

онлайн-курс

СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

(с) ОмГТУ, 2022

1. Распознавание картинок: нейросети, фильтры и свёртки

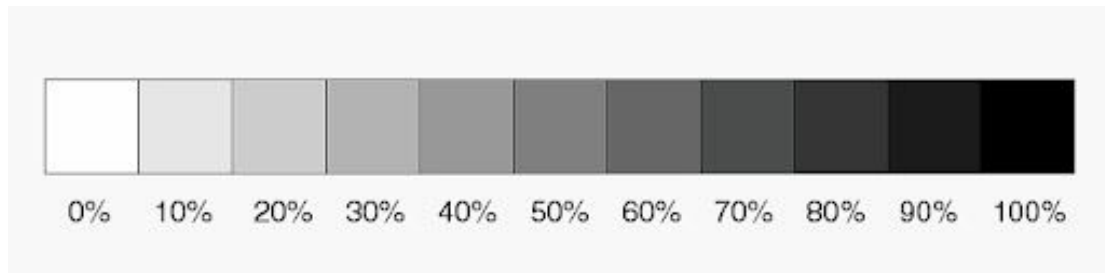
Картинки как объект для нейронных сетей

Как подать картинку на вход нейросети?

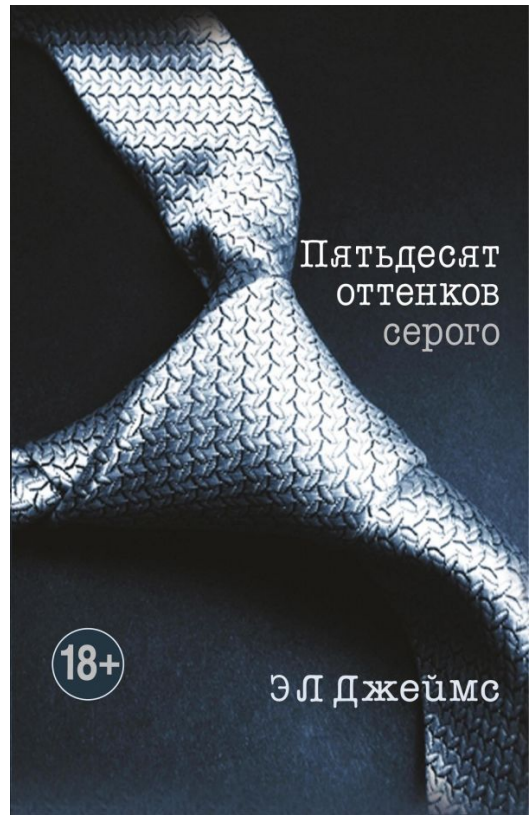
Надо её представить в виде массива (матрицы) чисел!

Это делается так.

Сначала рассмотрим **монохромные картинки**.
В них каждый пиксел содержит оттенки серого.

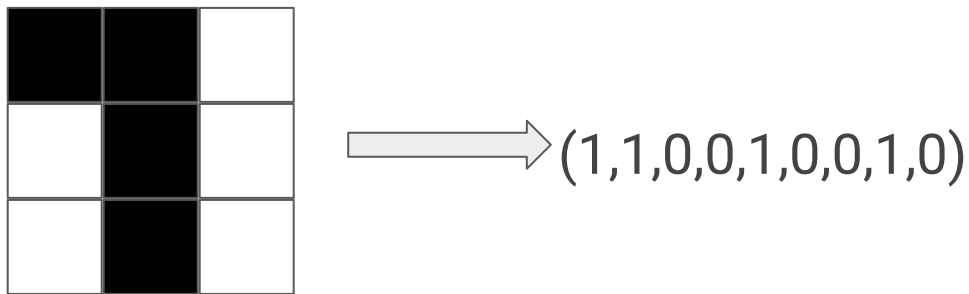


Логичнее каждый пиксел представить числом от 0 (белый) до 1 (черный).



Монохромная картинка для нейросети

После этого мы картинку превращаем в массив чисел:


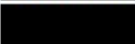

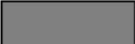













и подаём в качестве значений входного слоя НС.

Цветная картинка для нейросети

Цветную картинку тоже легко превратить в массив чисел. Для этого нужно вспомнить, что такое формат RGB.

Фактически каждый пиксел — это **тройка** целых чисел из интервала $[0,255]$.

Named	Numeric	Color name	Hex rgb	Decimal
		<i>black</i>	#000000	0,0,0
		<i>silver</i>	#C0C0C0	192,192,192
		<i>gray</i>	#808080	128,128,128
		<i>white</i>	#FFFFFF	255,255,255
		<i>maroon</i>	#800000	128,0,0
		<i>red</i>	#FF0000	255,0,0
		<i>purple</i>	#800080	128,0,128
		<i>fuchsia</i>	#FF00FF	255,0,255
		<i>green</i>	#008000	0,128,0
		<i>lime</i>	#00FF00	0,255,0
		<i>olive</i>	#808000	128,128,0
		<i>yellow</i>	#FFFF00	255,255,0
		<i>navy</i>	#000080	0,0,128
		<i>blue</i>	#0000FF	0,0,255
		<i>teal</i>	#008080	0,128,128
		<i>aqua</i>	#00FFFF	0,255,255

Цветная картинка для нейросети

Фактически каждый пиксел — это **тройка** целых чисел из интервала $[0,255]$.

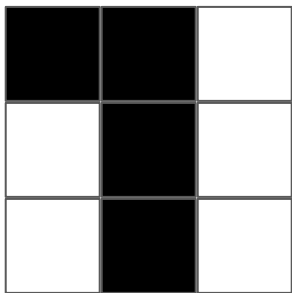
А сама картинка — это набор из трёх числовых таблиц (**18+**: матриц).

Все эти таблицы можно «вытянуть» в линию, получится один большой вектор. Его и подаём на вход в нейросеть.

		165	187	209	58	7
	14	125	233	201	98	159
253	144	120	251	41	147	204
67	100	32	241	23	165	30
209	118	124	27	59	201	79
210	236	105	169	19	218	156
35	178	199	197	4	14	218
115	104	34	111	19	196	
32	69	231	203	74		

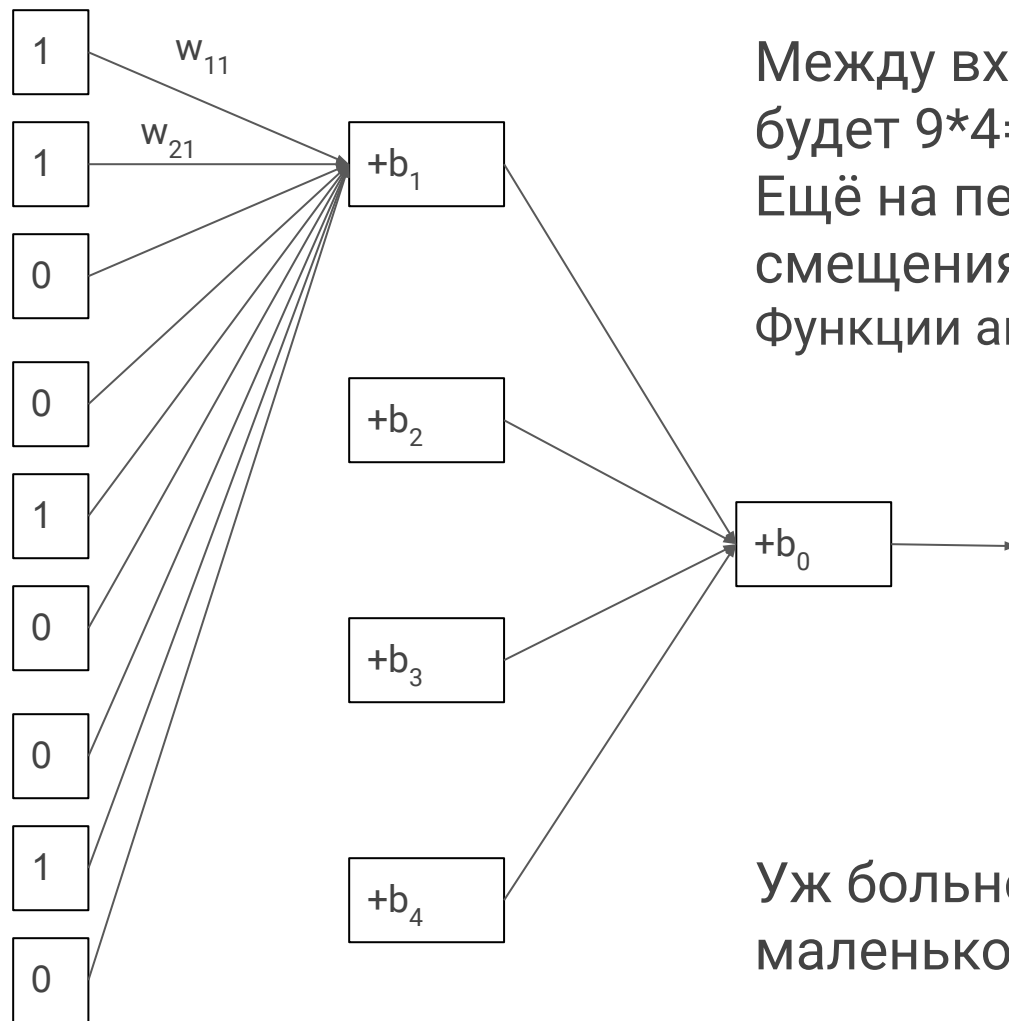
Картинка идёт в...

... нейросеть.



→ (1,1,0,0,1,0,0,1,0)

Мы её подаём на входной слой нейросети (в качестве примера на следующем слайде возьмем полносвязную нейросеть с одним внутренним слоем).



Между входным и первым слоем НС будет $9 \times 4 = 36$ весов-связей. Ещё на первом слое НС будет 4 веса-смещения. Функции активации на рисунке не указаны.

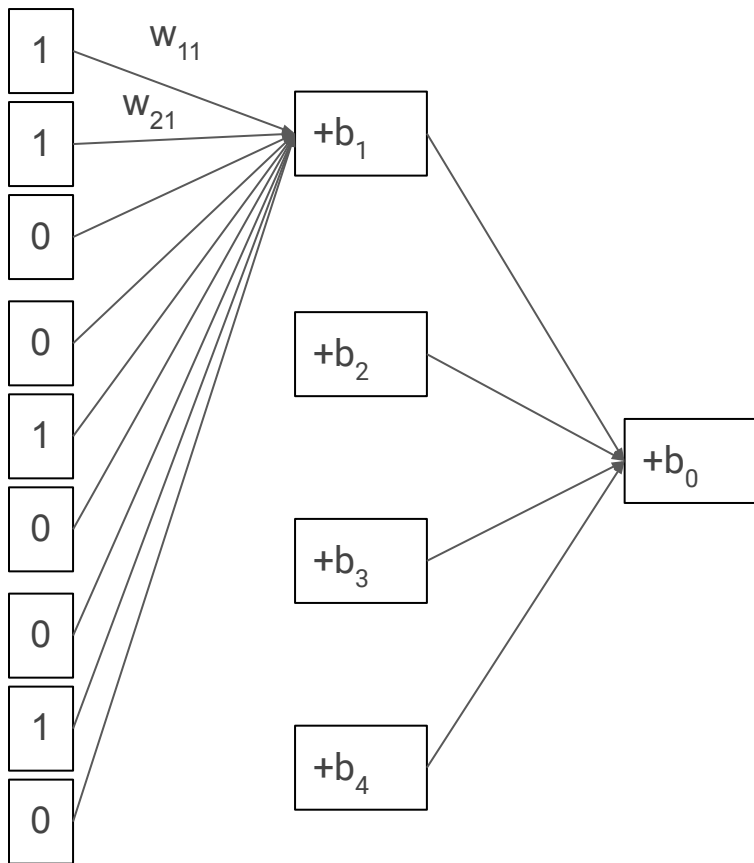
Уж больно много весов для такой маленькой картинке!

Проблемы

На самом деле **стандартная архитектура полносвязной нейросети** неудобна для распознавания картинок!



Экономим веса



Многие связи между нейронами последней НС **избыточны**.

Это связано с тем, что практических задачах нейрону не нужно знать информацию обо **всей картинке**.

Что делать?

Спойлер:

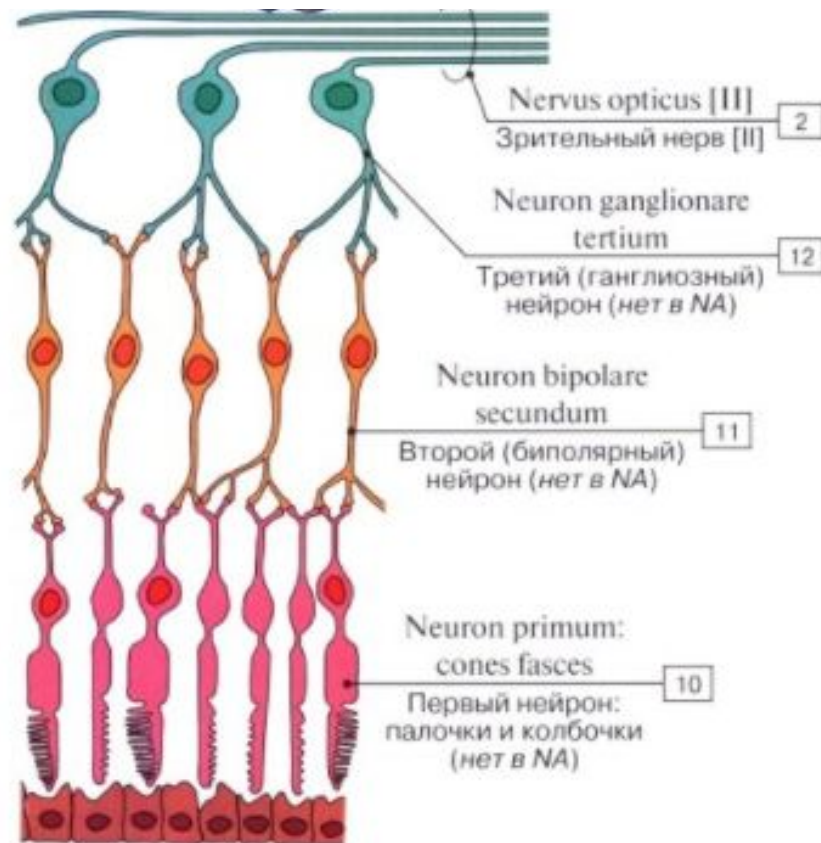
достаточно картинку разделить на области и каждому нейрону дать небольшую область для изучения.

Свёрточная архитектура

А что у человека в голове?

По какому принципу работают клетки в зрительном центре биологического мозга?

Ни один из нейронов нижних отделов зрительного центра не имеет информации обо всём изображении.

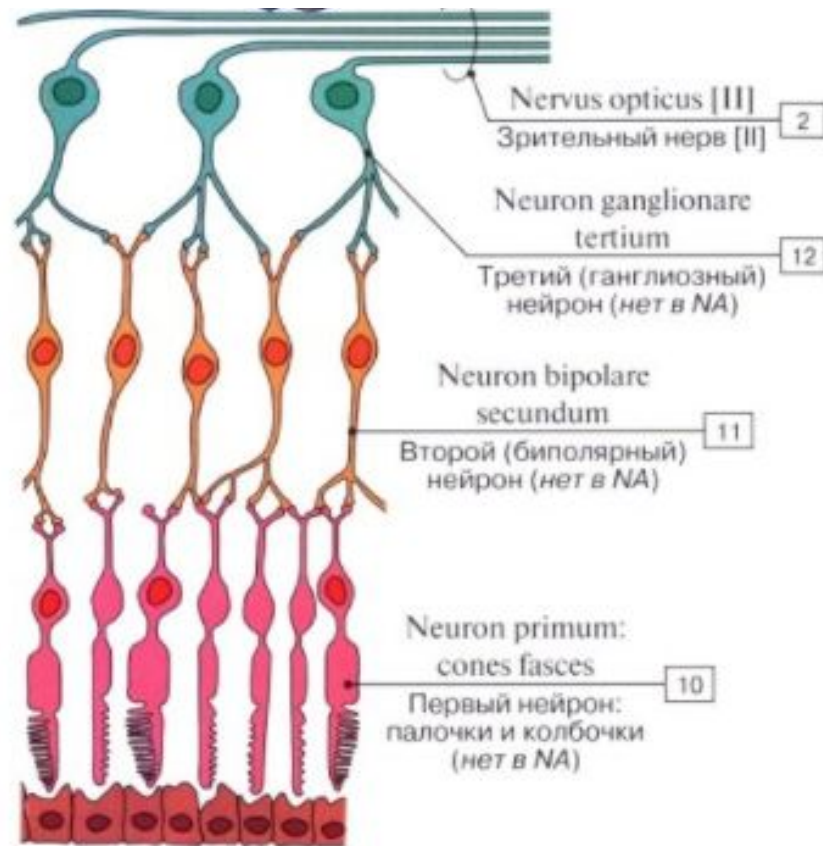


Идея свёрточной архитектуры...

... идёт из анатомии.

Каждый слой нейросети будет не одномерным (в виде столбика), а двумерным (в виде таблицы).

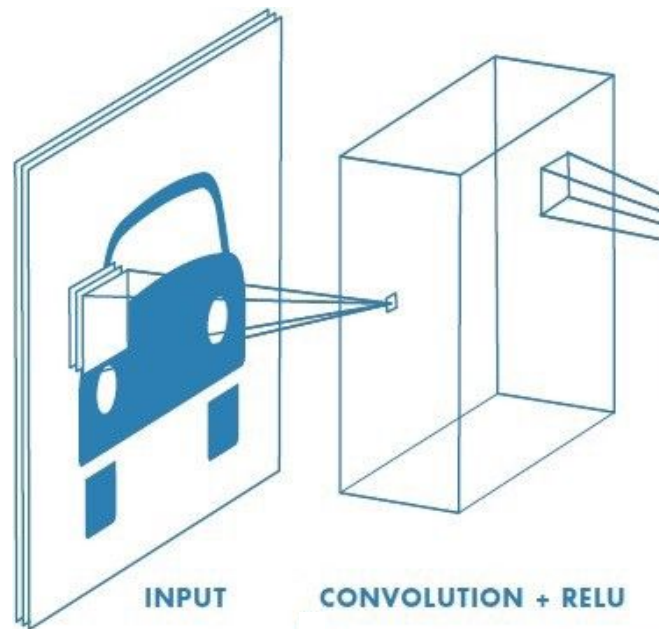
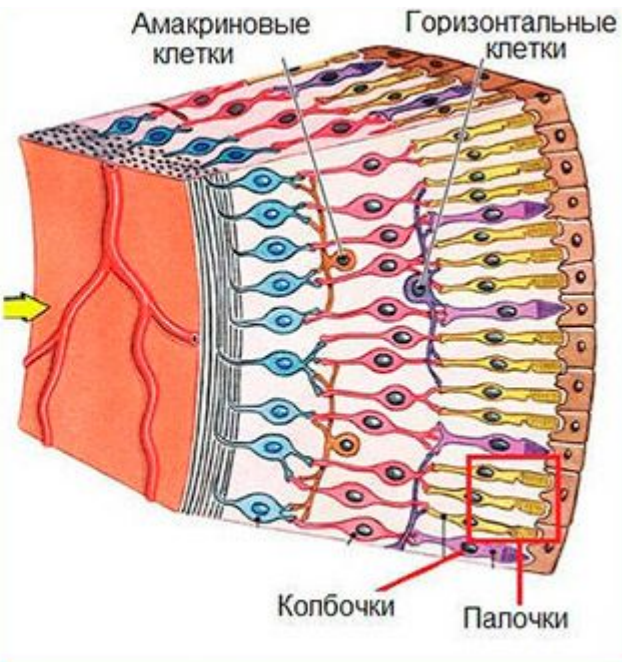
Ибо сетчатка и слои (!) зрительного центра двумерные.



Слева — изображение сетчатки и первых слоёв зрительного центра.

Справа — архитектура нейросети по распознаванию изображений.

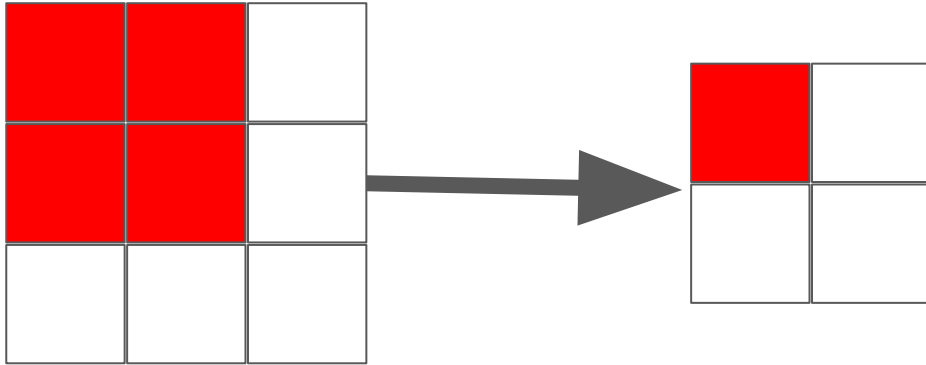
Смотри не перепутай.



Свёрточная архитектура

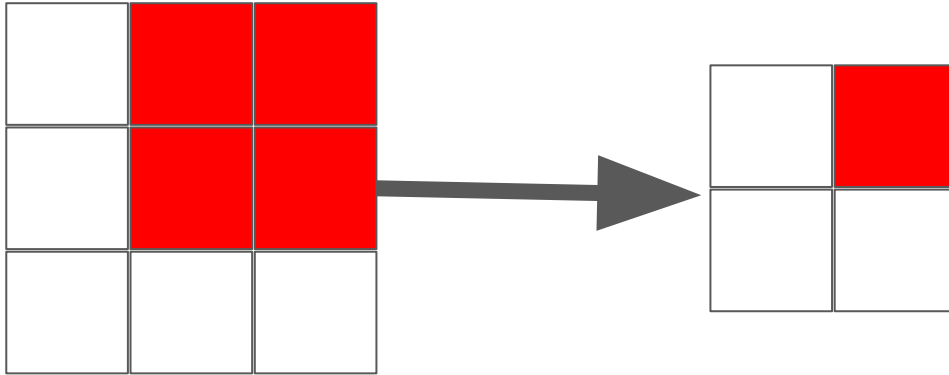
Все слои свёрточной нейросети (СНС) представляются в виде матриц. Каждый нейрон последующего слоя соединён только с нейронами определённой области предыдущего слоя.

Размер области определяется при создании СНС. В нашем примере размеры области 2x2.



Свёрточная архитектура

Все слои СНС представляются в виде матриц. Каждый нейрон последующего слоя соединен только с нейронами определённой области предыдущего слоя. Размер области определяется при создании СНС. В нашем примере размеры области 2x2.

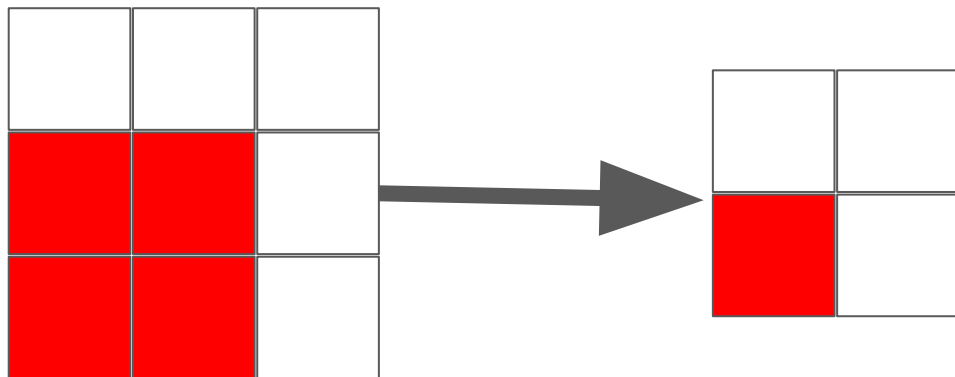


Нейроны второго слоя могут иметь пересекающиеся области с предыдущего слоя.



Свёрточная архитектура

Все слои СНС представляются в виде матриц. Каждый нейрон последующего слоя соединен только с нейронами определённой области предыдущего слоя. Размер области определяется при создании СНС. В нашем примере размеры области 2x2.

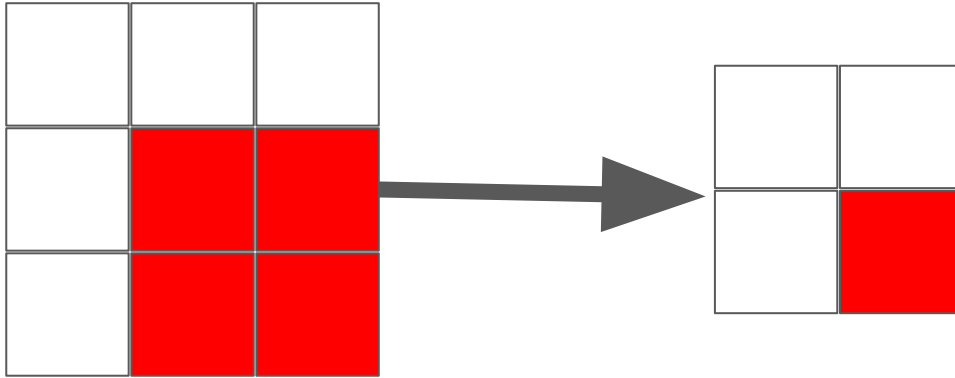


Нейроны второго слоя могут иметь пересекающиеся области с предыдущего слоя.



Свёрточная архитектура

Все слои СНС представляются в виде матриц. Каждый нейрон последующего слоя соединен только с нейронами определённой области предыдущего слоя. Размер области определяется при создании СНС. В нашем примере размеры области 2x2.



Нейроны второго слоя могут иметь пересекающиеся области с предыдущего слоя.

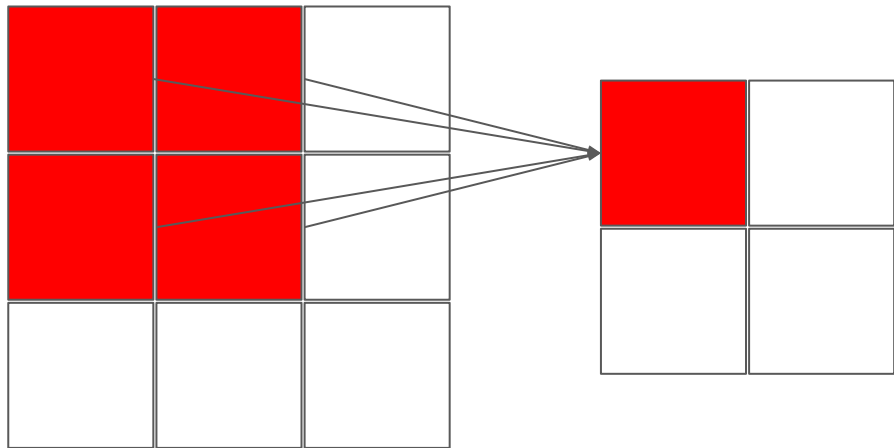


Сколько всего весов в свёрточном слое?

Вот так и выглядит **свёрточный слой**.

Он совершенно **не похож** на полносвязный.

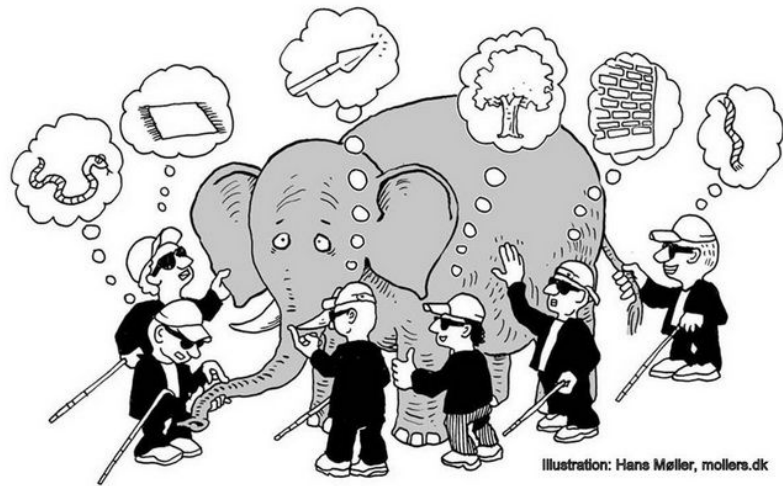
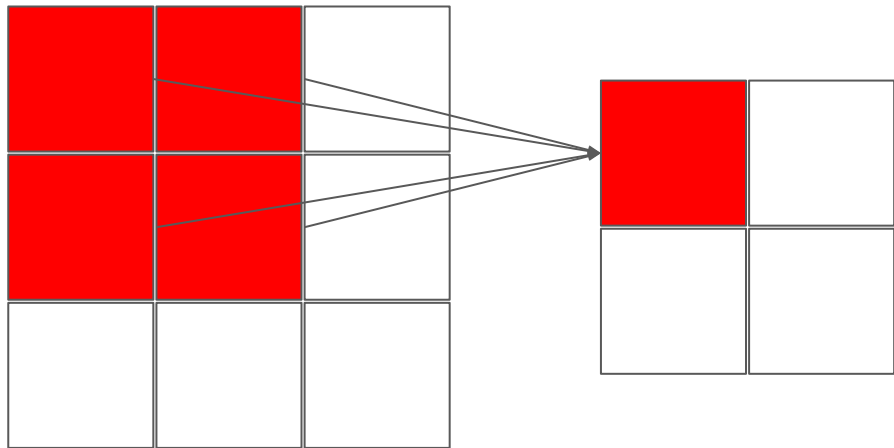
Связи в нейрон следующего слоя ведут лишь из выделенной области предыдущего слоя.



Сколько всего весов в свёрточном слое?

Связи в нейрон следующего слоя ведут лишь из выделенной области предыдущего слоя.

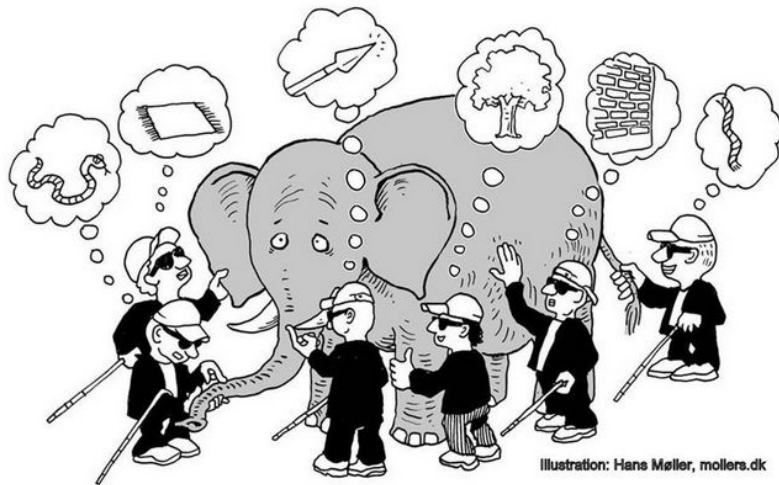
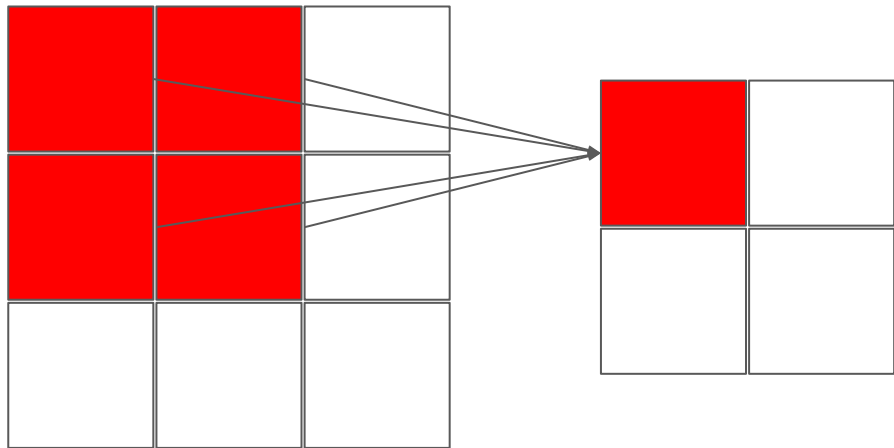
Нейроны в свёрточном слое — это как слепые мудрецы, ощупывающие слона!



Сколько всего весов в свёрточном слое?

Проблема слепых мудрецов: нет «мудреца 2го уровня», который бы смог обработать информацию от каждого мудреца.

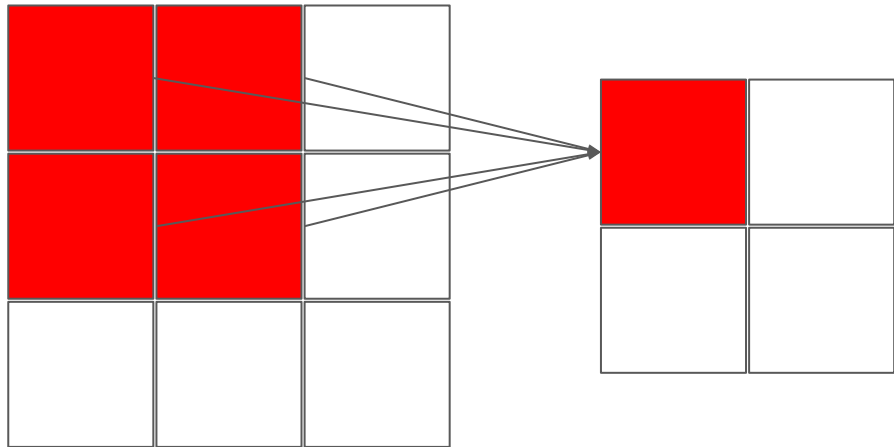
А в нейросетях мы можем добавить ещё один слой, анализирующий информацию с предыдущего слоя (об этом позже).



Сколько всего весов в свёрточном слое?

Если размер области 2×2 , то будет $2 \times 2 = 4$ весов связей у каждого нейрона второго слоя.

Поскольку во втором слое всего 4 нейрона, то получается, что будет $4 \times 4 = 16$ весов связей между свёрточными слоями.



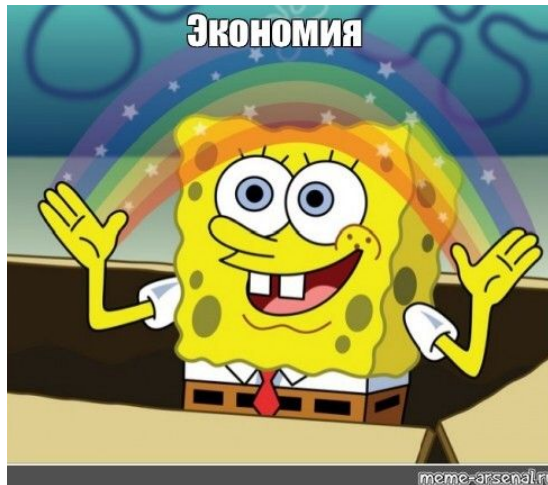
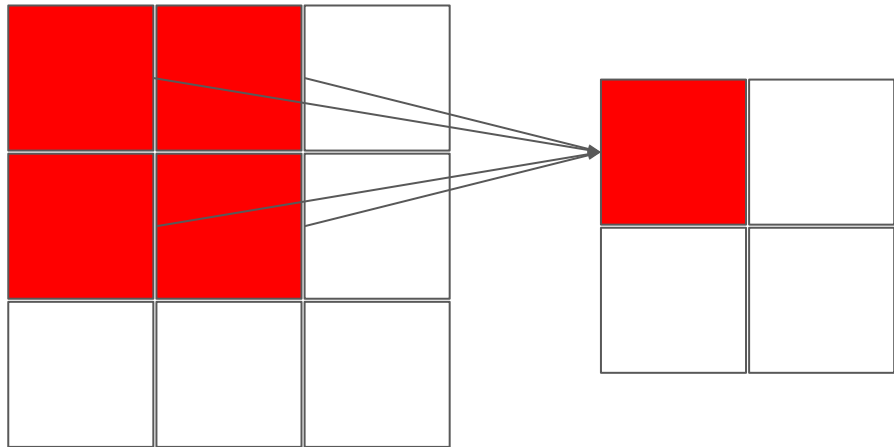
Сколько всего весов в свёрточном слое?

А где же веса-смещения?

А в свёрточной архитектуре их **просто нет!**

Но это ещё не предел экономии))))

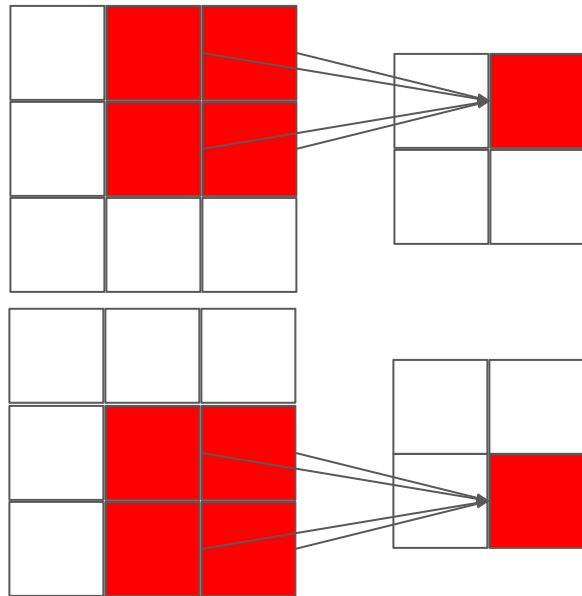
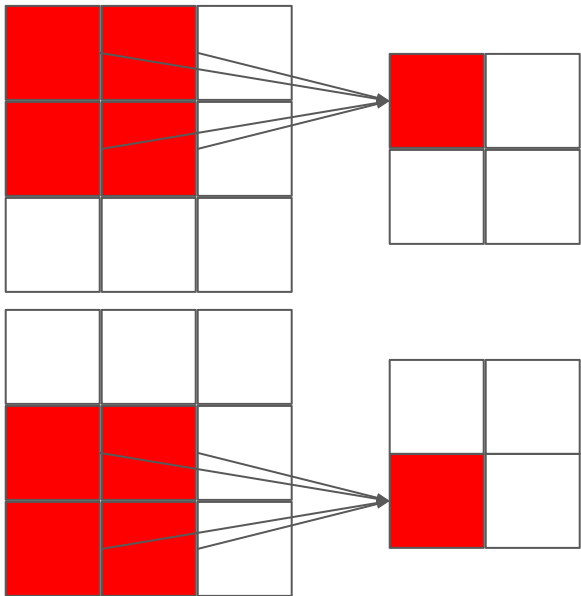
На следующем слайде — просто апофеоз нищевродства.



Сколько всего весов в свёрточном слое?

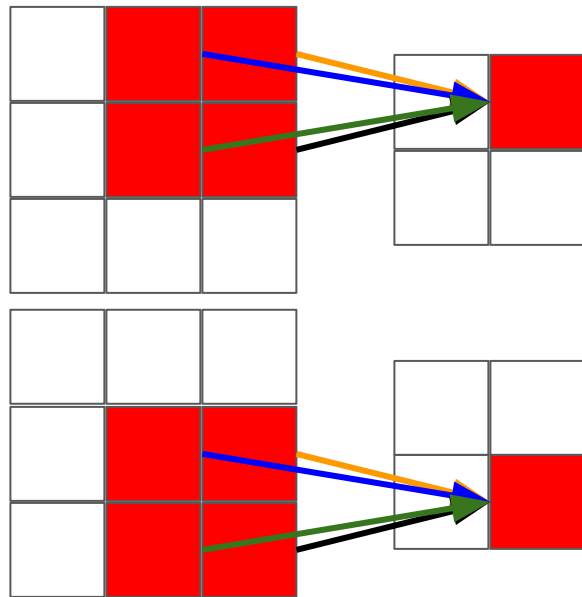
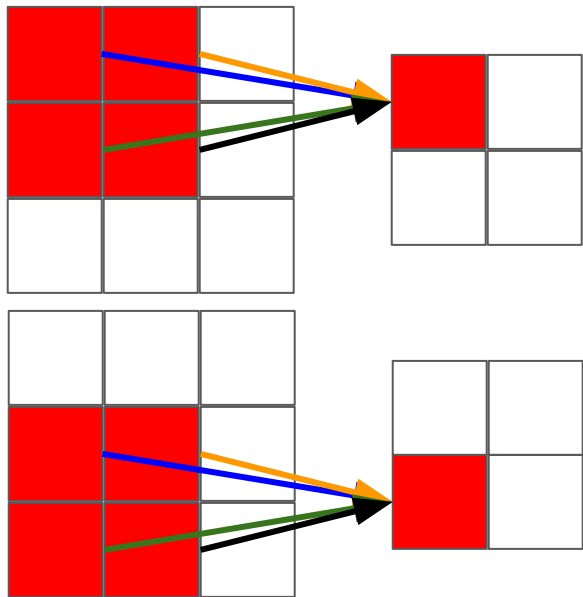
Здесь мы видим 4 пучка связей, ведущих в разные нейроны второго слоя. По идее тут должно быть 16 разных весов для тренировки.

А давайте **ещё сэкономим**: будем считать, что **соответствующие** веса на картинках **равны друг другу**.



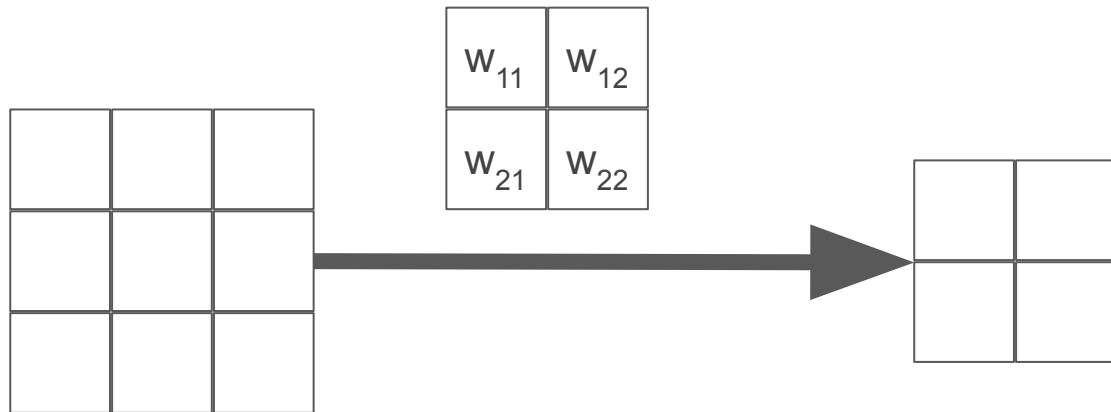
Сколько всего весов в свёрточном слое?

То есть здесь будет только 4 различных веса для тренировки (ниже идентичные веса показаны одним цветом).



Веса пишем в виде матрицы

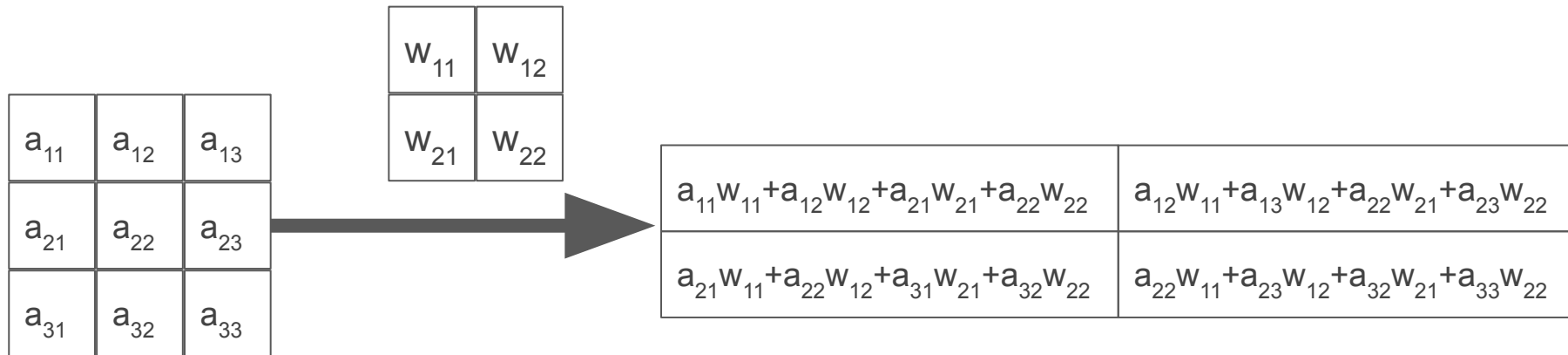
Это позволяет веса записать в виде матрицы (как и слои НС):



Веса пишем в виде матрицы

Вспоминаем, что происходит с весами в нейросети?

Они домножаются на входные значения, и результат суммируется.



Эти преобразования приводят нас к идее **фильтра**.

Матричные фильтры

Матричные фильтры применялись и до эпохи НС в задачах обработки изображений:

- нахождение границ предметов на изображении;
- размытие изображений
- увеличение контрастности, и т.д.

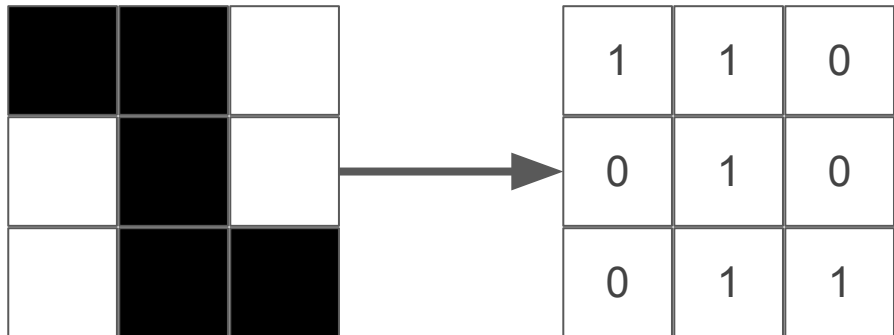
Как работает фильтр:

1. **Фильтр** — это матрица с числами (это **маска фильтра**).
2. **Фильтр прикладывается** к каждой области изображения (изображение тоже представляется в виде матрицы чисел) и результат **операции свёртки** записывается в итоговое изображение.

И операция свёртки полностью аналогична преобразованию данных в НС!!!

Представление изображения и фильтра

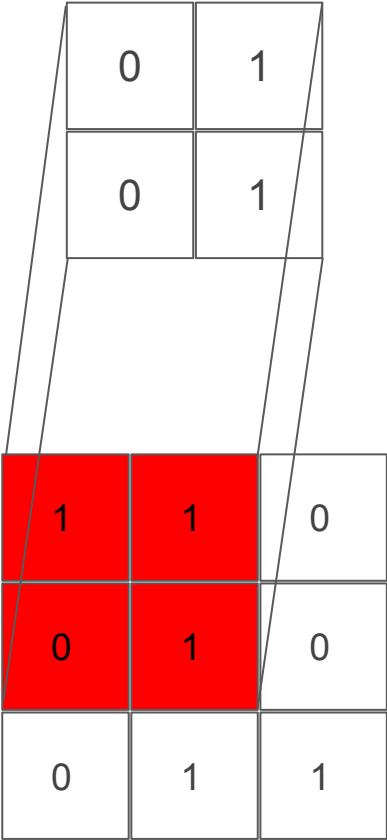
Изображение (ч/б: 0=белый, 1=черный, числа из $[0,1]$ — оттенки серого):



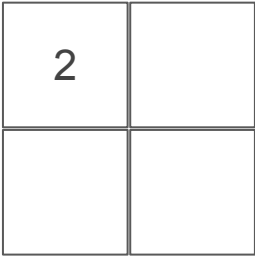
Фильтр:

0	1
0	1

Операция свёртки (перемножить числа и сложить)



Итоговое изображение:



Операция свёртки (перемножить числа и сложить)

0	1
0	1

Итоговое изображение:

2	0

1	1	0
0	1	0
0	1	1

Операция свёртки (перемножить числа и сложить)

0	1
0	1

Итоговое изображение:

2	0
2	

1	1	0
0	1	0
0	1	1

Операция свёртки (перемножить числа и сложить)

0	1
0	1

Итоговое изображение:

2	0
2	1

1	1	0
0	1	0
0	1	1

Фактический результат:

**фильтр выделил фрагменты
изображения, где есть пиксели:**

(в этих клетках появились максимальные значения)



Главное свойство фильтра

Фильтр выделяет те области изображения, которые наиболее сильно похожи на маску фильтра.

Фильтр:

010

111

010

Изображение:

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

Результат свёртки:

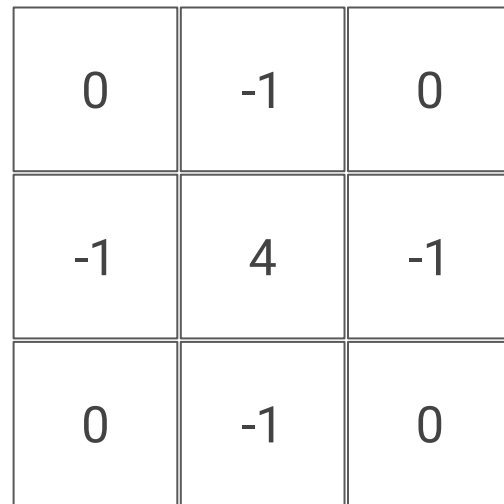
5335

Более сложные фильтры

Этот фильтр умеет выделять **границы изображения**

Изображение:

00000000
01110000
01111000
01111100
00111110
00011111



0	-1	0
-1	4	-1
0	-1	0

0 -1 0
-1 4 -1
0 -1 0

00000000
01110000
01111000
01111100
00111110
00011111

Результат свёртки (имеет размеры 6x6):

2?????
??????
??????
??????
??????
??????

0 -1 0
-1 4 -1
0 -1 0

00000000
01110000
01111000
01111100
00111110
00011111

Результат свёртки (имеет размеры 6x6):

21????
??????
??????
??????
??????
??????

0 -1 0
-1 4 -1
0 -1 0

00000000
01110000
01111000
01111100
00111110
00011111

Результат свёртки (имеет размеры 6x6):

212???
??????
??????
??????
??????
??????

0 -1 0
-1 4 -1
0 -1 0

00000000
01110000
01111000
01111100
00111110
00011111

Результат свёртки (имеет размеры 6x6):

212-2??
??? ???
??? ???
??? ???
??? ???
??? ???

Оставшиеся пиксели в первом ряду будут равны 0.

Переходим к следующей строке изображения

0 -1 0
-1 4 -1
0 -1 0

Результат свёртки (имеет размеры 6x6):

00000000
01110000
01111000
01111100
00111110
00011111

212-200
1?? ???
??? ???
??? ???
??? ???
??? ???

И так далее... Получаем очередной ряд итогового изображения.

В итоге получаем:

0 -1 0
-1 4 -1
0 -1 0

00000000
01110000
01111000
01111100
00111110
00011111

Результат свёртки (имеет размеры 4x6):

2 1 2-2 0 0
1 0 0 2-2 0
2 0 0 0 2-2
-2 2 0 0 0 2

Оказывается, что данный фильтр применяется при **выделении границ изображения**.



Ещё один известный фильтр

А что делает фильтр

1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9

с картинкой:

0001000

0011100

0001000

Можете догадаться, каково предназначение такого фильтра?

Ещё один известный фильтр

Фильтр:

1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9

Изображение:

00100

01110

00100



1/9 4/9 5/9 4/9 1/9

Получилось одномерное изображение.

Вы почувствовали главное свойство этого фильтра?

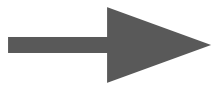
Для какой обработки изображения он применяется?

Спойлер: это фильтр размытия изображения

Возьмём изображение побольше, и всё станет понятнее!

1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9

0001000
0011100
0001000
0011100
0001000
0011100
0001000



1/9 4/9 5/9 4/9 1/9
2/9 5/9 7/9 5/9 2/9
1/9 4/9 5/9 4/9 1/9
2/9 5/9 7/9 5/9 2/9
1/9 4/9 5/9 4/9 1/9
2/9 5/9 7/9 5/9 2/9
1/9 4/9 5/9 4/9 1/9

Цвета мы выделили интенсивность цвета (чтобы было видно размытие).

Какой размер после свёртки?

Из примеров выше мы видим, что свёртка **уменьшает** размер изображения!

Например, если было изображение 6×6 и маска фильтра 3×3 , то результат будет размера...



Какой размер после свёртки?

Из примеров выше мы видим, что свёртка **уменьшает** размер изображения!

Например, если было изображение 6×6 и маска фильтра 3×3 , то результат будет размера 4×4 !

А в общем виде сможете подсчитать?

У вас есть изображение $n \times n$, фильтр $m \times m$.

Тогда ответ будет...



Какой размер после свёртки?

У вас есть изображение $n \times n$, фильтр $m \times m$.

Тогда ответ будет размера

$(n-m+1) \times (n-m+1)$.

«Доказательство».

В последнем примере изображение было 6×6 , фильтр 3×3 , результат 4×4 , где $4 = 6 - 3 + 1$.

Сделаем замену: $6 := n$, $3 := m$.

Тогда $4 := n - m + 1$.

Что и требовалось доказать))))



Параметры свёртки

Тонкая настройка

Фильтр прикладывается к областям изображения. Этот процесс **можно подкрутить** с помощью параметров, отвечающих за:

- **величину шага** (как сильно смещается фильтр по сравнению с предыдущей итерацией). По умолчанию шаг равен 1 (как по вертикали, так и по горизонтали);
- **добавление пустой рамки вокруг изображения** (это нужно для того, чтобы фильтр приложился к крайним точкам изображения).

Stride

Величину шага задают два параметра:

stride_x, stride_y — насколько пикселей сдвигается по горизонтали и вертикали фильтр после очередной итерации.

По умолчанию $\text{stride}_x = \text{stride}_y = 1$, то есть фильтр в ряду изображения каждый раз сдвигается на 1 пиксел, а при окончании ряда фильтр смещается на 1 пиксел вниз.

Stride

Вот так бегают фильтр 3x3 по изображению 6x6,
если `stride_x=1`, `stride_y=2`:

000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000

А потом мы прыгаем...

Stride

Вот так бегают фильтр 3x3 по изображению 6x6, если `stride_x=1`, `stride_y=2`:

000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000
000000	000000	000000	000000

При `stride_y=2` дальше прыгать некуда.



ПАРА-ПАРА-ПАМ

ВСЁ!

Padding — добавление пустой рамки вокруг изображения

Если этого не делать, то:

- а) крайние пиксели редко используются фильтром;
- б) размерность результата будет меньше исходного изображения.

Например, если **padding=2**, то это означает, что вокруг изображения добавляется рамка из белых пикселей шириной 2.

Пример padding=1

Изображение:

110

010

111

После padding=1:

00000

01100

00100

01110

00000

Padding позволяет лучше обработать пиксели на краях изображения и бороться с уменьшением размерности.

Домашнее задание на padding

Задание. Проверим, что padding повышает качество распознавания изображения.

Добавьте вокруг своей фотографии в соцсети рамку из пикселей (лучше черного цвета).

И вы удивитесь, как много людей заявят о наличии в вас прекрасных душевных качеств.

Почему-то без рамки прекрасные душевные качества в человеке обществом не распознаются. С чем это связано?

Композиция фильтров

После применения одного фильтра можно применить второй фильтр

Это позволяет **распознавать более сложные** изображения, имея в своём распоряжении лишь слабые фильтры.

Пример. Научимся распознавать области изображения, в которых присутствует фрагмент

011

110

Конечно, это изображение и можно взять в качестве фильтра. Но мы попробуем обойтись более простыми фильтрами.

После применения одного фильтра можно применить второй фильтр

Оказывается, что фрагмент

011
110

можно детектировать с помощью **двух простых фильтров**.

Сначала применяем фильтр:

01
10

а потом ещё один:

11

...продолжение примера

Итак, берем фильтр:

01

10

а потом ещё один:

11

Тогда изображение

0111

0111

1110

0101

после первого фильтра будет:

122

222

120

А потом после второго фильтра:

34

44

32

Максимальные значения соответствуют областям в исходном изображении, которые содержали изображение

011

110

Действительно, мы получили

34

44

32

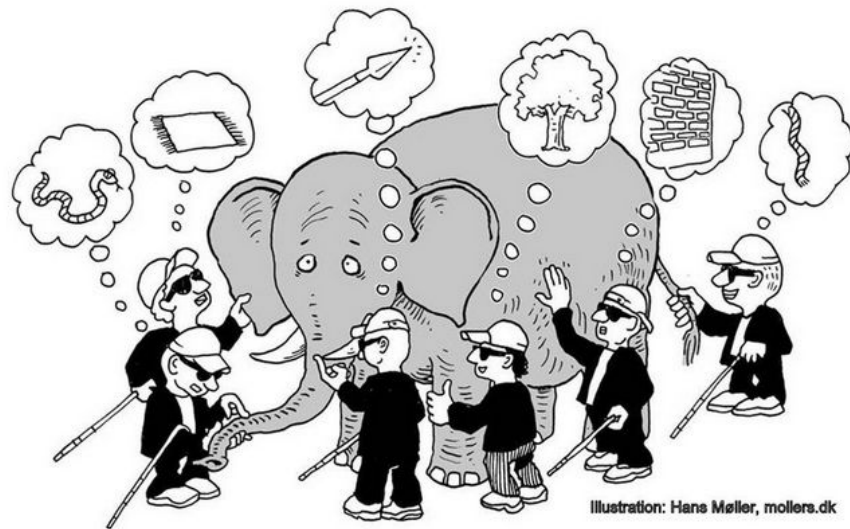
Вот эти фрагменты:

0111	0111	0111
0111	0111	0111
1110	1110	1110
0101	0101	0101

Полная аналогия со слепыми мудрецами

Первый фильтр: Несколько слепых мудрецов ощупали слона (мы знаем, что их информация противоречива).

Второй фильтр: один сверх-мудрец опросил слепых мудрецов и понял, что это был слон.



Важный вывод про композицию фильтров

Если к изображению последовательно применяется несколько фильтров (то есть результат одного фильтра подаётся на вход следующему), то первые фильтры распознают самые базовые закономерности, более сложные закономерности распознаются лишь последними фильтрами.

Это настолько важная мысль, что она полностью набрана жирным шрифтом.



Выводы

- Мы познакомились с новой архитектурой нейросетей — свёрточными нейронными сетями. Они гораздо лучше (чем классические полносвязные сети) справляются с задачей распознавания изображения.
- Мы изучили операцию свёртки и работу фильтра.
- Мы рассмотрели результат композиции нескольких фильтров.