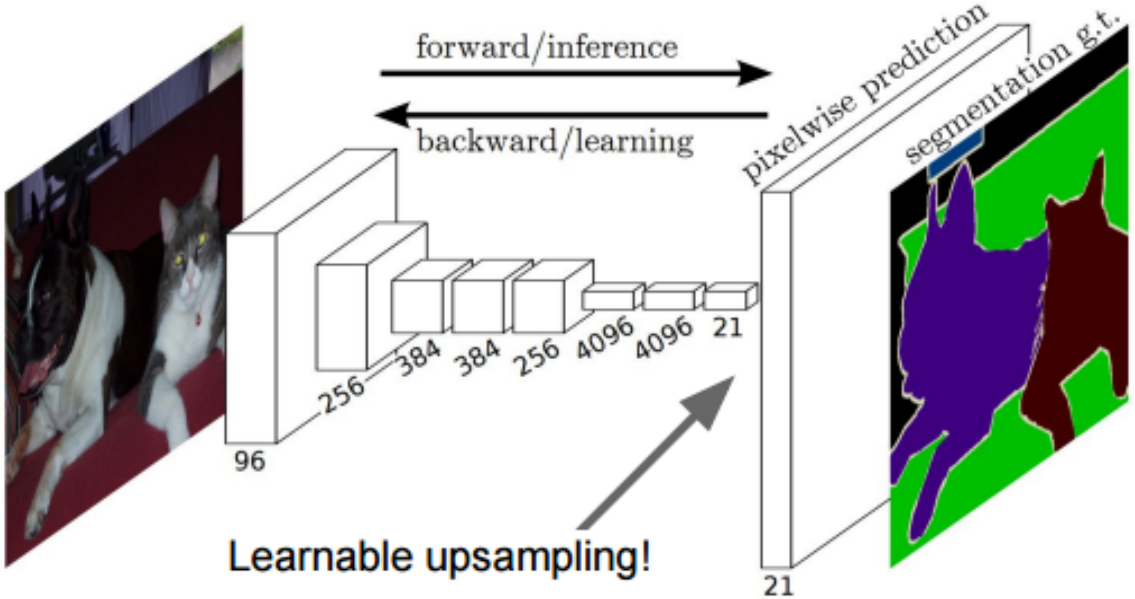
- Полносверточная сеть была предложена для устранения patch-by-patch scanning и для повышения эффективности
- FCN заменяет полносвязанные слои в сверточной сети на 1×1 сверточные ядра.
- FCN в качестве входа использует всю картинку и получает сегментационную карту на выходе одним проходом прямого распространения

Fully Convolutional Network (FCN)

- Преобразование полносвязных слоев в сверточные, охватывающих всю входную область
- Каждая из этих сверток будет выводить coarse heatmap меток

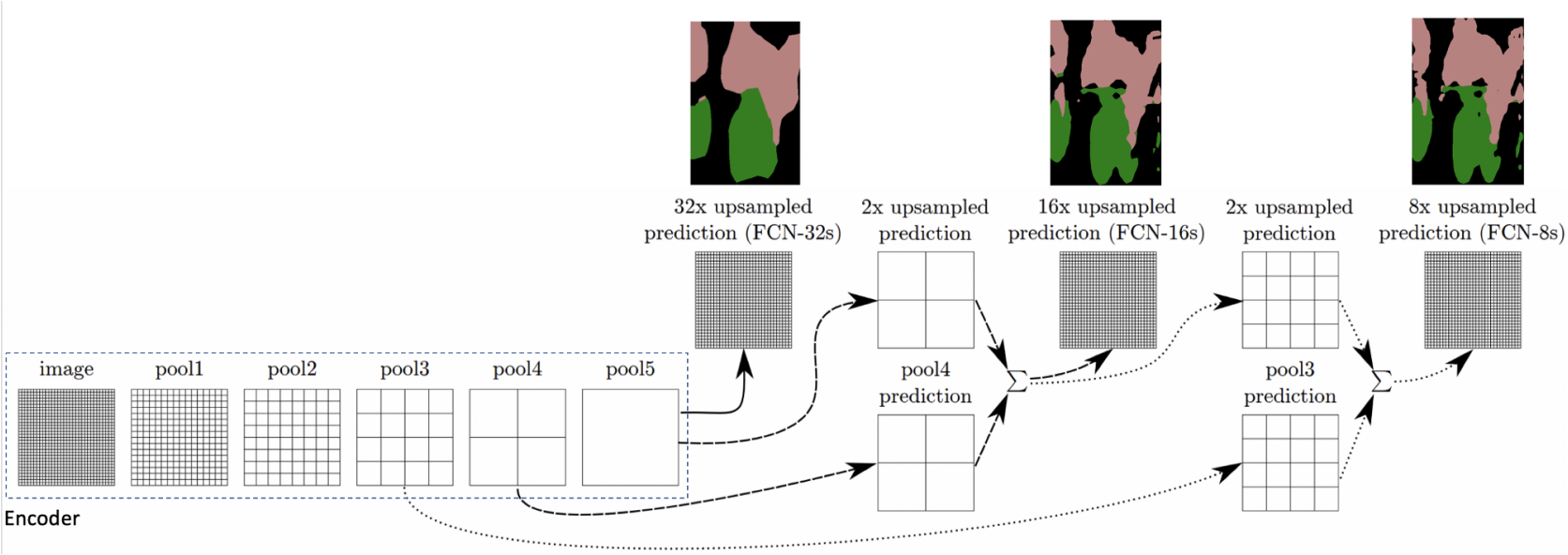


FCN



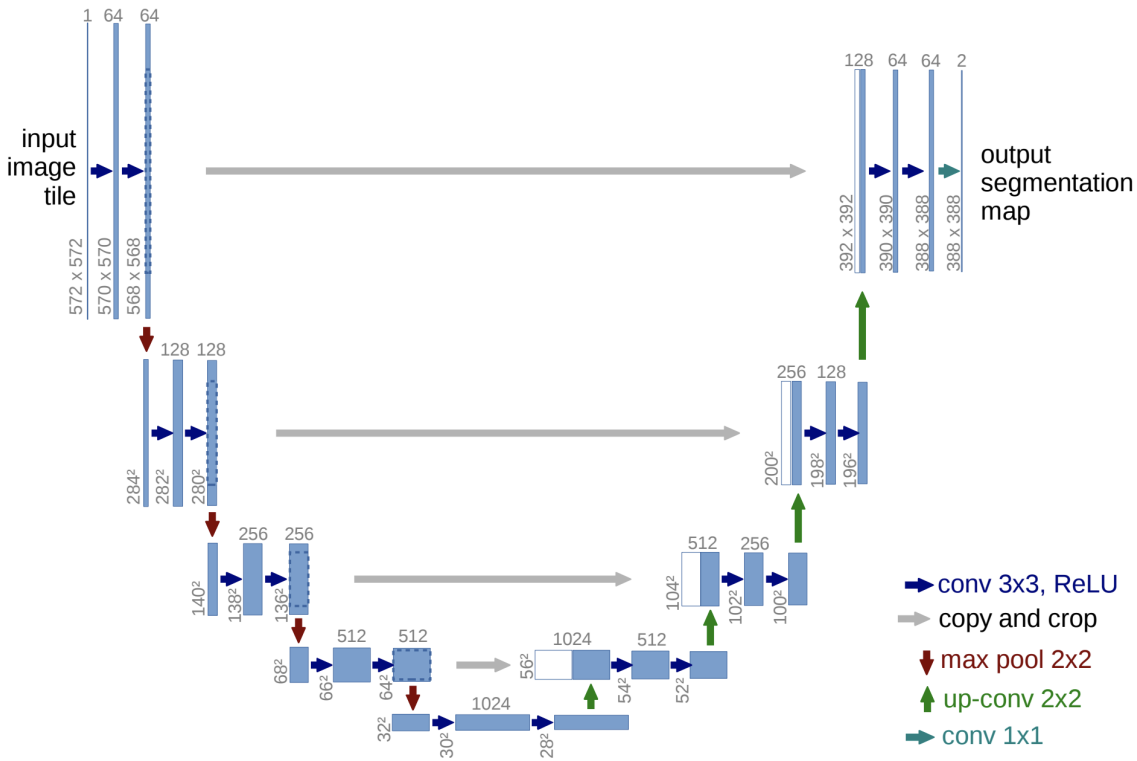
Кодер сжимает изображение в изображение с более низким разрешением, что приводит к грубой сегментации после операции повышения дискретизации (upsampling)

FCN



Положение на слоях более высокого уровня соответствуют положениям на изображении, с которыми они связаны путями, называемыми *рецептивными полями*

U-Net


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

U-Net - первая часть

- Первая часть состоит из обычных сверток, ReLU и пулинга 2x2 с шагом 2, который приводит к понижающей дискретизации (downsampling) посредством сочетания операций свертки и субдискретизации по максимуму.
- Свертки ассоциируются с попиксельными ReLU-активациями.
- Каждая операция свертки выполняется с фильтром размером 3x3 и без дополнения нулями, что приводит к редукции двух пикселей в каждом пространственном измерении для выходных карт признаков.

U-Net - вторая часть

- Вторая часть состоит из последовательности слоев повышающей дискретизации (upsampling), конкатенации соответствующей карты признаков, которая необходима из-за потери границ пикселей после каждой свертки.
- Во второй части сети разрешение субдискретизированных изображений увеличивается до достижения последнего слоя, где выходные карты признаков соответствуют конкретным классам сегментируемых объектов.
- Как и FCN, признаки высокого разрешения из пути сжатия объединяются с upsampled выходом, который затем подается на ряд сверточных слоев.

U-Net - еще

- Важно: выходные карты признаков в U-Net имеют меньшие размеры по сравнению с размерами входных изображений. Например, входное изображение с пространственными размерами 572x572 произведет выходные карты признаков с пространственными размерами 388x388.
- Как выполняется попиксельное сравнение классов для вычисления функции потерь? Идея проста: сегментированные выходные карты признаков сравниваются с целевым сегментированным изображением с размером фрагмента 388x388, извлеченным из центра входного изображения.

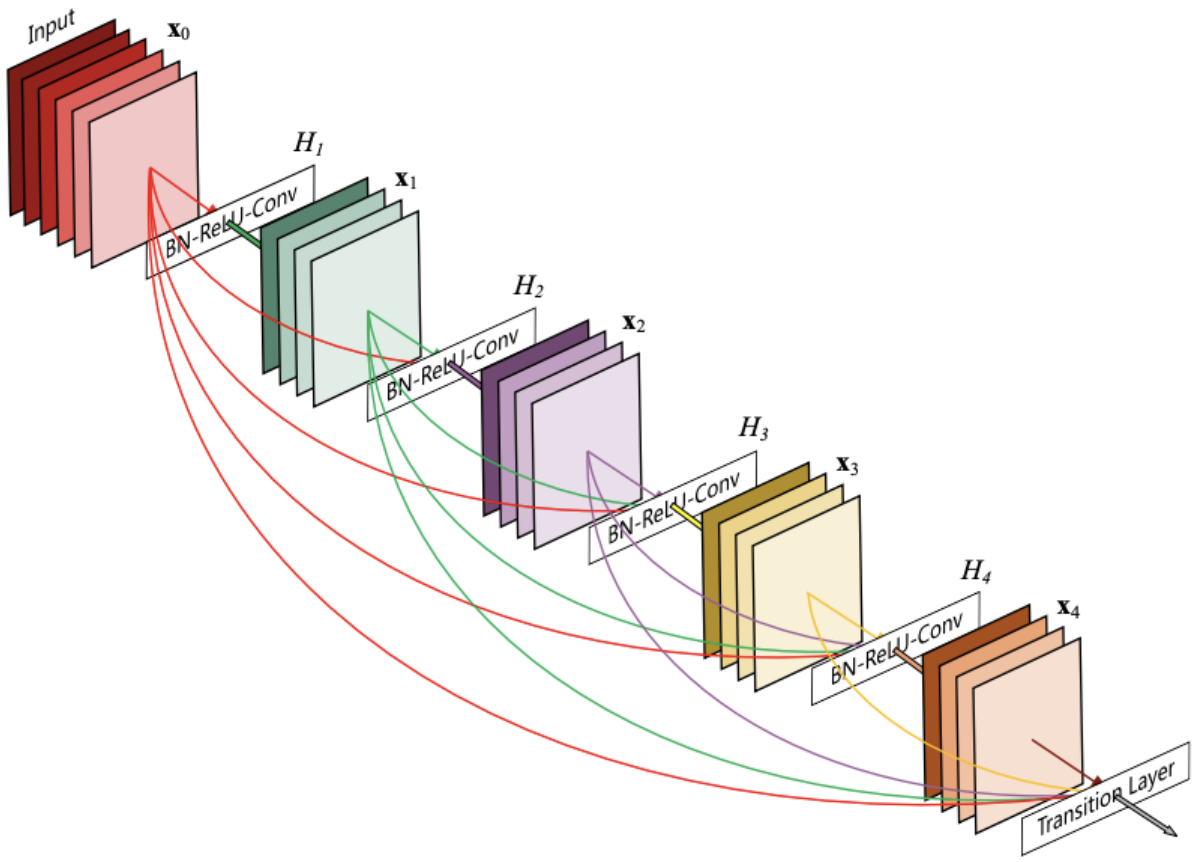
U-Net - еще

- Основная мысль заключается в том, что при наличии изображений с более высоким разрешением, скажем, 1024x1024, можно создать из него множество случайных изображений с пространственными размерами 572x572 для целей обучения.
- Целевые изображения также создаются из этих вспомогательных изображений размером 572x572 путем извлечения центрального фрагмента размером 388x388 и снабжения каждого его пикселя меткой соответствующего класса.
- Это позволяет тренировать сеть на значительных объемах данных, даже если для обучения доступно не так уж много изображений.

U-Net - плюсы

- Используя всего лишь небольшое количество аннотированных сегментированных изображений, можно генерировать значительные объемы данных.
- Хорошо справляется с сегментацией изображений даже при наличии соприкасающихся объектов одного класса, которые должны быть разделены. Это делается за счет введения больших весов неправильной классификации пикселей вблизи границ соприкасающихся сегментов

FC-DenseNet

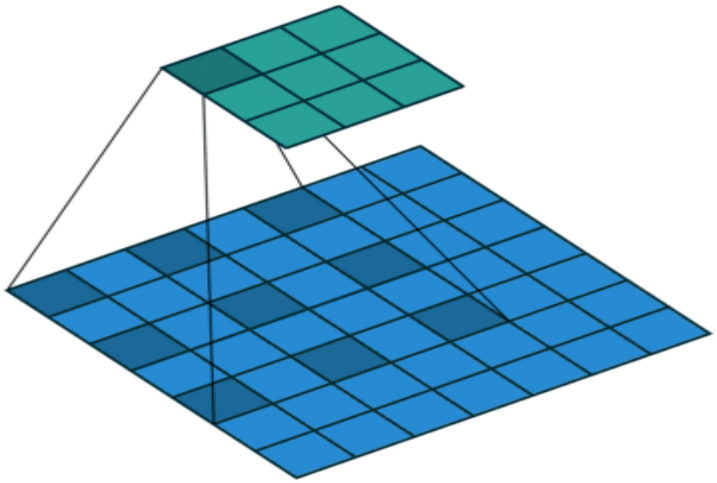

Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer

FC-DenseNet

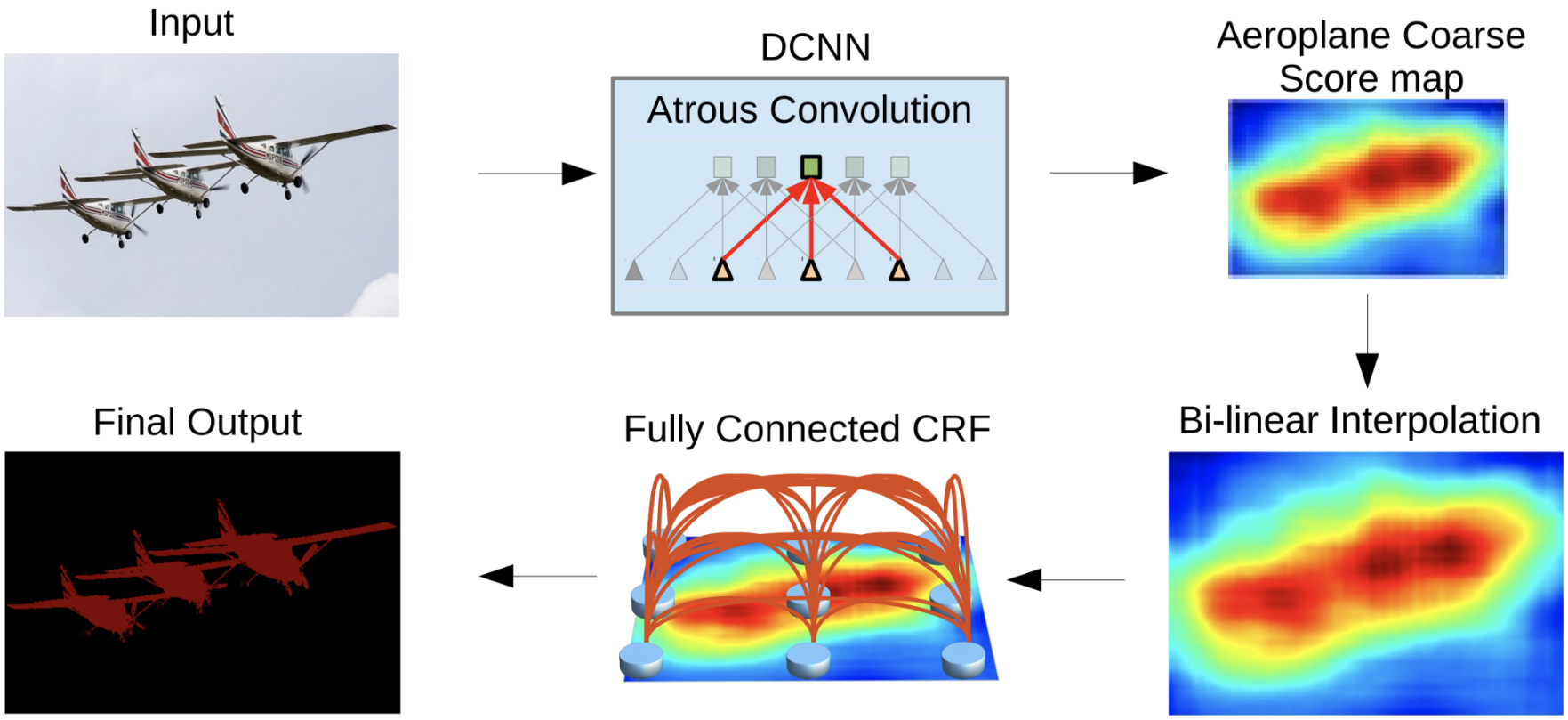
- FC DenseNet или 100 Layers Tiramisu - это метод сегментации, основанный на архитектуре DenseNet
- DenseNet основан на том, что “перемычки” с ранних уровней устанавливаются на поздние уровни и все слои связаны друг с другом
- Каждый слой передает свои карты признаков всем последующим слоям
- Если ResNet использует поэлементное суммирование для объединения признаков, то DenseNets - конкатенацию
- Каждый уровень получает совокупный набор “знаний” от всех предыдущих уровней

DeepLab

- DeepLab v1 основан на двух идеях: Atrous Convolution и полносвязанное условное случайное поле (FC CRF)
- Atrous convolution с французского “a trous” - дыра (hole), также назывется “расширяющаяся (dilated) свертка”
- “Расширяющаяся (dilated) свертка” - это стандартная свертка, через которую пропускается некоторое число пикселей в двух измерениях



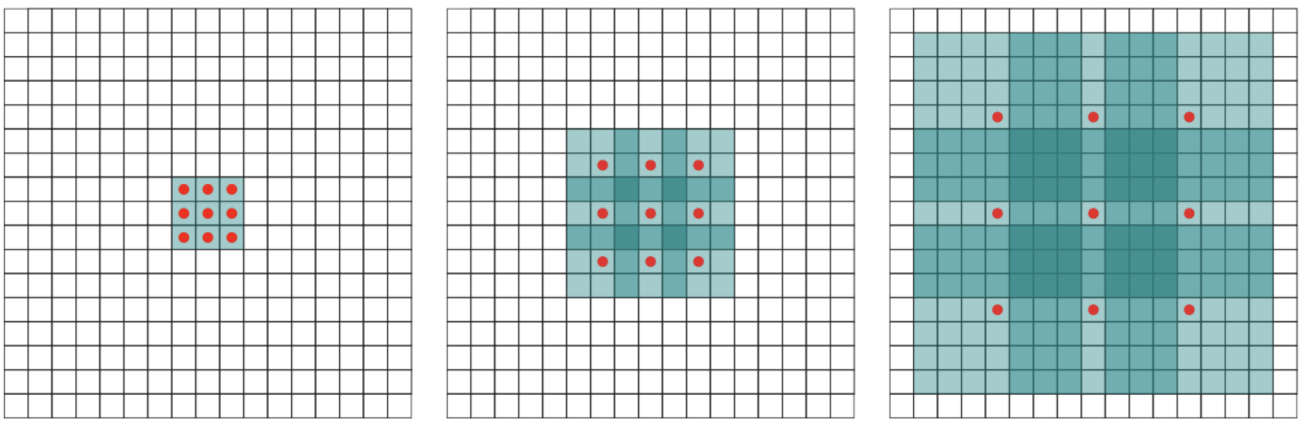
DeepLab



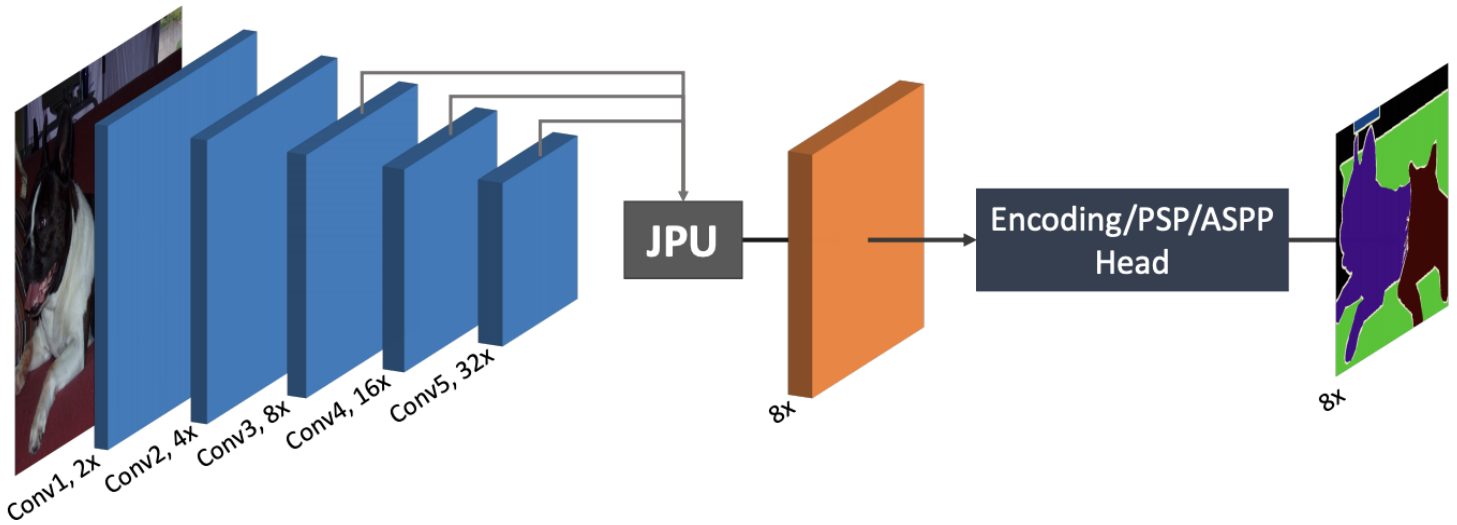
FC CRF is applied to refine the segmentation result

DeepLab

- Вместо типового пулинга DeepLab использует расширенные слои для решения проблемы балансировки
- Контролируя поле зрения в dilated свертках, можно найти лучший компромисс между точной локализацией (малое поле зрения) и ассимиляцией контекста (большое поле зрения)

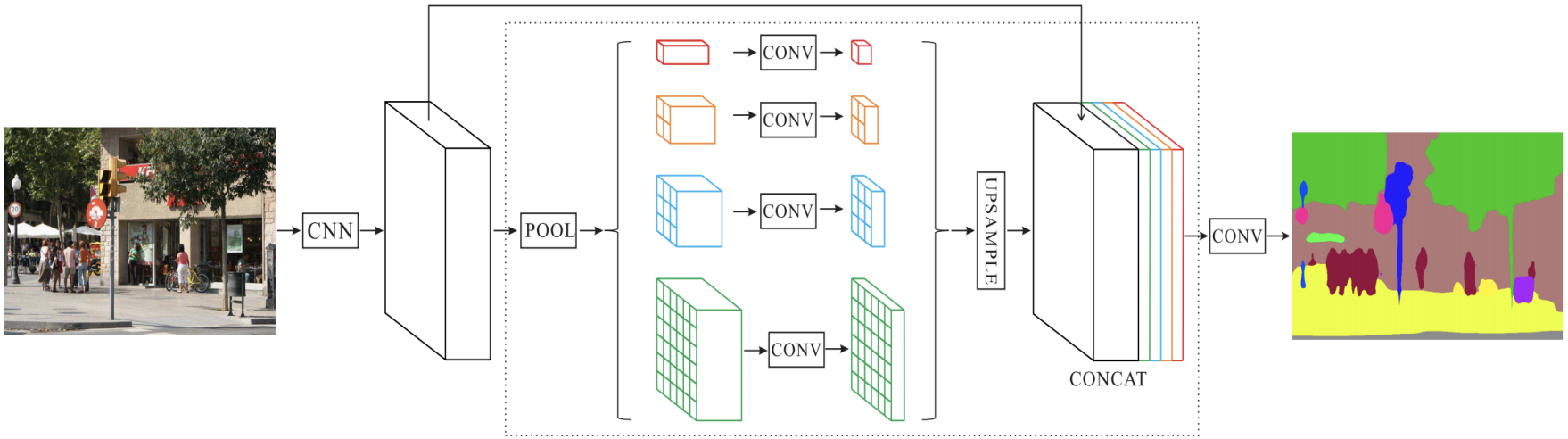


Fast FCN

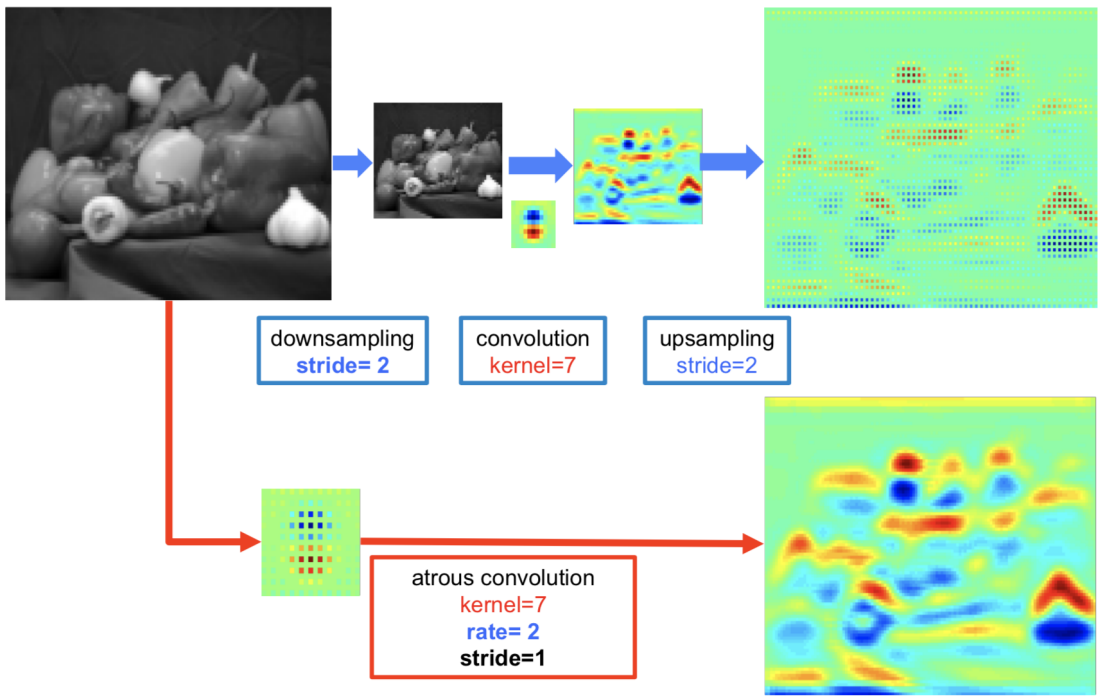


ASPP - atrous spatial pyramid pooling; PSP - pyramid scene parsing network

PSP - pyramid scene parsing network



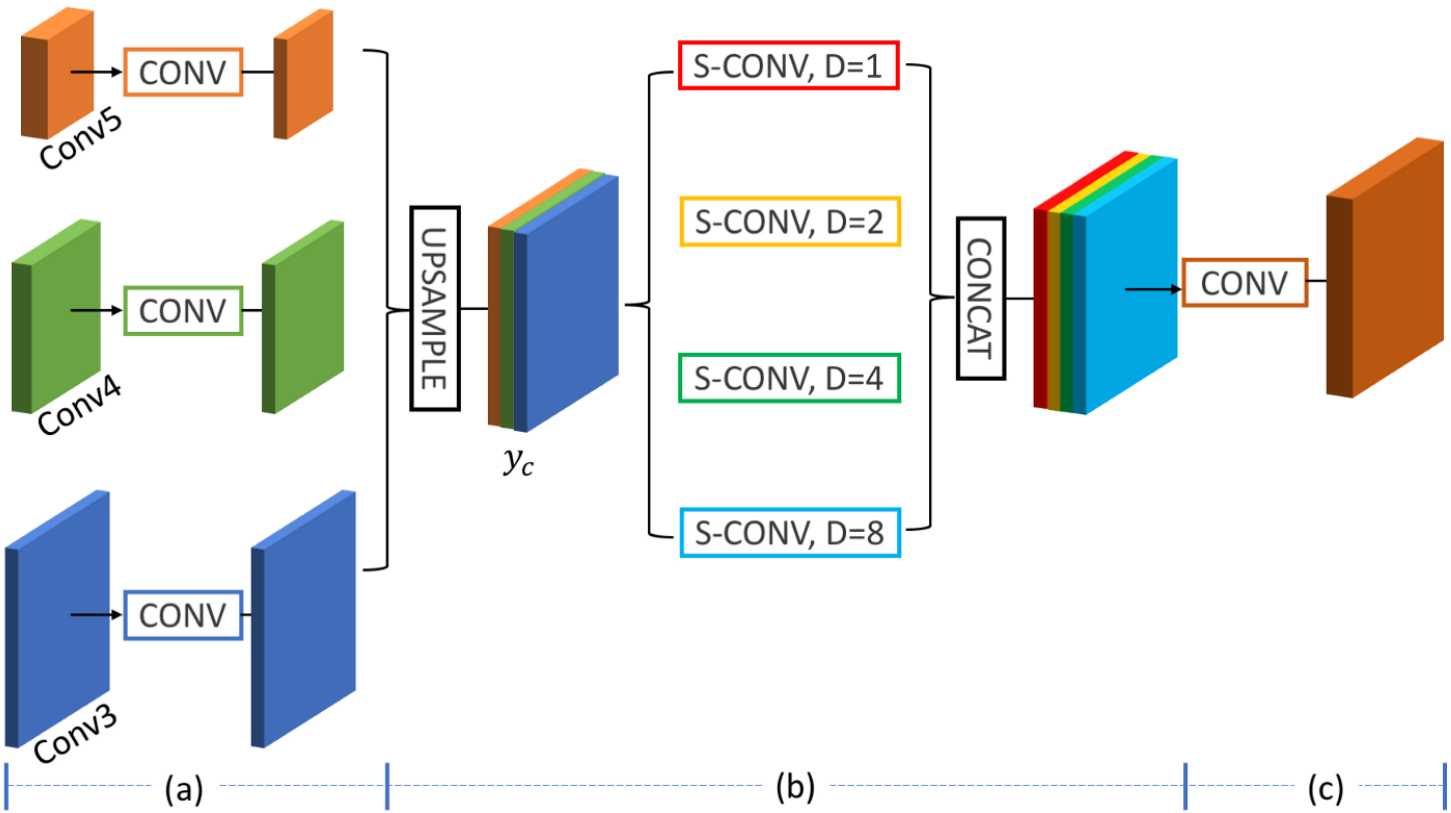
Пример “atrous convolution”



Fast FCN

- Принцип расширенной свертки был заменен совместным пирамидальным upsampling (JPU)
- Различия между Fast FCN и DilatedFCN заключаются в последних двух стадиях свертки
- Как правило, карта входных объектов сначала обрабатывается обычной сверткой, а затем серией расширенных сверток.
- Fast FCN концептуально обрабатывает входную карту признаков сверткой, а затем использует несколько сверток для генерации выходных данных
- Это снижает вычислительную сложность по сравнению с DilatedFCN
- JPU создан, чтобы упростить процесс оптимизации.

Fast FCN - архитектура JPU



Fast FCN - архитектура JPU

- Каждая карта признаков проходит через обычный сверточный блок
- Затем карты признаков подвергаются дискретизации и конкатенации, которые затем проходят через четыре свертки с различными степенями расширения
- Результаты свертки снова объединяются и проходят через слой окончательной свертки
- Где ASPP использует только информацию из последней карты признаков, JPU извлекает информацию о всем контексте из карт многоуровневых объектов, что приводит к повышению производительности

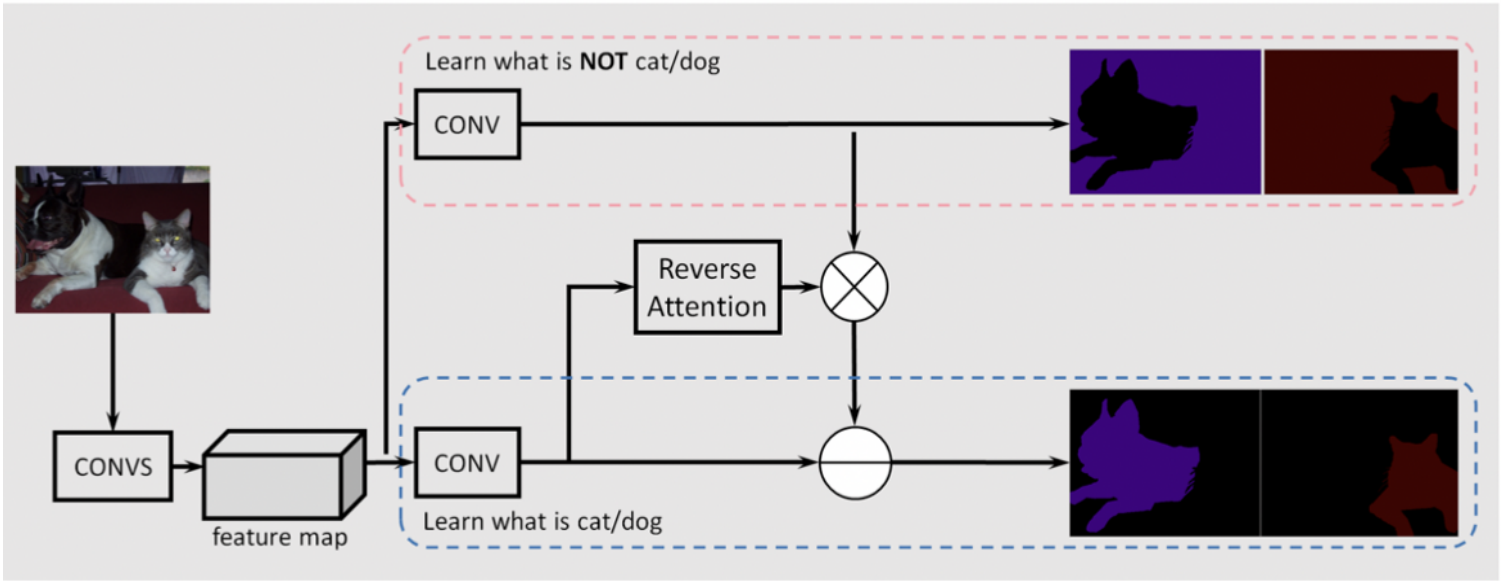
Семантическая сегментация с обратным вниманием

- Qin Huang, et al. Semantic Segmentation with Reverse Attention. arXiv:1707.06426, 2017.
- Материалы также: https://medium.com/@WellWritten. Oct 28, 2022
- Проблемы: из-за визуального сходства классы обладают общими высокоуровневыми признаками, что может привести к смешиванию классов. Например, фон может сливаться с объектом, поскольку они имеют схожую силу нейронную активации на участках, где фон и объект “смешаны”.

Семантическая сегментация с обратным вниманием

- Сеть с обратным вниманием обучается не только распознавать фоновый класс, но и дифференцировать различные объекты, находящиеся на изображении.
- RAN (reverse attention network, сеть обратного внимания).
- В RAN есть две ветви (одна красной пунктирной линией, другая - синей), предназначенные для изучения признаков фона и признаков объекта.

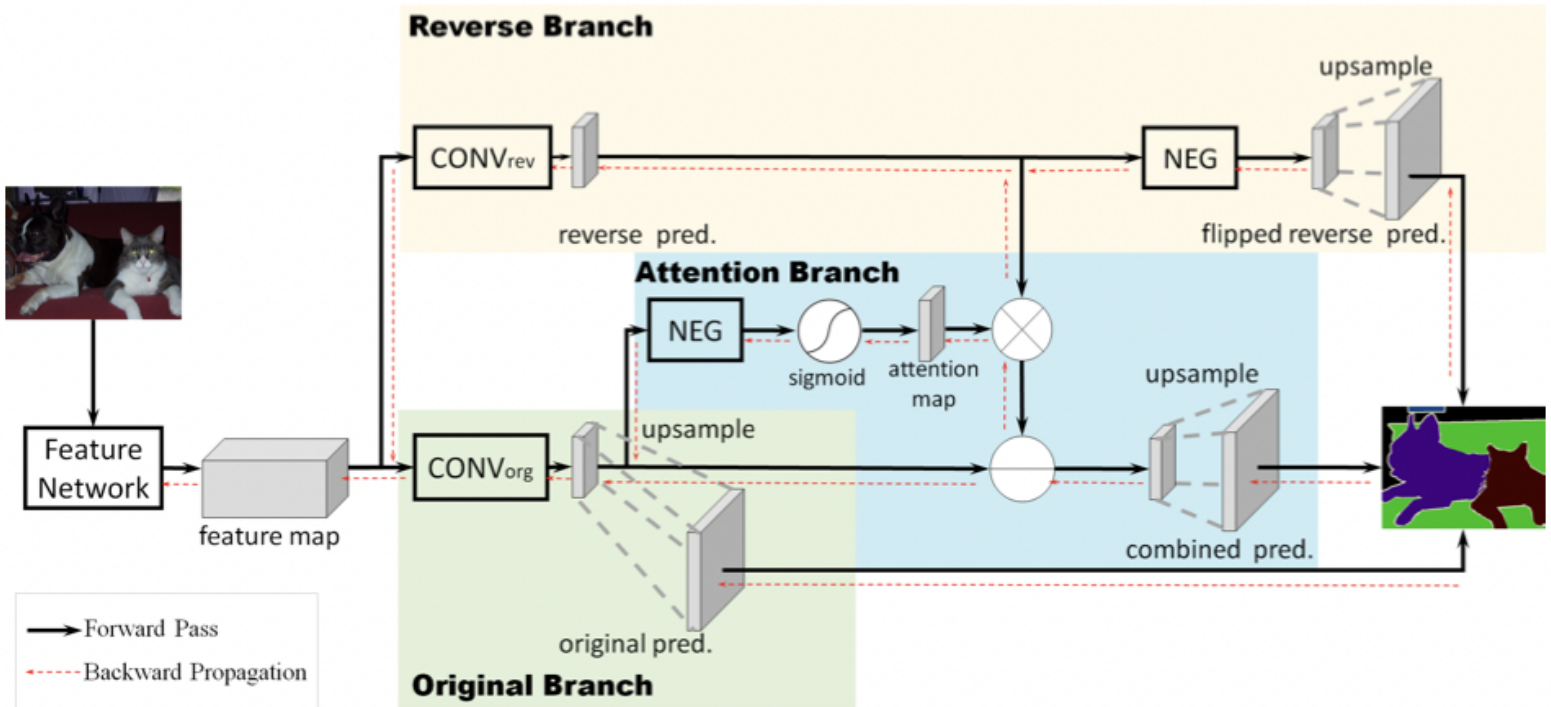
Семантическая сегментация с обратным вниманием



Семантическая сегментация с обратным вниманием

- За дальнейшее выделение информации, полученной моделью от класса object, отвечает структура обратного внимания. Она генерирует маски для каждого класса, чтобы усилить активации класса object в смешанной области.
- В завершение прогнозы объединяются для получения окончательного прогноза.

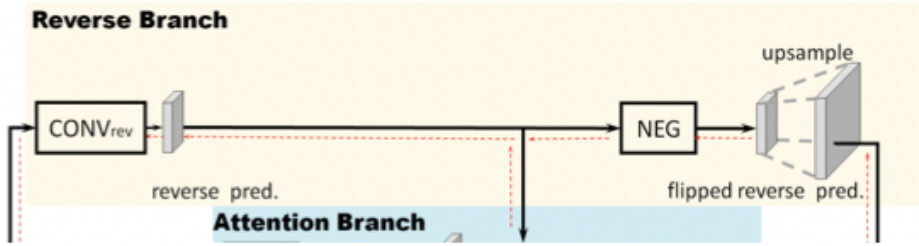
Сеть обратного внимания (RAN)



Сеть обратного внимания (RAN)

- После получения входного изображения процесс можно разбить на несколько этапов.
  - Создается карта признаков с использованием выбранной архитектуры модели (обычно ResNet-101 или VGG16, но возможны и другие вариации) для изучения признаков объекта.
  - Затем карта разделяется на две ветви.

Обратная ветвь (RB)



- Сначала обучается слой CONV_rev, выделенный желтым цветом, для явного изучения “класса reverse object” — это обратные эталонные данные для класса object.
- Чтобы получить класс reverse object, фон и другие классы устанавливаются на 1, а класс object - на 0.

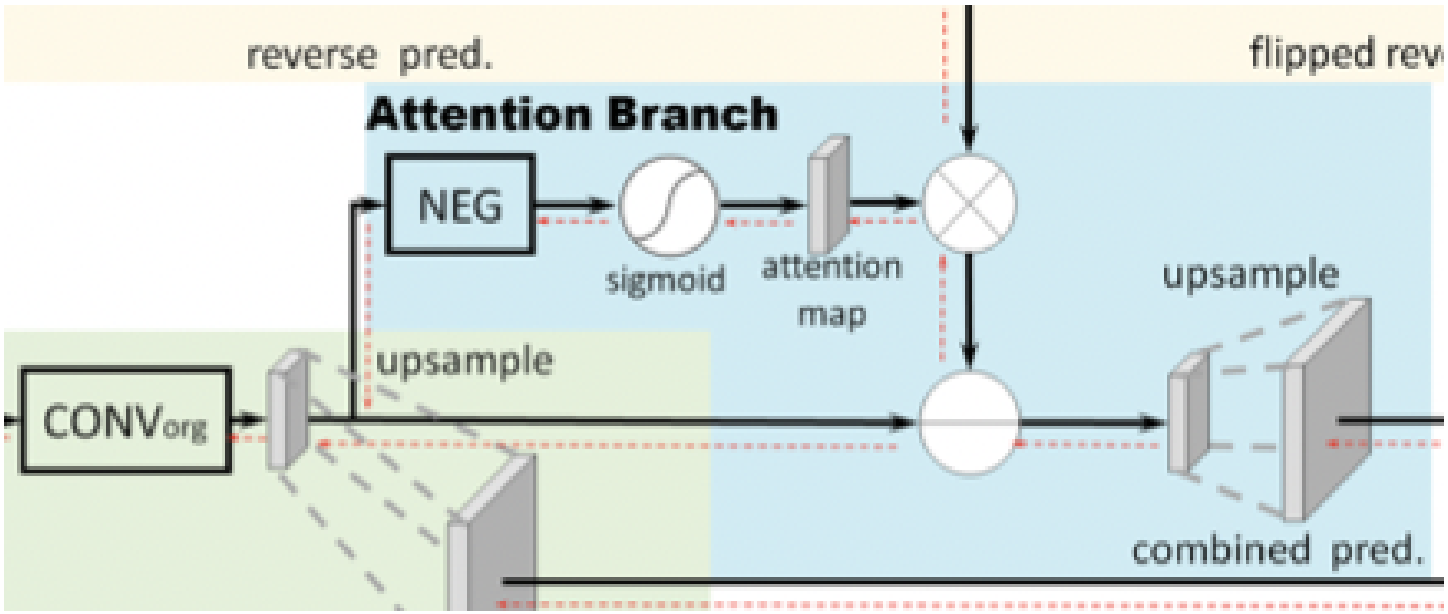
Обратная ветвь (RB)

Однако, когда необходимо решить проблему многоклассовой сегментации, обычно используется альтернативный способ, при котором перед подачей в классификатор, основанный на функции softmax, меняется знак всех активаций по классам (блок NEG). Этот подход позволяет обучить слой CONV_rev, используя ту же самую метку по классам “ground-truth”.

Ветвь обратного внимания (RAB)

- Вместо того чтобы напрямую применять поэлементное вычитание к исходному прогнозу путем активаций обратной ветви (что снижает производительность), предлагается использовать ветвь обратного внимания для выделения областей, пропущенных исходным прогнозом (включая смешанные и фоновые области).
- Вывод обратного внимания генерирует маску, ориентированную на класс, чтобы усилить карту обратной активации.

Ветвь обратного внимания (RAB)



Ветвь обратного внимания (RAB)

- Исходная карта признаков входного изображения подается в слой CONV_org.
- Затем значения пикселей полученной карты признаков переворачиваются (flip the sign of the pixel value) блоком NEG.
- Потом применяется сигмоидная функция для преобразования значений пикселей в диапазоне [0,1].
- Карта признаков подается на карту внимания, где применяется маска внимания.
- Значения пикселейв обратной карте внимания равно

$$I_{ra}(i,j) = \mathrm{Sigmoid}(-F_{CONV_{org}}(i,j))$$

- где i, j - расположение пикселей, $F_{CONV_{org}}$ - карта признаков.

Сеть обратного внимания (RAN)

- Т.о. область с малой или отрицательной реакцией будет выделена с помощью NEG и сигмоидной операции, но области с положительной активацией (или оценками уверенности) будут подавляться в обратной ветви внимания.

Объединение результатов

- Затем карта из ветви обратного внимания (Reverse Attention Branch) поэлементно перемножается с картой из обратной ветви (Reverse Branch).
- Полученная карта вычитается из исходного прогноза для получения окончательного прогноза.

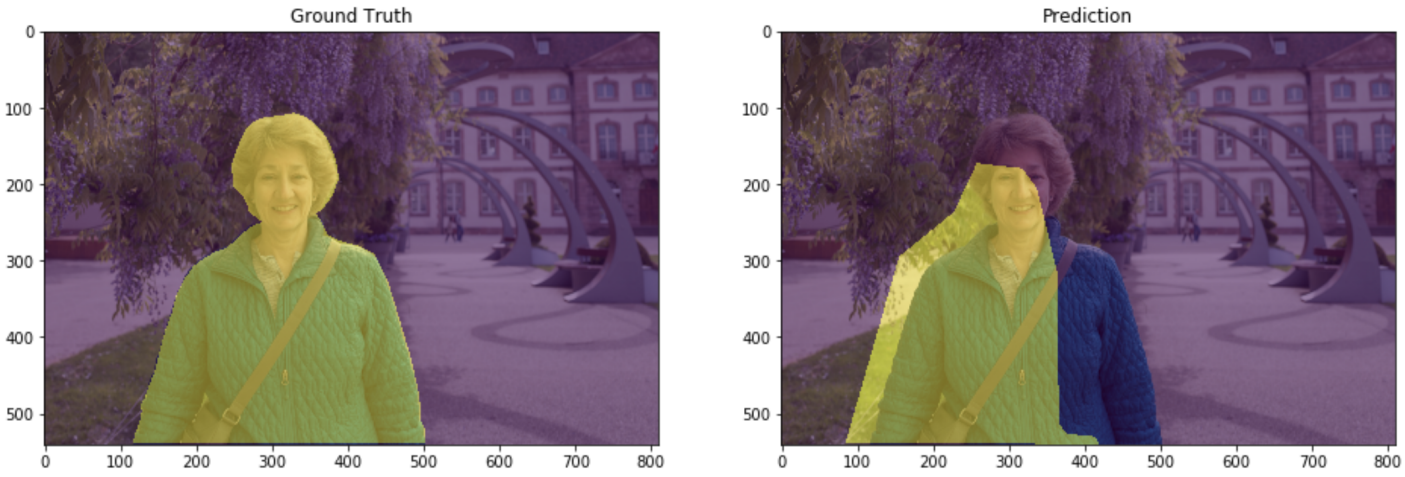
Обучение (1)

- Для обучения RAN осуществляется обратное распространение сфункцией потерь - кросс-энтропией на трех ветвях одновременно.
- Используются softmax-классификаторы на трех выходах предсказания.
- Со всеми тремя потерями необходимо считаться для обеспечения сбалансированного сквозного процесса обучения.

Обучение (2)

Потеря оригинального предсказани и потеря обратногопредсказани позволяют CONVorg и CONVrev параллельно обучать целевые классы и их обратные классы. Кроме того, потеря комбинированного прогноза позволяет сети обучать обратное внимание.

Оценка качества сегментации



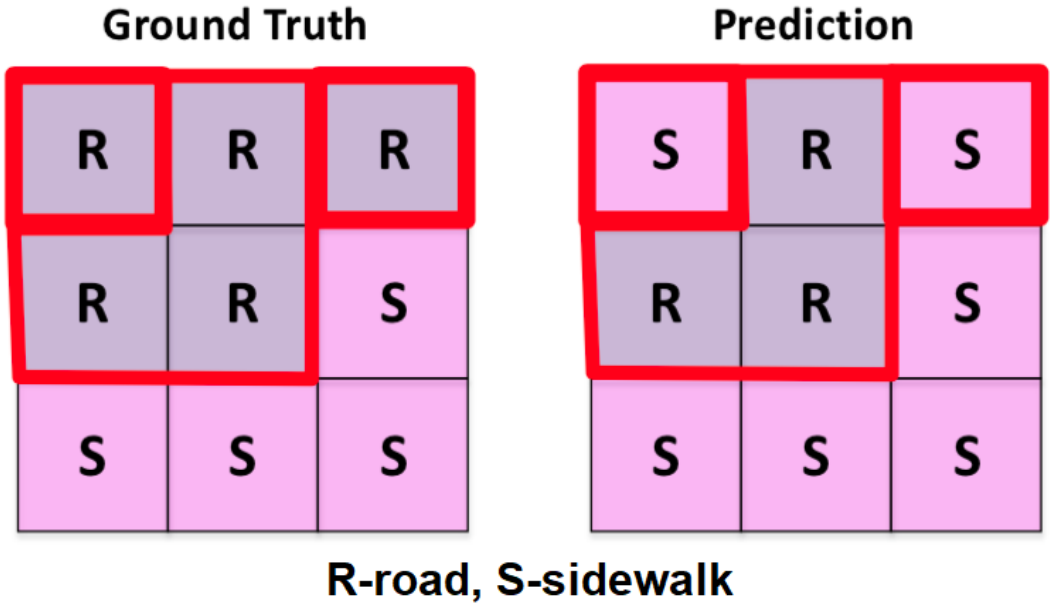
https://www.jeremyjordan.me/evaluating-image-segmentation-models/

Оценка качества сегментации - Pixel Accuracy

- Pixel Accuracy - процент пикселей в изображении, которые были правильно классифицированы
- Pixel Accuracy вычисляется для каждого класса отдельно, а также для всех классов в целом
  - TP: пиксель правильно классифицирован как X
  - FP: пиксель не правильно классифицирован как X
  - TN: пиксель правильно классифицирован как не X
  - FN: пиксель не правильно классифицирован как X

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

Pixel Accuracy - пример



$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3 + 4}{3 + 4 + 0 + 2} = \frac{7}{9}$$

Pixel Accuracy - недостаток

Птицы - positive класс и Небо - negative класс. TN (Небо) - высокий -> Pixel Accuracy - высокая.

$$accuracy = \lim_{TN \to \infty} \frac{TP + TN}{TP + TN + FP + FN} \to 1$$

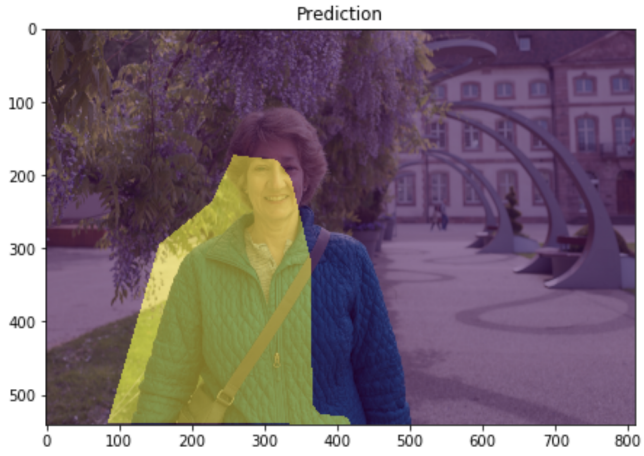
Оценка качества сегментации - IoU

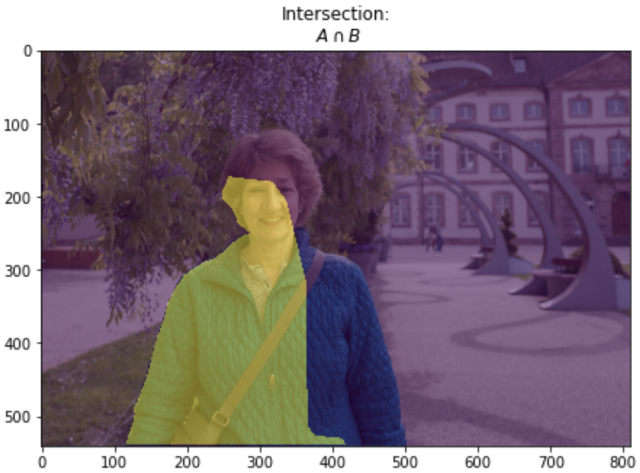
- Intersection over Union (IoU) = Jaccard index

$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

- IoU измеряет количество пикселей, общих для целевой маски и маски после сегментации, деленное на общее количество пикселей, присутствующих в обеих масках
- IoU вячисляется для каждого класса в отдельности, а затем усредняется по всем классам, чтобы получить глобальный средний показатель IoU

Оценка качества сегментации - пример





https://www.jeremyjordan.me/evaluating-image-segmentation-models/

Оценка качества сегментации - Dice

- Dice - показатель перекрытия
- Используется больше при определении функции потерь

$$Dice = 2\frac{\#\{target \cap prediction\}}{\#target + \#prediction}$$

- $\#\{target \cap prediction\}$ заменяют поэлементным умножением сегментационной и целевой масок, а затем суммируют полученную матрицу

Dice index

$$|A \cap B| = \underbrace{\begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix}}_{\text{prediction}} * \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{\text{target}}$$

$$\xrightarrow{\text{element-wise multiply}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

Вопросы

?