онлайн-курс

СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Сегментация изображений

Постановка задачи

В чём проблема?

Для каждого пиксела картинки нужно указать, какому объекту он принадлежит. Также нужно указать класс каждого объекта на картинке.

Сегментация = классификация

В сегментации **мы фактически классифицируем пиксели** в сегментированном изображении. Например, в указанной картинке каждый пиксел отнесен к одному из следующих классов: автомобили, трамваи, знаки и светофоры, пешеходы, тротуар, дорога, прочее...

Что сложнее: сегментация или классификация?

С одной стороны, сегментация сложнее, поскольку мы классифицируем каждый пиксел, а не картинку в целом.

С другой стороны, мы не обязаны каждый пиксел классифицировать в точности. Главное, чтобы контуры объектов **в основном** совпадали с реальными.

Простой способ

Нужно **свести** задачу сегментации к задаче **распознавания изображений**. Для этого берём большую CHC (например,VGG), которая может распознавать объекты из тысячи классов.

Далее **для каждого пиксела** P изображения выполняем следующие действия:

1) Строим S — квадратик небольшого размера с центром в пикселе P

Простой способ

2) Квадратик S подаем на вход СНС, и смотрим, какой класс она предскажет. Например, для квадратика на рисунке ниже будет предсказано, что на нём изображён автомобиль.

3) Тогда мы считаем, что пиксель P принадлежит объекту из класса «автомобиль».

Простой способ

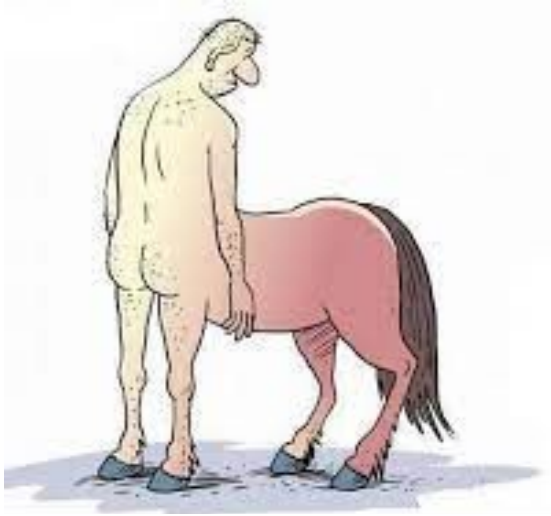
Т.о. для каждого пиксела будет предсказан класс объекта, к которому он принадлежит.

Например, следующий пиксел P будет классифицирован как принадлежащий «луже» (а может быть как «морю»).

Ограничения метода

СНС, используемая для сегментации, должна уметь распознавать **абсолютно все классы**, представители которых могут встретиться на картинках для сегментации.

Кроме того возникают проблемы, когда объект состоит из **разнородных частей** (например, эта картинка будет сегментирована на «мужчину» и «лошадь», а не на «кентавра»).



Второе ограничение — это...

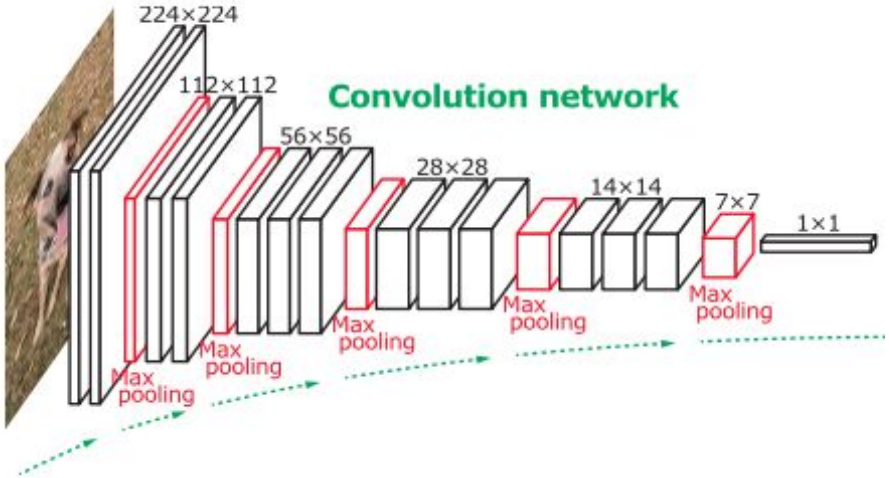
...высокая вычислительная сложность. Ибо приходится запускать большую СНС N раз, где N - количество пикселей изображений.

Вывод: сегментирующую НС для своих задач лучше тренировать самим.

Специальные архитектуры НС для сегментации

Что CHC умеют делать с изображениями?

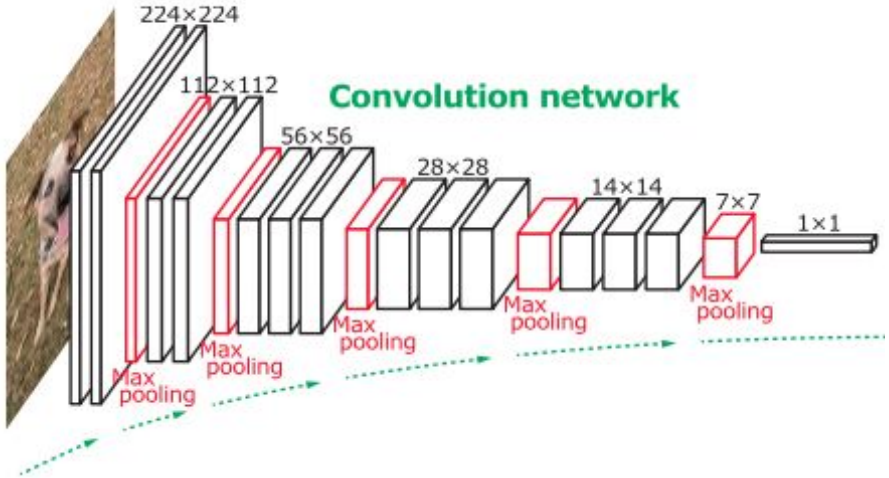
- могут применять к ним фильтры (свёрточные слои);
- уменьшать размерность (пулинг);
- то есть обычная CHC уменьшает размер изображения, но увеличивает количество каналов.



Размер выхода сегментирующей НС

Проблема в том, что размер выхода **сегментирующей** НС совпадает с размером входа (поскольку сегментация — это фактически перекрашивание картинки в ограниченное число цветов).

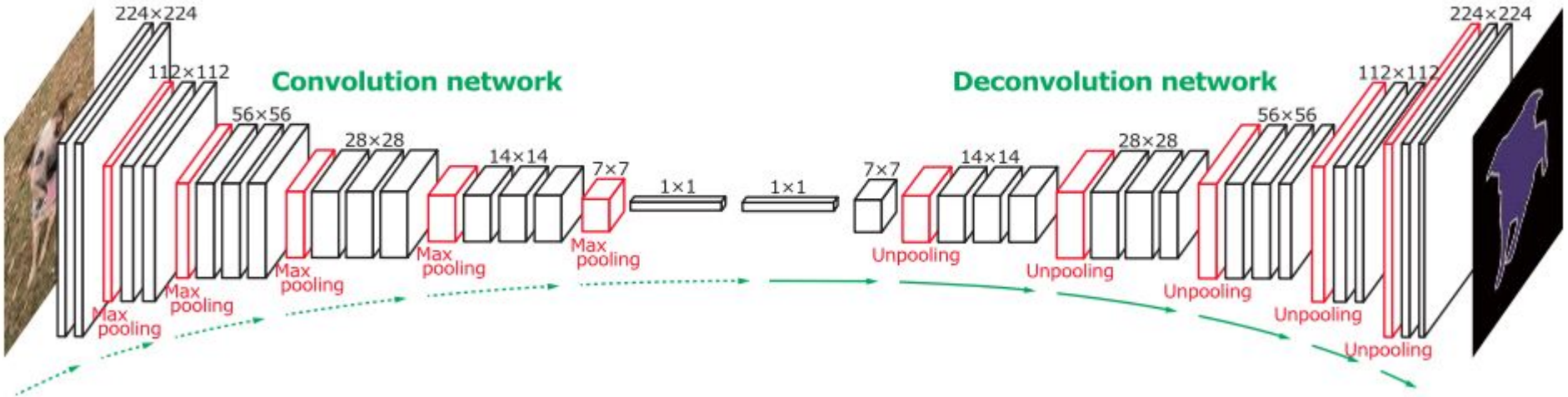
То есть архитектура должна быть такой:



.....

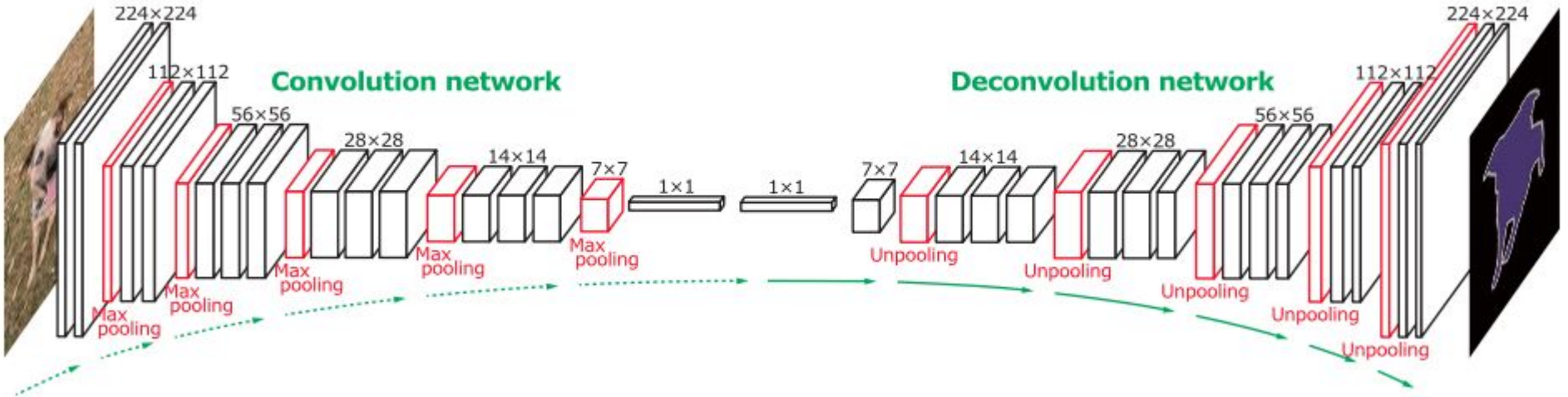
Размер выхода сегментирующей НС

Следовательно, во второй части НС нужно применять преобразования, которые **наоборот** (!) увеличивают размерность изображения и уменьшают количество каналов!
То есть возникает **архитектура песочных часов (hourglass architecture)**.



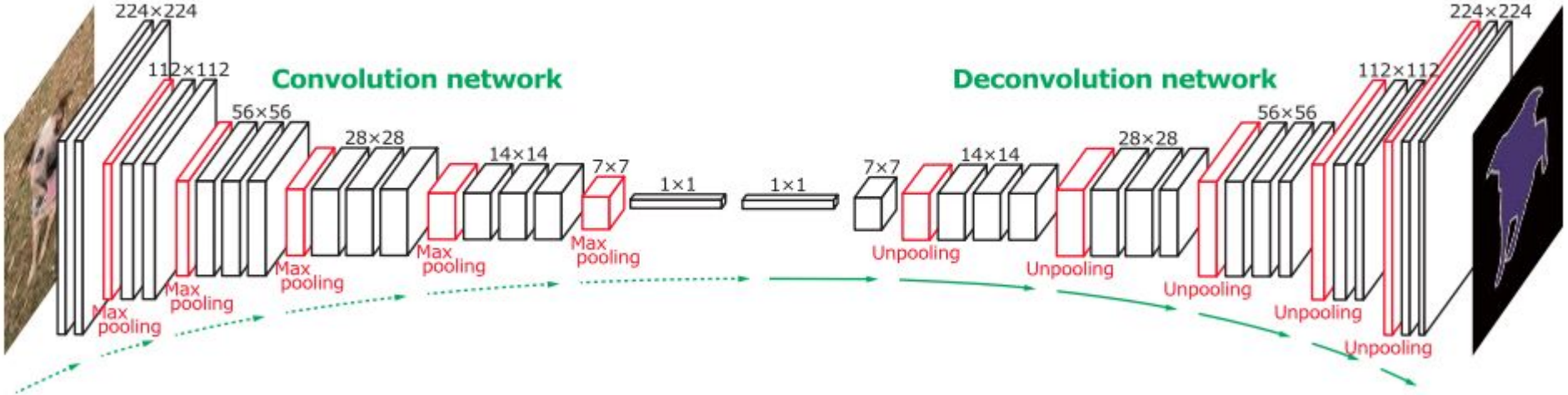
Обратные преобразования к свёртке и пулингу

Во второй части сегментирующей НС применяются операции **decovolution** и **unpooling** — обратные к операциям свёртки (конволюции) и пулингу.



Симметричность сети

Сегментирующая НС должна быть **полностью симметричной**.
Каждой свёртке в первой части должна соответствовать «развёртка» во второй части с аналогичными параметрами.
Точно так же и с пулингом.

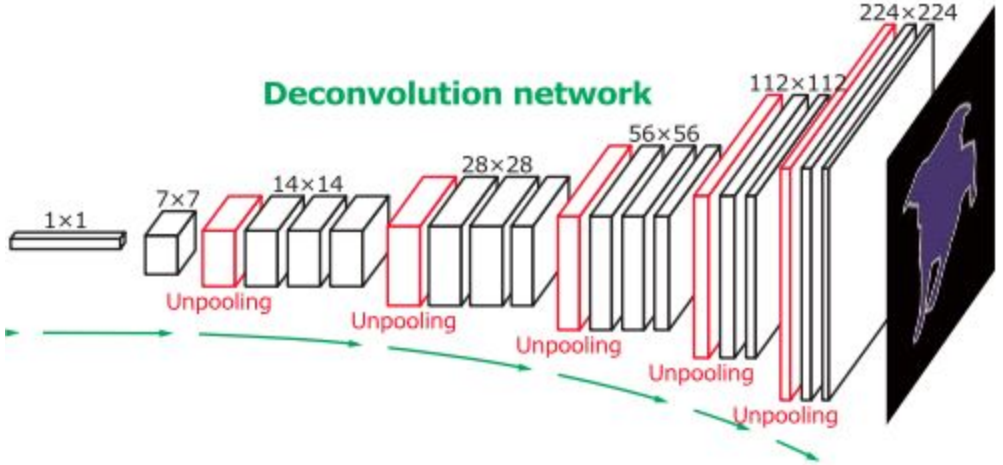


Деконволюция

Как делать развёртку (деконволюцию)?

Из симметричности сегментирующей НС следует, что если в первой части сети был свёрточный слой, то во второй части должен стоять «развёрточный» слой.

А что это такое?



Деконволюция

Пусть в левой части НС изображение размера n x n подвергалось свёртке с помощью фильтра r x r и получалось изображение m x m.

Тогда в симметричном слое НС деконволюция должна изображение m x m с помощью фильтра r x r превращать в изображение n x n.

Причём значения параметров **stride, padding** у свёртки и деконволюции **должны совпадать**.

Пример деконволюции

Изображение:

1111
1001
0110
0000

Фильтр:

010
111
010

stride_x=stride_y=1, padding=0.

Сначала вычисляем размер деконволюции.
Какой должно быть изображение, чтобы при применении фильтра 3x3 получилось бы изображение 4x4?
Ответ: **6x6**.

Изображение:

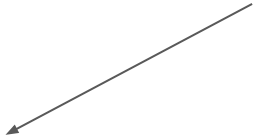
1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

Результат:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



В самом начале деконволюции результат состоит из нулей.

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

Результат:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Перебираем подряд все пиксели исходного изображения.
Пусть **x** — значение очередного пиксела **P**. Домножаем все ячейки фильтра на **x** и прибавляем полученную матрицу к результирующей.
Позиция для прикладывания и суммирования двух матриц определяется так: это центральный пиксел области, которая при обычной свёртке даёт пиксел P в изображении.

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

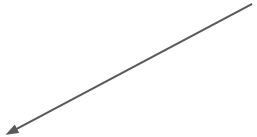
Фильтр:

010
111
010

Результат:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	0	0	0	0
1	1	1	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



Изображение:

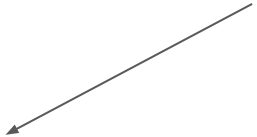
1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

Результат:

0	1	0	0	0	0
1	1	1	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



0	1	1	0	0	0
1	2	2	1	0	0
0	1	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Изображение:

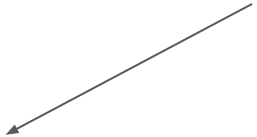
1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

Результат:

0	1	1	0	0	0
1	2	2	1	0	0
0	1	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



0	1	1	1	0	0
1	2	3	2	1	0
0	1	1	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Изображение:

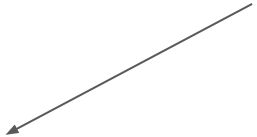
1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

Результат:

0	1	1	1	0	0
1	2	3	2	1	0
0	1	1	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



0	1	1	1	1	0
1	2	3	3	2	1
0	1	1	1	1	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

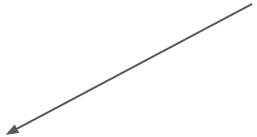
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	2	3	3	2	1
0	1	1	1	1	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

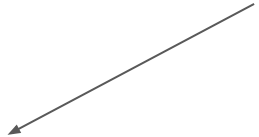
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



тут фильтр умножается на 0,
фактически ничего не происходит

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

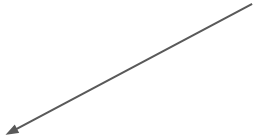
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



тут фильтр тоже умножается на 0,
фактически ничего не происходит

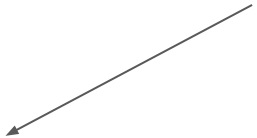
Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

Фильтр:

010
111
010

0	1	1	1	1	0
1	3	3	3	3	1
1	3	2	2	2	1
0	1	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0



Результат:

0	1	1	1	1	0
1	3	3	3	2	1
1	3	2	1	1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

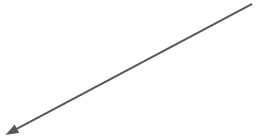
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	3	3	3	3	1
1	3	2	2	2	1
0	1	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	0
1	3	3	3	3	1
1	3	2	2	2	1
0	1	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0



тут фильтр тоже умножается на 0,
фактически ничего не происходит

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

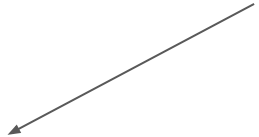
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	3	3	3	3	1
1	3	2	2	2	1
0	1	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	0
1	3	3	3	3	1
1	3	3	2	2	1
0	2	1	1	1	0
0	0	1	0	0	0
0	0	0	0	0	0



тут фильтр тоже умножается на 0,
фактически ничего не происходит

Изображение:

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

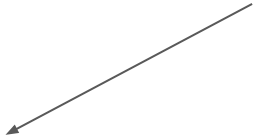
Фильтр:

010
111
010

Результат:

0	1	1	1	1	0
1	3	3	3	3	1
1	3	3	2	2	1
0	2	1	1	1	0
0	0	1	0	0	0
0	0	0	0	0	0

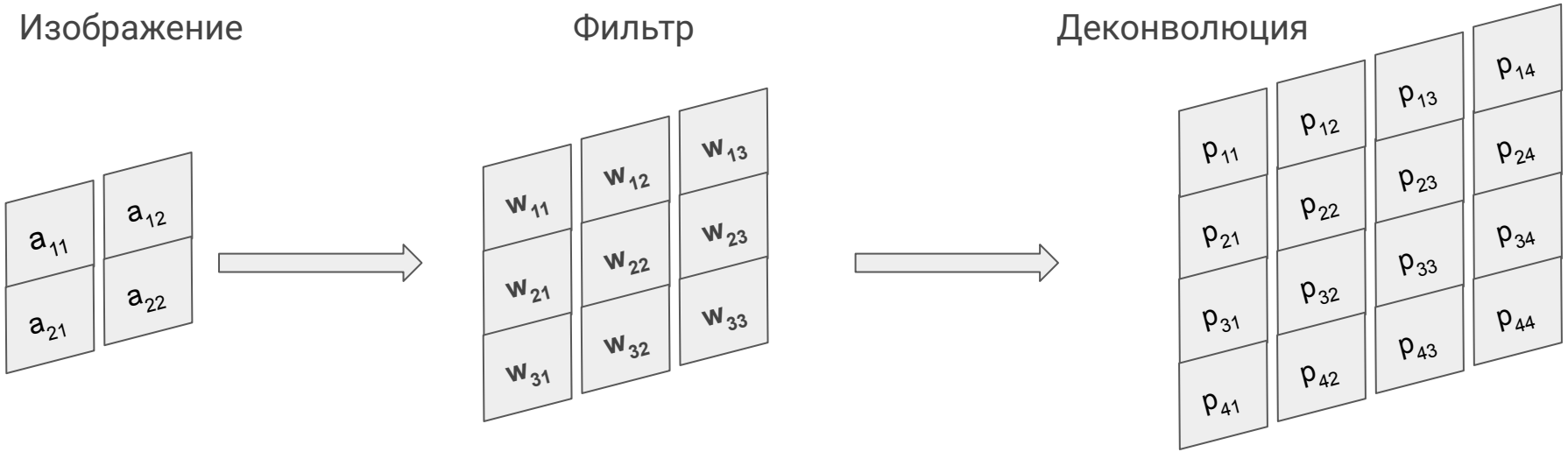
0	1	1	1	1	0
1	3	3	3	3	1
1	3	3	3	2	1
0	2	2	2	2	0
0	0	1	1	0	0
0	0	0	0	0	0



Остальные пиксели изображения нулевые, поэтому результат больше не изменится.

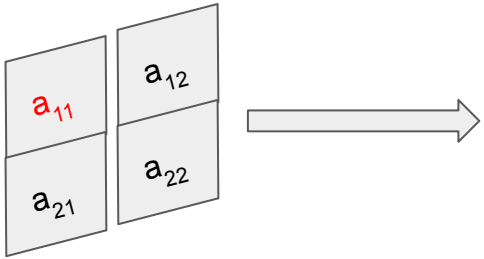
Как тренируется слой деконволюции?

Можно выписать формулы, по которым вычисляется каждый пиксел деконволюции.
Эти формулы зависят от a_{ij} w_{ij}.
Во время тренировки НС частные производные для p_{ij} будут вычисляться через a_{ij} w_{ij}.

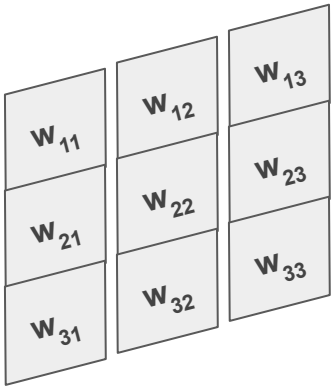


Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

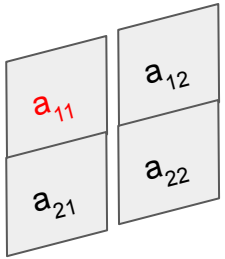


Деконволюция

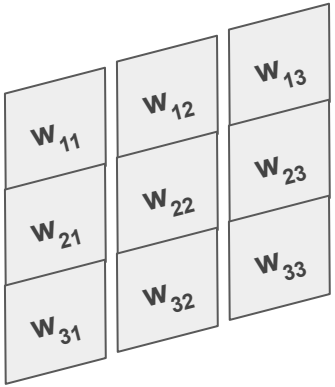
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

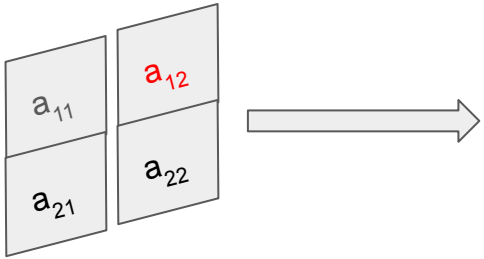


Деконволюция

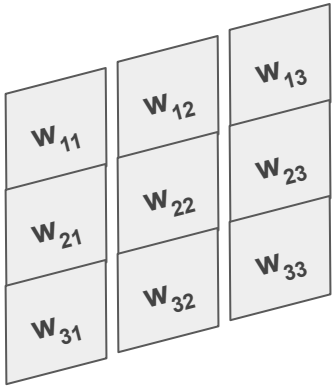
$a_{11}w_{11}$	$a_{11}w_{12}$	$a_{11}w_{13}$	0
$a_{11}w_{21}$	$a_{11}w_{22}$	$a_{11}w_{23}$	0
$a_{11}w_{31}$	$a_{11}w_{32}$	$a_{11}w_{33}$	0
0	0	0	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

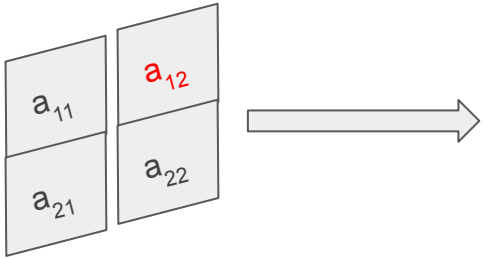


Деконволюция

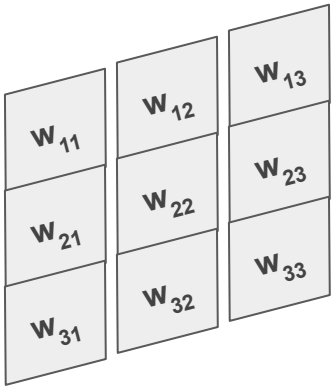
$a_{11}w_{11}$	$a_{11}w_{12}$	$a_{11}w_{13}$	0
$a_{11}w_{21}$	$a_{11}w_{22}$	$a_{11}w_{23}$	0
$a_{11}w_{31}$	$a_{11}w_{32}$	$a_{11}w_{33}$	0
0	0	0	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

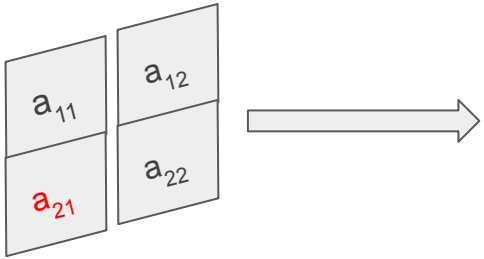


Деконволюция

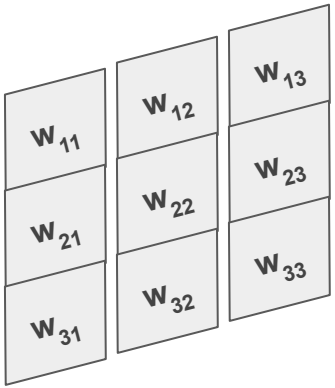
$a_{11}w_{11}$	$a_{11}w_{12}+a_{12}w_{11}$	$a_{11}w_{13}+a_{12}w_{12}$	$a_{12}w_{13}$
$a_{11}w_{21}$	$a_{11}w_{22}+a_{12}w_{21}$	$a_{11}w_{23}+a_{12}w_{22}$	$a_{12}w_{23}$
$a_{11}w_{31}$	$a_{11}w_{32}+a_{12}w_{31}$	$a_{11}w_{33}+a_{12}w_{32}$	$a_{12}w_{33}$
0	0	0	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

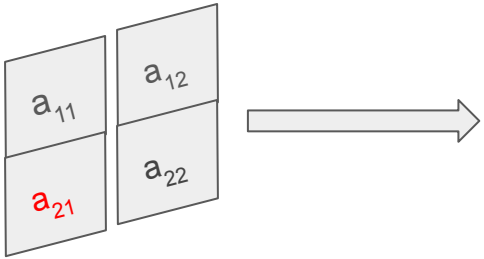


Деконволюция

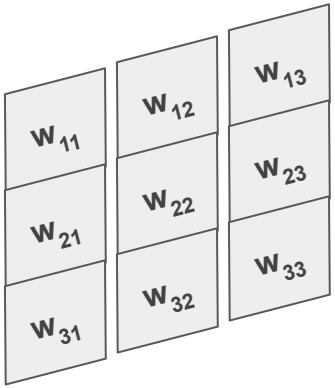
$a_{11}w_{11}$	$a_{11}w_{12}+a_{12}w_{11}$	$a_{11}w_{13}+a_{12}w_{12}$	$a_{12}w_{13}$
$a_{11}w_{21}$	$a_{11}w_{22}+a_{12}w_{21}$	$a_{11}w_{23}+a_{12}w_{22}$	$a_{12}w_{23}$
$a_{11}w_{31}$	$a_{11}w_{32}+a_{12}w_{31}$	$a_{11}w_{33}+a_{12}w_{32}$	$a_{12}w_{33}$
0	0	0	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

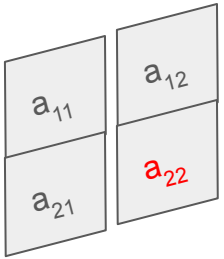


Деконволюция

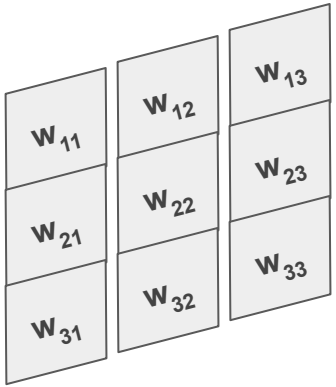
$a_{11}w_{11}$	$a_{11}w_{12}+a_{12}w_{11}$	$a_{11}w_{13}+a_{12}w_{12}$	$a_{12}w_{13}$
$a_{11}w_{21}+a_{21}w_{11}$	$a_{11}w_{22}+a_{12}w_{21}+a_{21}w_{12}$	$a_{11}w_{23}+a_{12}w_{22}+a_{21}w_{13}$	$a_{12}w_{23}$
$a_{11}w_{31}+a_{21}w_{21}$	$a_{11}w_{32}+a_{12}w_{31}+a_{21}w_{22}$	$a_{11}w_{33}+a_{12}w_{32}+a_{21}w_{23}$	$a_{12}w_{33}$
$a_{21}w_{31}$	$a_{21}w_{32}$	$a_{21}w_{33}$	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



Фильтр

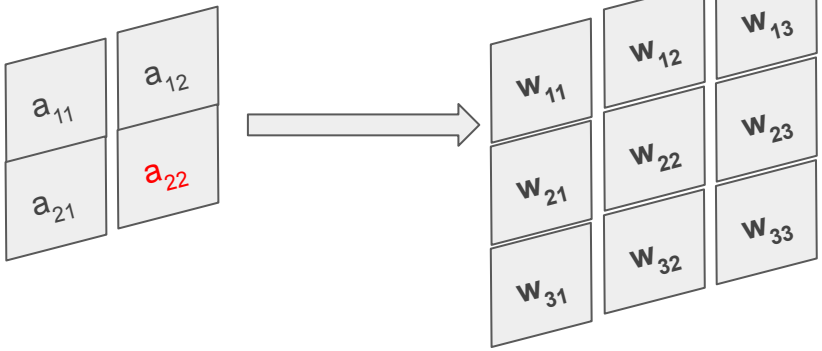


Деконволюция

$a_{11}w_{11}$	$a_{11}w_{12}+a_{12}w_{11}$	$a_{11}w_{13}+a_{12}w_{12}$	$a_{12}w_{13}$
$a_{11}w_{21}+a_{21}w_{11}$	$a_{11}w_{22}+a_{12}w_{21}+a_{21}w_{12}$	$a_{11}w_{23}+a_{12}w_{22}+a_{21}w_{13}$	$a_{12}w_{23}$
$a_{11}w_{31}+a_{21}w_{21}$	$a_{11}w_{32}+a_{12}w_{31}+a_{21}w_{22}$	$a_{11}w_{33}+a_{12}w_{32}+a_{21}w_{23}$	$a_{12}w_{33}$
$a_{21}w_{31}$	$a_{21}w_{32}$	$a_{21}w_{33}$	0

Деконволюция в общем виде (BDSM, 18+)

Изображение



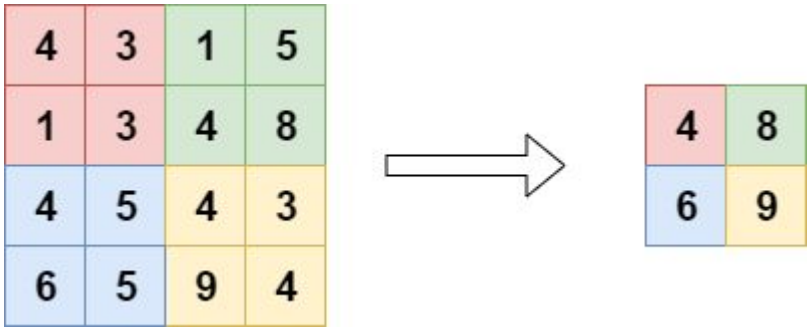
Фильтр

Деконволюция

$a_{11}w_{11}$	$a_{11}w_{12}+a_{12}w_{11}$	$a_{11}w_{13}+a_{12}w_{12}$	$a_{12}w_{13}$
$a_{11}w_{21}+a_{21}w_{11}$	$a_{11}w_{22}+a_{12}w_{21}+a_{21}w_{12}+a_{22}w_{11}$	$a_{11}w_{23}+a_{12}w_{22}+a_{21}w_{13}+a_{22}w_{12}$	$a_{12}w_{23}+a_{22}w_{13}$
$a_{11}w_{31}+a_{21}w_{21}$	$a_{11}w_{32}+a_{12}w_{31}+a_{21}w_{22}+a_{22}w_{21}$	$a_{11}w_{33}+a_{12}w_{32}+a_{21}w_{23}+a_{22}w_{22}$	$a_{12}w_{33}+a_{22}w_{23}$
$a_{21}w_{31}$	$a_{21}w_{32}+a_{22}w_{31}$	$a_{21}w_{33}+a_{22}w_{32}$	$a_{22}w_{33}$

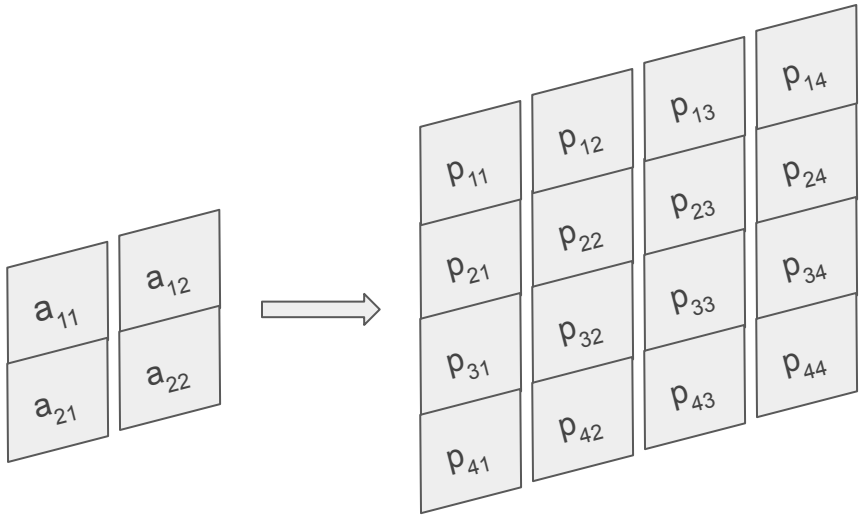
Операция unpooling

Пулинг уменьшает размерность

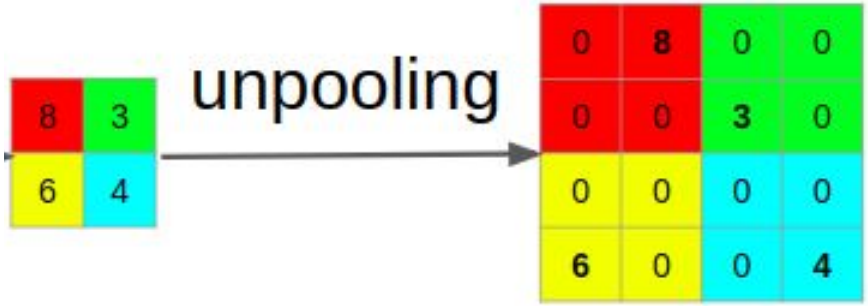


Следовательно, **unpooling должен увеличивать изображение.**

А откуда взять дополнительную информацию о новых пикселах?

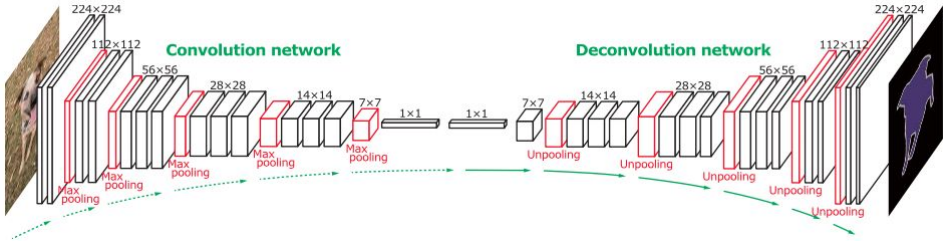


Забейте новые пиксели нулями!



Но по какому правилу определить, в какой пиксел ставить число с предыдущего слоя?

А тут нужно вспомнить, что сегментирующая НС симметричная!!!

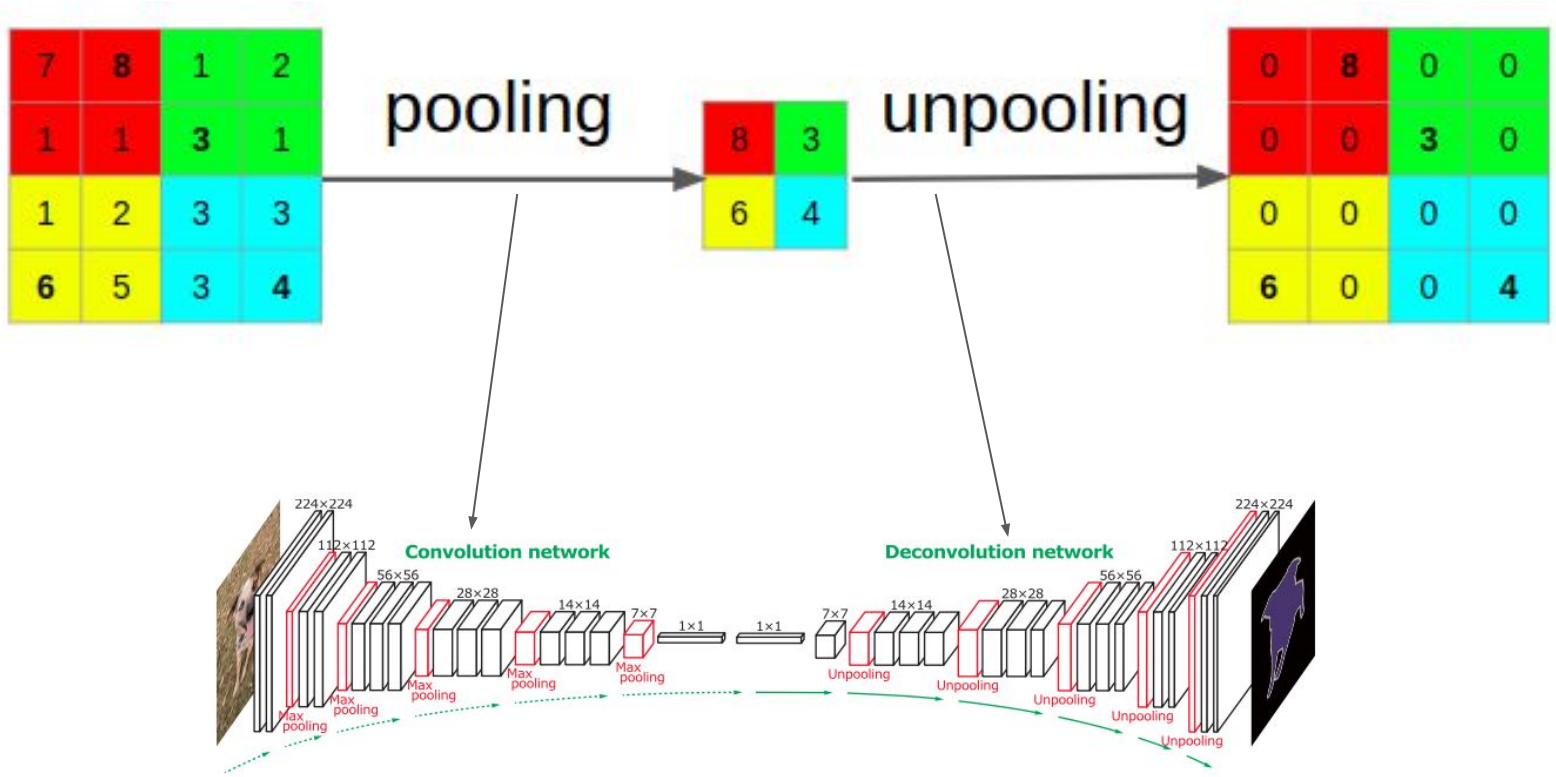


Нужна информация из симметричного слоя пулинга

Если в симметричном слое пулинга (из первой части НС) максимум достигался в ячейке с координатами i,j, то в соотвтествующем слое unpooling-а число ставится в ячейку с координатами i,j.

В остальные ячейки ставится 0.

То есть произойдет примерно так



Как это выразить в виде формул?

Допустим для простоты, что слои пулинга и анпулинга стоят рядом друг с другом. По каким формулам вычисляются элементы q_{ij}?

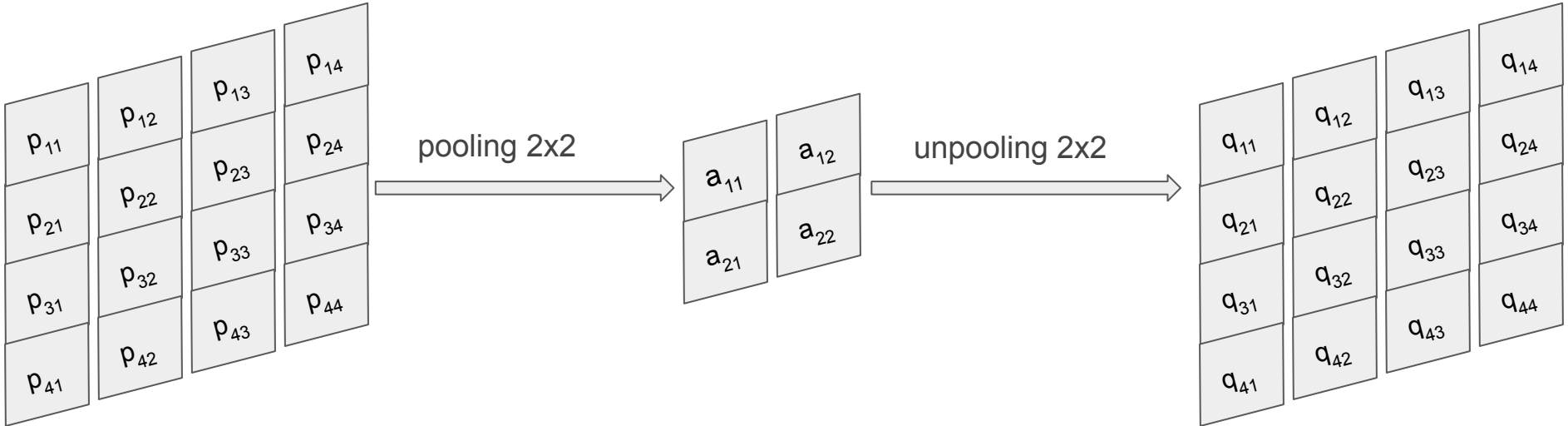
Для пулинга имеем:

$a_{11}=\max(p_{11},p_{12},p_{21},p_{22})$

$a_{12}=\max(p_{13},p_{14},p_{23},p_{24})$

$a_{21}=\max(p_{31},p_{32},p_{41},p_{42})$

$a_{22}=\max(p_{33},p_{34},p_{43},p_{44})$



Как это выразить в виде формул?

$a_{11}=\max(p_{11},p_{12},p_{21},p_{22})$

$a_{12}=\max(p_{13},p_{14},p_{23},p_{24})$

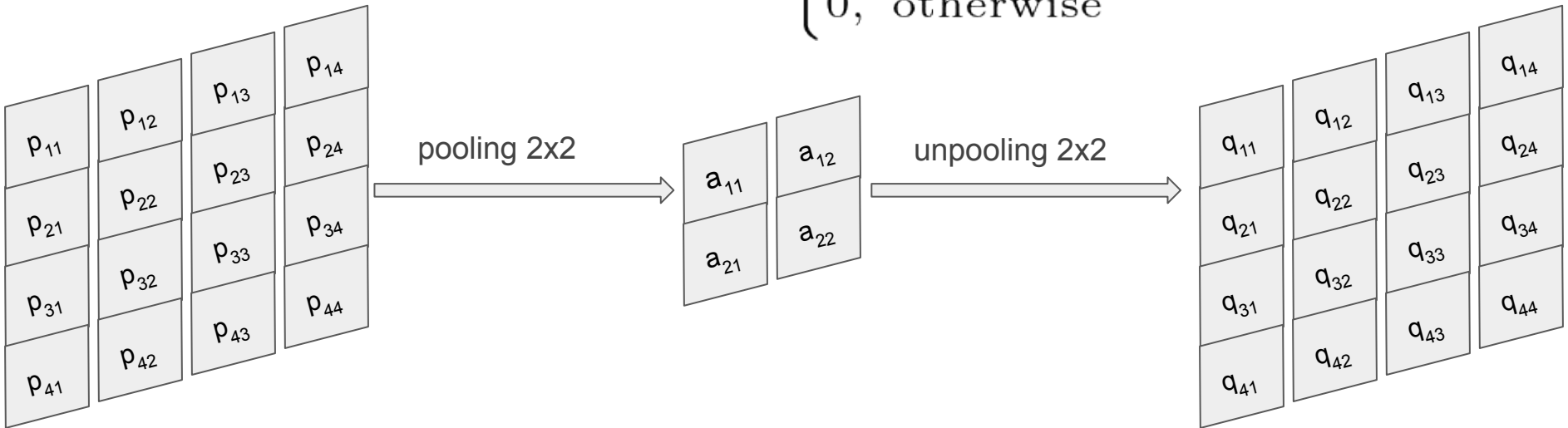
$a_{21}=\max(p_{31},p_{32},p_{41},p_{42})$

$a_{22}=\max(p_{33},p_{34},p_{43},p_{44})$

$$q_{11} = \begin{cases} p_{11}, & \text{if } p_{11} = a_{11}, \\ 0, & \text{otherwise} \end{cases}$$

...

$$q_{44} = \begin{cases} p_{44}, & \text{if } p_{44} = a_{22}, \\ 0, & \text{otherwise} \end{cases}$$



А можно ли применить ГС к функциям подобного типа?

Давайте сделаем два шага ГС у следующей функции

$$f(w_1, w_2, w_3) = \begin{cases} w_1, & \text{if } w_1 = \max(w_1, w_2, w_3), \\ 0, & \text{otherwise} \end{cases}$$

начиная с точки (1.1 , 1.05 , 1) и с шагом спуска h=0.1.

В этой точке функция f равна выражению: $f(w_1,w_2,w_3)=w_1$.

Следовательно, её частные производные равны:
$\partial f/\partial w_1=1$, $\partial f/\partial w_2=0$, $\partial f/\partial w_3=0$.

Тогда следующая точка равна (1, 1.05, 1).

А можно ли применить ГС к функциям подобного типа?

$$f(w_1, w_2, w_3) = \begin{cases} w_1, & \text{if } w_1 = \max(w_1, w_2, w_3), \\ 0, & \text{otherwise} \end{cases}$$

В новой точке (1, 1.05, 1) функция по-прежнему равна выражению $f(w_1,w_2,w_3)=w_1$.

Как и ранее:
$\partial f/\partial w_1=1$, $\partial f/\partial w_2=0$, $\partial f/\partial w_3=0$.

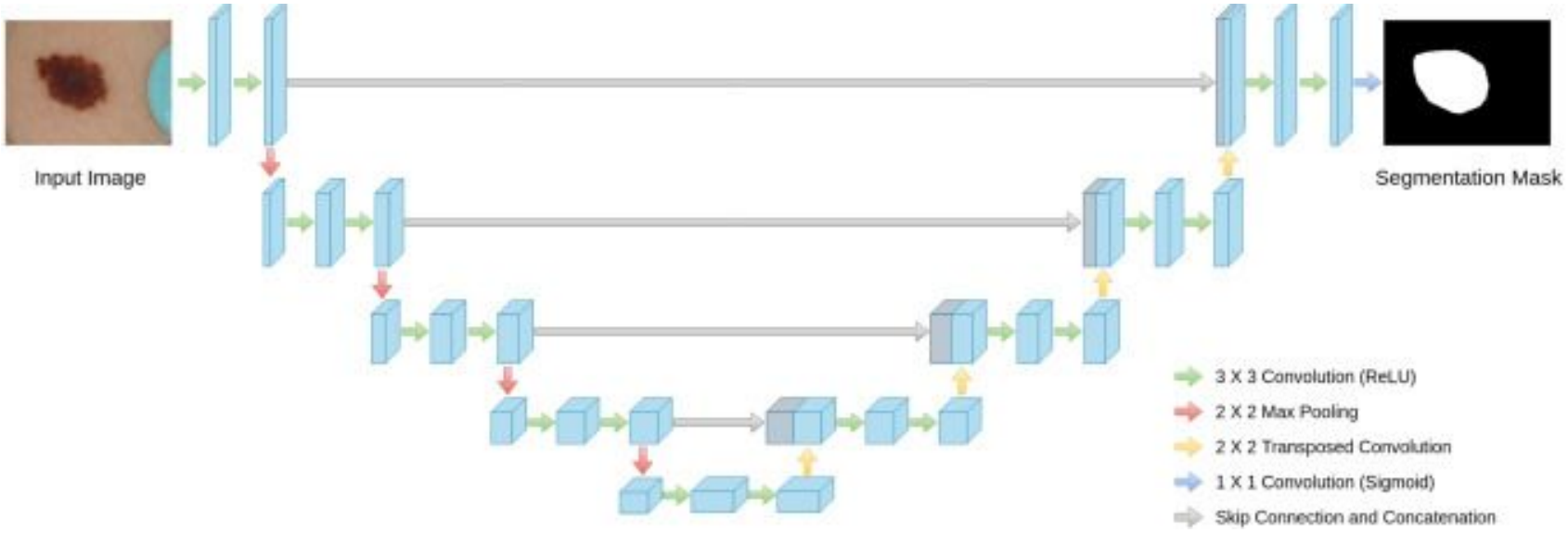
Тогда следующая точка равна (0.9, 1.05, 1).

А в этой точке функция (и все её ЧП) уже равна 0, и ГС остановится.

Достижения и следствия из сегментации

U-net (2015)

Это мощная НС для задачи сегментации.
В ней используется принцип как **Hourglass architecture**, так и **связи между далёкими слоями** (как в СНС ResNet)



Задачи, которые следуют из сегментации

Допустим, что вы научились хорошо сегментировать изображения. Какие новые задачи, связанные с обработкой изображения, вы можете ещё решить?

1. **Выделение границ изображения.**

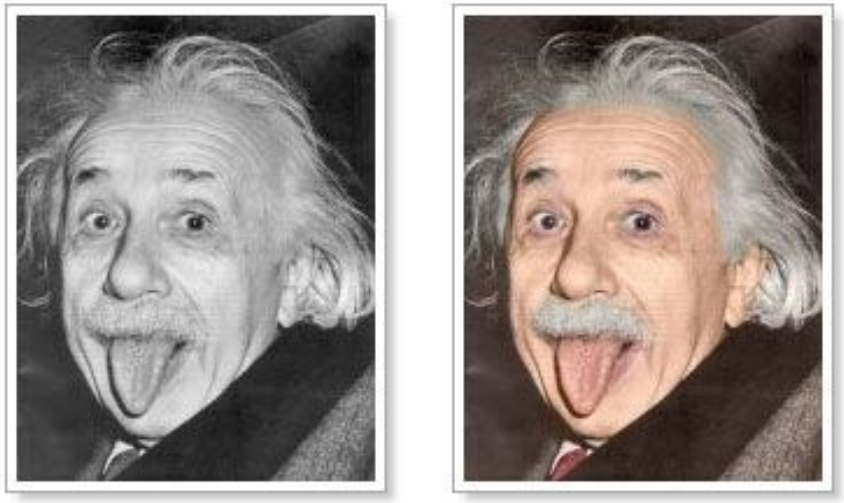
При чём тут сегментация?

Фактически мы должны сегментировать изображение на 2 области: «то, что является границей» и «всё остальное».

Задачи, которые следуют из сегментации

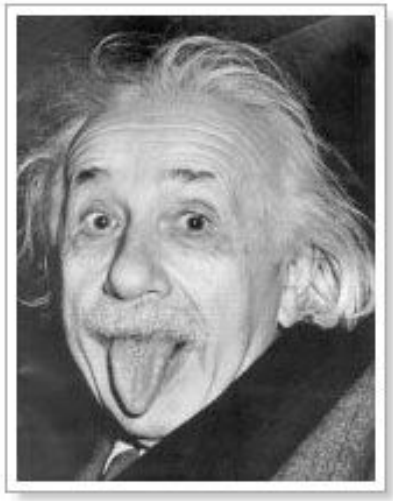
Допустим, что вы научились хорошо сегментировать изображения. Какие новые задачи, связанные с обработкой изображения, вы можете ещё решить?

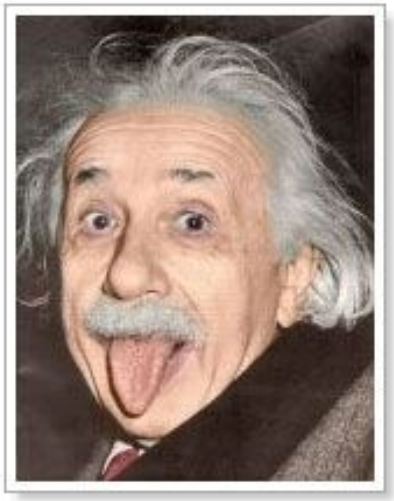
2. **Раскрашивание ч/б изображений.**



При чём тут сегментация?

Это тоже сегментация! Фактически мы должны выделить на картинке области «красного», «зелёного», «синего», «коричневого» и т.д. То есть разделить на сегменты по цветам.





Тренировка сегментирующей НС

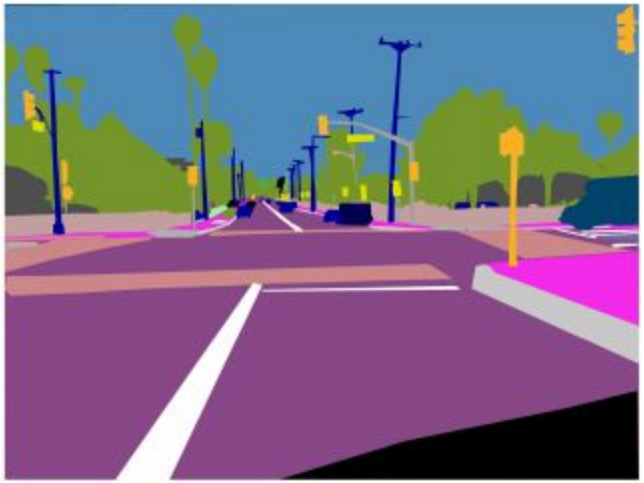
Как она тренируется?

С ТВ всё вроде как понятно.

Это пары (картинка, сегментированная картинка).

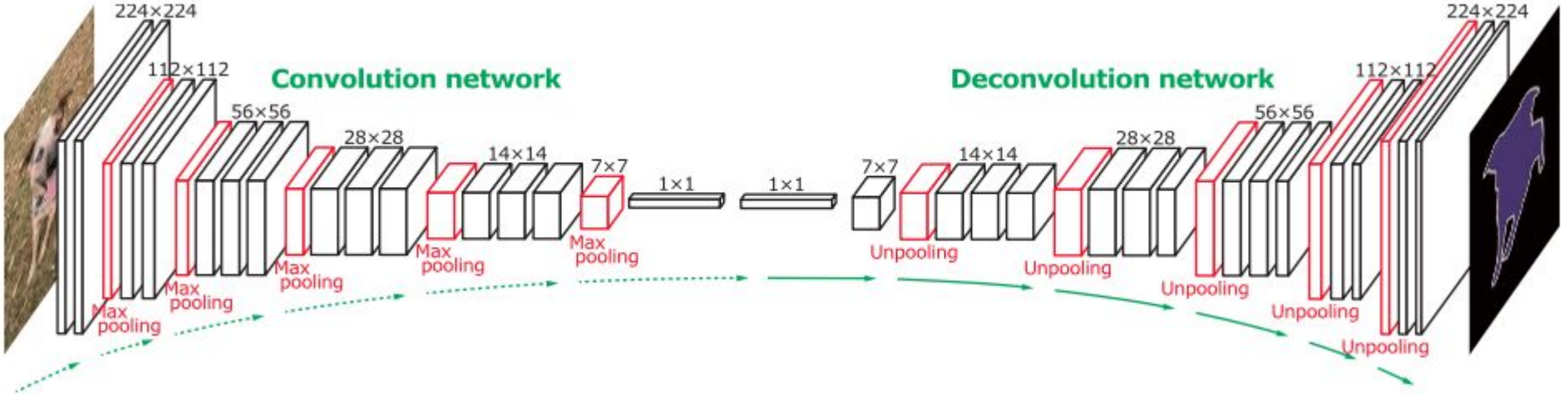
X

Y

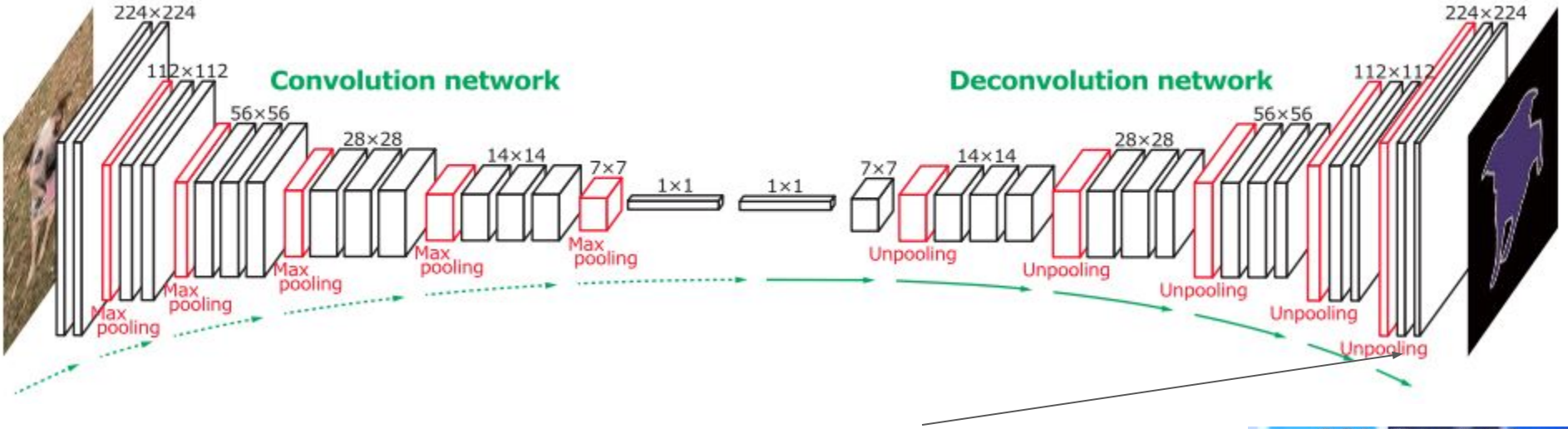


Но какова тут функция потерь?

Как вычисляется ошибка НС при сегментации изображения?



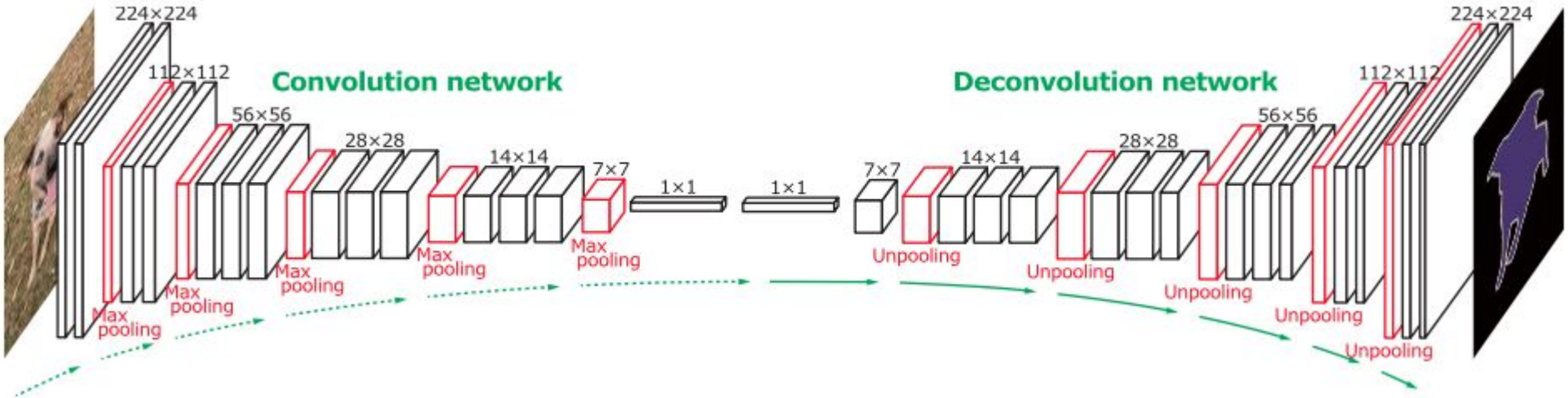
Но какова тут функция потерь?



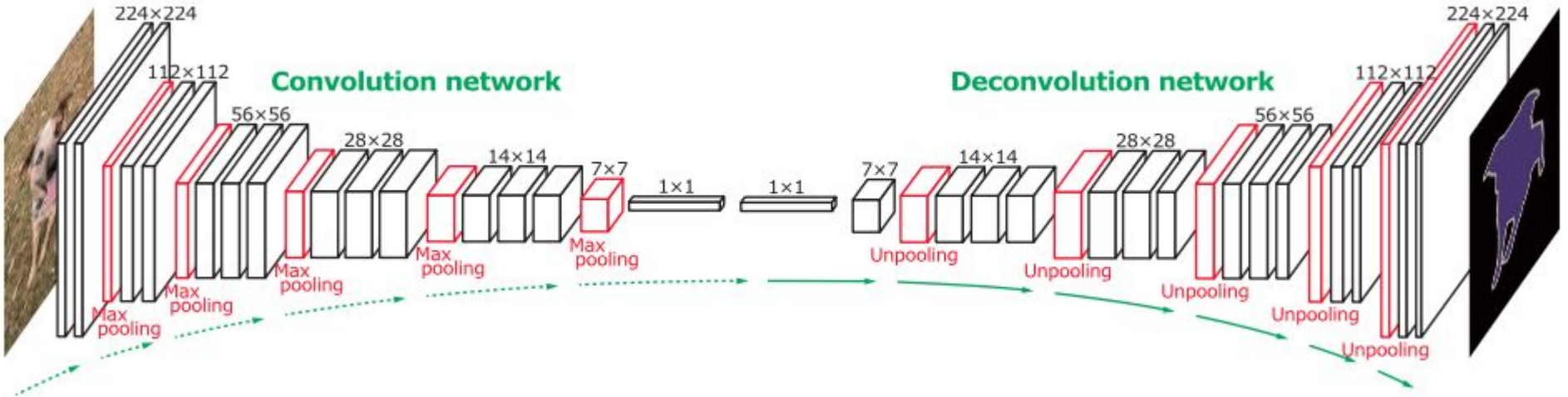
У этой сегментирующей сети в последнем слое 2 канала,
и сегментация состоит тоже из 2х областей?

Первое правило: количество каналов на последнем слое сегментирующей равно максимальному количеству различных сегментов, которые могут быть детектированы НС.

Например, если на последнем слое 2 канала, то НС может разделять изображения максимум на 2 области.

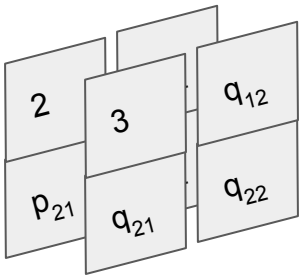


Второе правило: пиксель с координатами (i,j) из k-го канала последнего слоя НС соответствует вероятности принадлежности пиксела исходного изображения с координатами (i,j) k-й области.

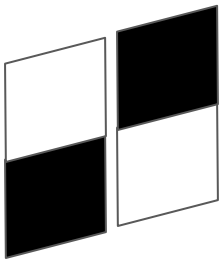


Например, если на последнем слое 2 канала, и пиксел (1,1) в первом канале имеет значение 2, а во втором канале 3, то это означает, что вероятность того, что на изображении пиксел (1,1) с вероятностью $e^2/(e^2+e^3)=0.27$ принадлежит первой области, и с вероятностью $e^3/(e^2+e^3)=0.73$ принадлежит второй области.

2 канала



сегментированное изображение (выход НС)



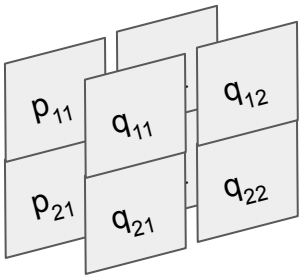
Ну да: к каналам последнего слоя применяется softmax, и числа превращаются в вероятности принадлежности сегментам.

Функция потерь в задаче сегментации

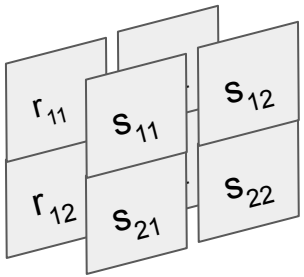
Поскольку сегментация — это фактически задача классификации каждого пиксела изображения, то для каждого объекта из ТВ можно вычислить кросс-энтропию по его пикселам, а потом суммировать кросс-энтропии по всем объектам из ТВ.

Поясним на примере…

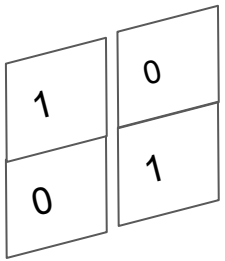
2 канала



softmax



Правильный ответ для объекта из ТВ



Функция потерь в задаче сегментации

r_{ij} - вероятность принадлежности пиксела классу 0

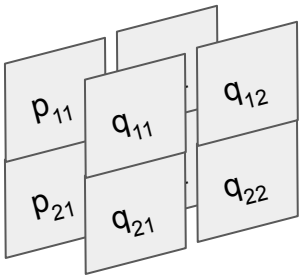
s_{ij} - вероятность принадлежности пиксела классу 1

Кросс-энтропия для данного объекта ТВ равна: $-\ln s_{11} - \ln r_{12} - \ln r_{21} - \ln s_{22}$

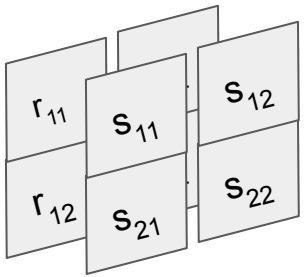
Осталось просуммировать по всем объектам ТВ.

И это выражение минимизировать.

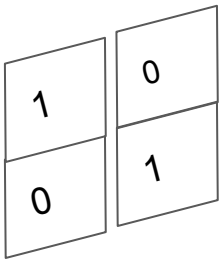
2 канала



softmax



Правильный ответ для объекта из ТВ



Выводы

- Мы рассмотрели задачу сегментации.
- Познакомились со специальной архитектурой НС для решения задачи сегментации.
- Рассмотрели операции деконволюции и анпулинга.
- Выписали функцию потерь в задаче сегментации.