

Глубокое обучение

Дмитрий Никулин

28 апреля 2021 г.

Неделя 6: Свёрточные нейросети

Agenda

- Как видит компьютер
- Свёртка и пулинг
- Играемся со свёртками
- Если успеем, собираем свою свёрточную сетку

Затравка (2006)



Please click on all the images that show cats:

The grid contains the following images:

- Row 1: A black and white dog, a brown dog, a white and brown dog, an orange cat.
- Row 2: A white and brown dog, a tan dog, a person petting a black cat, a brown and white cat.
- Row 3: A white dog, an orange cat, a brown dog, a black and white dog.
- Row 4: A white dog, an orange cat, a brown dog, a black and white dog.

Затравка (2006)

- Капча портит вид сайтов, довольно бесполезная, в Microsoft придумывают в 2006 году новый вид капчи. Надо отличать котов от собак.
- В то время разделение собак от кошек было очень сложной задачей, лучшая точность была 0.6.
- Если надо классифицировать 12 картинок одновременно, вероятность угадать — 0.6^{12} .
- Пул картинок пополнялся фотографиями из приютов (через сайт petfinder.com).

В 2014 проект закрыли



Dogs vs. Cats

Create an algorithm to distinguish dogs from cats
215 teams · 6 years ago

Overview Data Notebooks Discussion Leaderboard Rules

Public Leaderboard Private Leaderboard

The private leaderboard is calculated with approximately 70% of the test data. Refresh

This competition has completed. This leaderboard reflects the final standings.

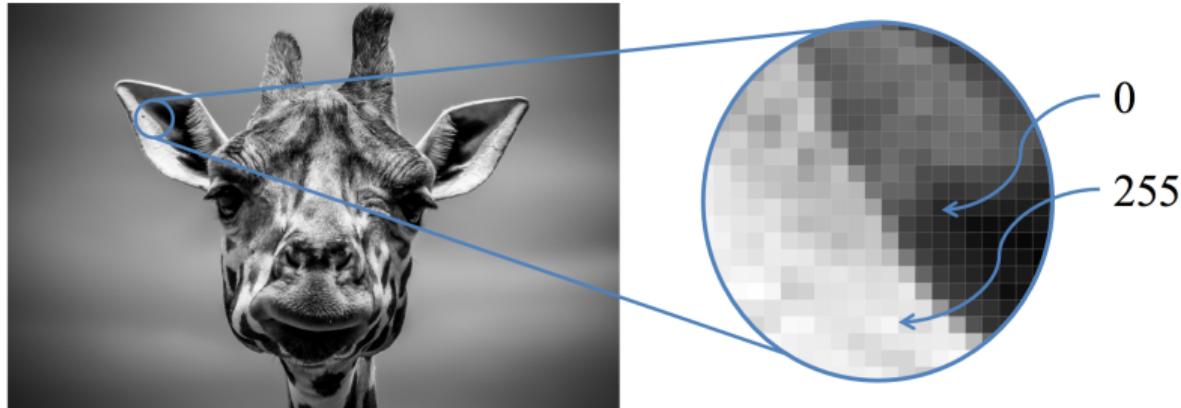
Gold Silver Bronze

#	△pub	Team Name	Notebook	Team Members	Score ⓘ	Entries	Last
1	—	Pierre Sermanet			0.98914	5	6y
2	▲ 4	orchid			0.98308	17	6y
3	—	Owen			0.98171	15	6y
4	—	Paul Covington			0.98171	3	6y

Как видит компьютер

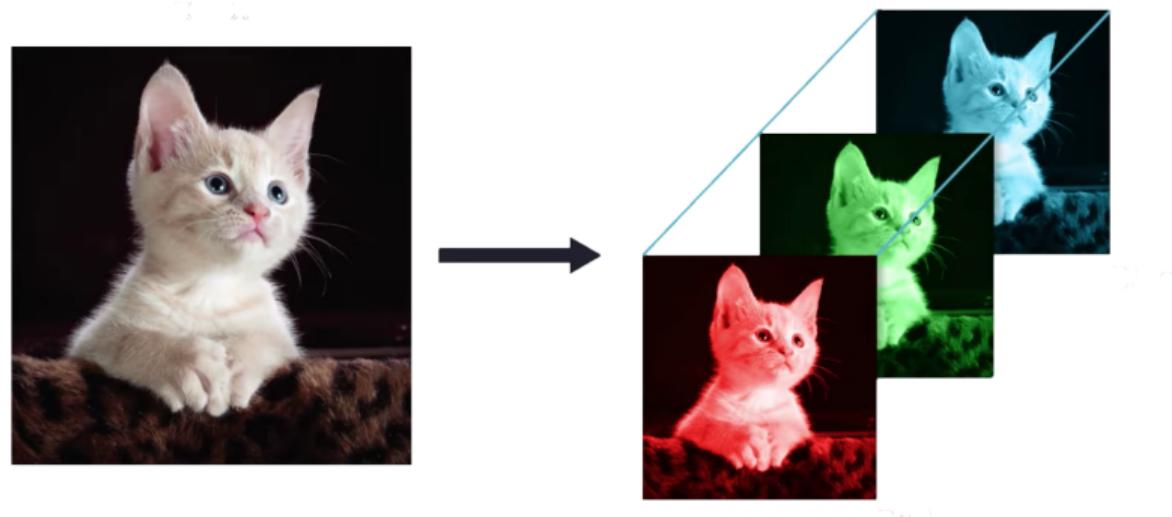
Картинка — тензор

- Каждая картинка — это матрица из пикселей
- Каждый пиксель обладает яркостью по шкале от 0 до 255

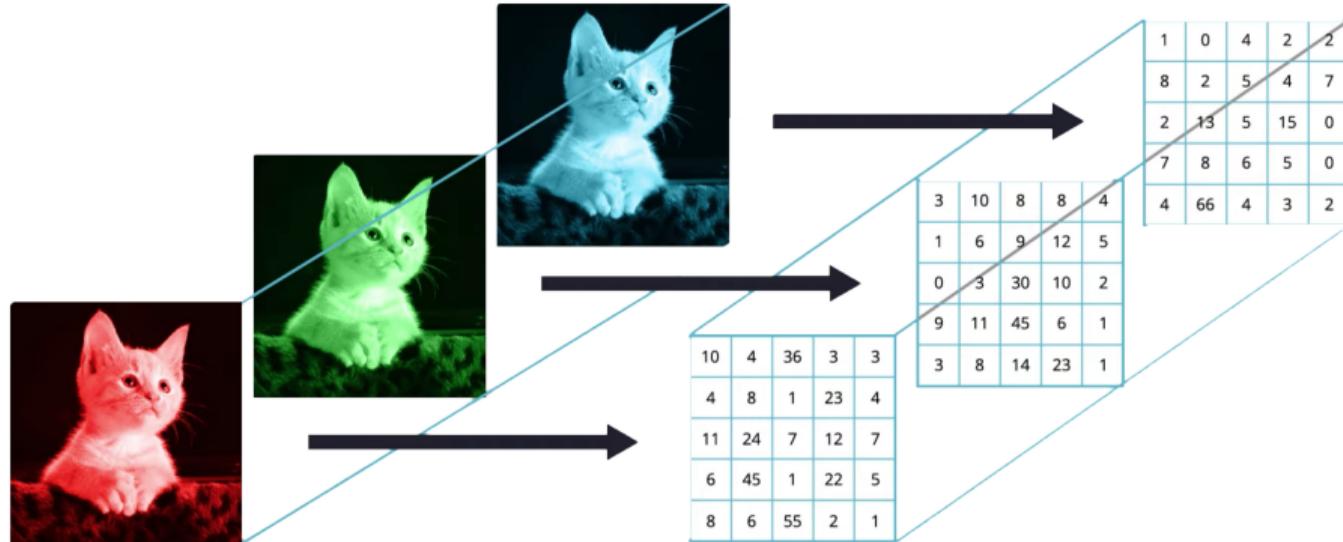


Картина — тензор

- Цветное изображение имеет три канала пикселей: красный, зелёный и синий (red, green, blue / RGB), размерность изображения $\text{height} \times \text{width} \times 3$



Картина — тензор



(для иллюстрации вообразим, что эта картинка имеет размер 5×5 пикселей)

Картина — тензор

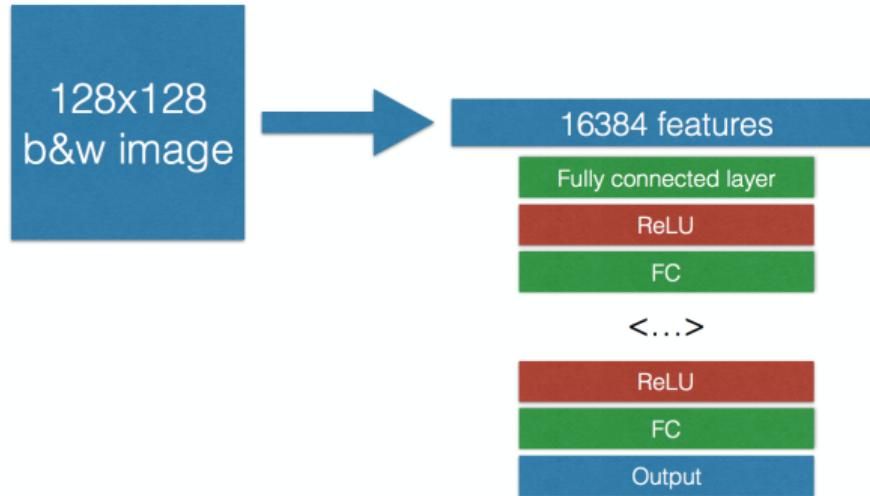
$5 \times 5 \times 3$



3D Array

5			5			5		
R			G			B		
10	2	36	3	3	3	5	1	1
4	8	1	23	4	4	2	0	0
11	24	7	12	7	7	1	0	0
6	45	1	22	5	5	1	1	1
8	6	55	2	1	1	1	1	1

Обычная сетка

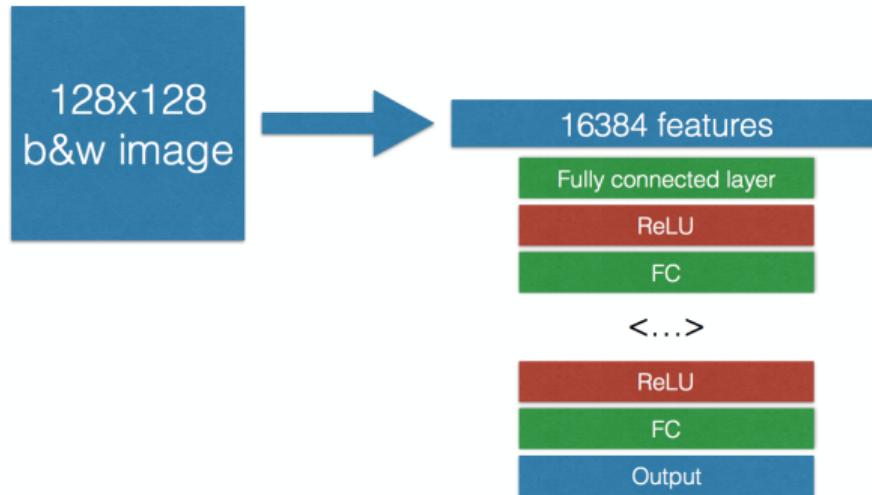


- Развернём картинку в вектор \Rightarrow **очень много весов**
- Сетка может легко переобучиться
- Лучшая борьба с переобучением — уменьшение числа параметров





Обычная сетка



- Изображение в разных местах картинки даёт разные веса

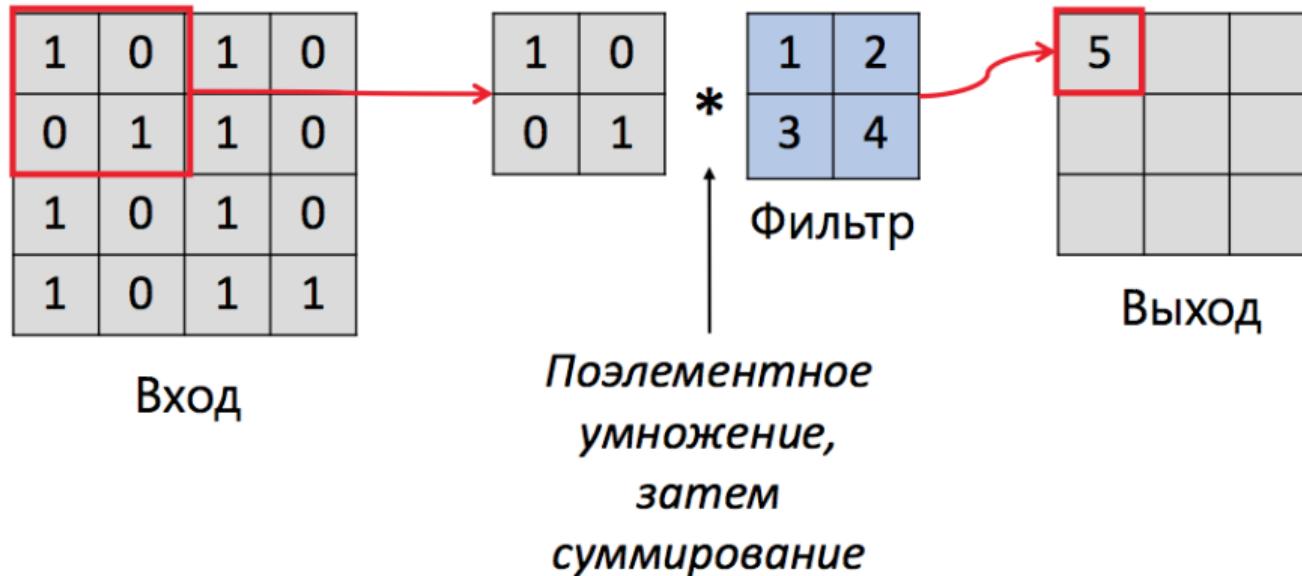
Хотелки

- Хотим меньше параметров, избежать переобучения
- Хотим, чтобы информация не терялась
- Хотим, чтобы модель была нечувствительна к сдвигам картинки в новые места

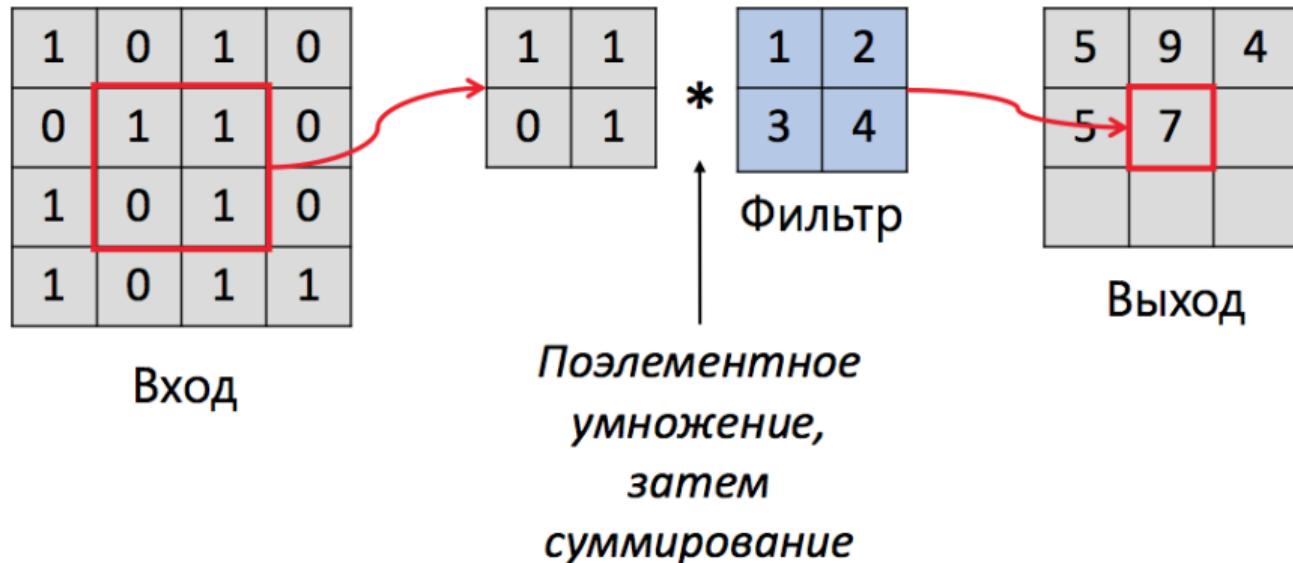
⇒ свёртка

Свёртка

Свёртка



Свёртка





```
[[62, 36, 9], [[62, 36, 9], [[62, 36, 9],  
[62, 36, 9], [61, 35, 8], [60, 34, 7],  
[61, 35, 8], [59, 33, 6], [57, 31, 4],  
..., ..., ...,  
[58, 43, 20], [53, 41, 17], [50, 38, 14],  
[57, 45, 21], [53, 41, 17], [49, 37, 13],  
[57, 45, 21]] [52, 40, 16]] [48, 36, 12]]
```

Красный

Зеленый

Синий

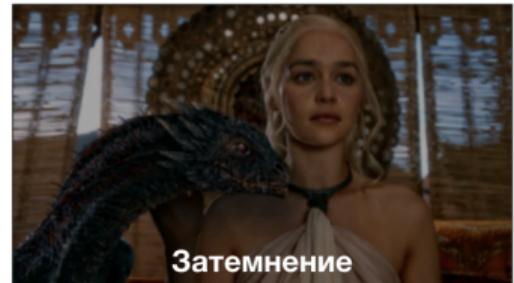
$$\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.7 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

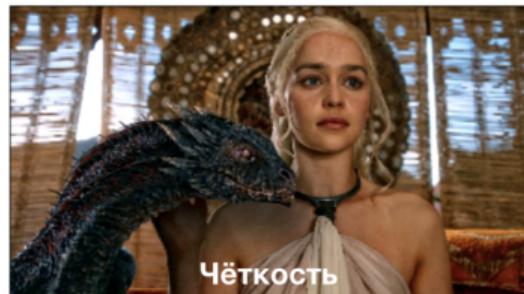


Размытие



Затемнение

$$\begin{pmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{pmatrix}$$

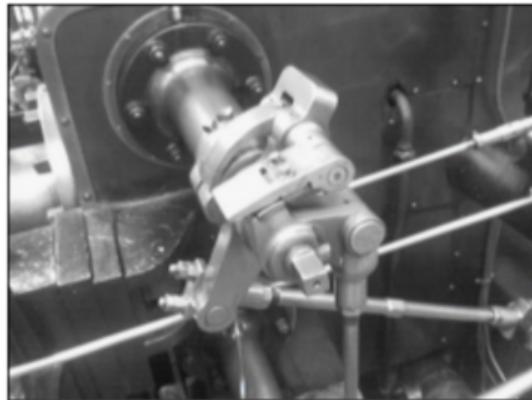


Чёткость

Выделение границ

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$



<https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html>

Свёртка

- Разные ядра помогают накладывать на картинку различные эффекты
- Какие-то ядра помогают искать границы
- **Идея:** возможно, с помощью некоторых ядер можно искать что-то кроме границ...

Классификатор слэшей

Input

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

Kernel

1	0
0	1

=

Output

0	0	0
0	1	0
0	0	2

The diagram shows a 4x4 input matrix and a 2x2 kernel matrix being multiplied to produce a 3x3 output matrix.

Input:

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

Kernel:

1	0
0	1

Output:

0	0	0
0	0	1
0	1	0

The result is obtained by applying the kernel to the input, shifting it right and down by one unit at each step. The highlighted regions in red and green indicate the receptive field of the bottom-right output unit.

Классификатор слэшей

Input

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

*

Kernel

1	0
0	1

=

Output

0	0	0
0	1	0
0	0	2

Max = 2

Simple

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 \\ \hline \end{array} \quad * \quad
 \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} =
 \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

↓
Max = 1

Input Kernel Output

Свёртка инвариантна к расположению

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 2 \\ \hline \end{array}$$

Input Kernel Output

$$\begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 2 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Input Kernel Output

Свёртка инвариантна к расположению

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

Input

$$\begin{matrix} & * & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} & = & \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{matrix} \end{matrix}$$

Kernel Output

Max = 2



Didn't change

Max = 2

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

Input

$$\begin{matrix} & * & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} & = & \begin{matrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} \end{matrix}$$

Kernel Output

Свёртка в виде формулы

Чёрно-белый случай:

$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} K[i, j] \cdot \text{Image}[h + i, w + j]$$

Цветной случай:

$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} \sum_{c=1}^C K[i, j, c] \cdot \text{Image}[h + i, w + j, c]$$

Свёртка в виде формулы

Чёрно-белый случай:

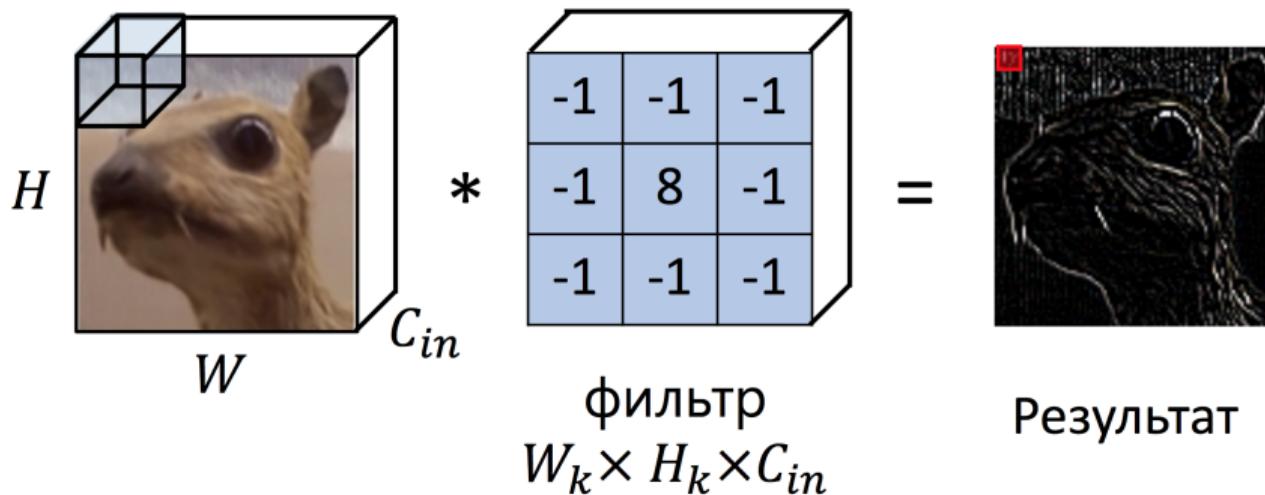
$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} K[i, j] \cdot \text{Image}[h + i, w + j]$$

Цветной случай:

$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} \sum_{c=1}^C K[i, j, c] \cdot \text{Image}[h + i, w + j, c]$$

В цветном случае ядро, как и картинка, становится трёхмерным!

Свёртка в виде формулы



Для тех, кто проходил матанализ

- У нас свёрткой называется такая операция:

$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} \sum_{c=1}^C K[i, j, c] \cdot \text{Image}[h + i, w + j, c]$$

- Свёртка из матанализа записывалась бы так:

$$out[h, w] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} \sum_{c=1}^C K[-i, -j, C - c + 1] \cdot \text{Image}[h + i, w + j, c]$$

Нейросетевая свёртка в матанализе называется (кросс-)корреляцией.

Свёртка

- Операция свёртки выявляет наличие на изображение паттерна, который задаётся конкретным фильтром (ядром)
- Чем сильнее на участке изображения представлен паттерн, тем больше будет значение свёртки
- Результат свёртки изображения с фильтром — новое изображение
- Нас будет интересовать много различных паттернов \Rightarrow будем использовать несколько свёрток сразу

Хорошее введение в арифметику свёрток: <https://arxiv.org/pdf/1603.07285.pdf>

Несколько свёрток сразу

- Несколько каналов может быть не только на входе, но и на выходе:

$$out[h, w, c_{out}] = \sum_{i=-H_k}^{H_k} \sum_{j=-W_k}^{W_k} \sum_{c_{in}=1}^{C_{in}} K[i, j, c_{in}, c_{out}] \cdot \text{Image}[h + i, w + j, c_{in}]$$

- Естественно, порядок размерностей не важен. Например, в PyTorch в слое `nn.Conv2d` ядро свёртки хранится как тензор размера $C_{out} \times C_{in} \times H_k \times W_k$.

Смотрим анимации

https://github.com/vdumoulin/conv_arithmetic

Padding

Padding

- Если применять свёртку по формуле, итоговое изображение будет меньше исходного
- Из-за этого мы можем терять информацию на краях изображения

Zero padding

0_2	0_0	0_1	0	0	0	0
0_1	2_0	2_0	3	3	3	0
0_0	0_1	1_1	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8

Zero padding

- Добавляем по границам нули так, чтобы посчитанная после этого свёртка давала изображение такого же размера, как исходное
- Есть риск, что модель научится понимать, где на изображении края — можем потерять инвариантность. Но на практике этот риск несущественен
- Этот режим будет использоваться, если слою nn.Conv2d передать параметр padding, то есть в PyTorch это в некотором смысле "режим по умолчанию"

Reflection padding

3	5	1
3	6	1
4	7	9

No padding

1	6	3	6	1	6	3
1	5	3	5	1	5	3
1	6	3	6	1	6	3
9	7	4	7	9	7	4
1	6	3	6	1	6	3

(1, 2) reflection padding

<https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/>

Replication padding

3	5	1
3	6	1
4	7	9

No padding

5	3	3	5	1	1	5
5	3	3	5	1	1	5
6	3	3	6	1	1	6
7	4	4	7	9	9	7
7	4	4	7	9	9	7

(1, 2) replication padding

<https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/>

Padding

- Паддинг позволяет контролировать размер выходных изображений
- Паддинг позволяет учитывать даже объекты на краях
- Разные типы паддингов допускают разные способы переобучения под края

Как оно работает
внутри нейронок

Свёрточный слой

В свёрточных слоях (например, nn.Conv2d в PyTorch) используется всё то, о чём мы говорили:

- Собственно операция свёртки
- Несколько каналов (как на входе, так и на выходе)
- Padding

Помимо этого, к выходу часто ещё добавляется bias — обучаемый вектор длиной C_{out} . Таким образом, к каждому выходному каналу прибавляется одно число.

Свёрточный слой

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

Input 3x3
image with
zero **padding**
(grey area)

Shared bias:

b

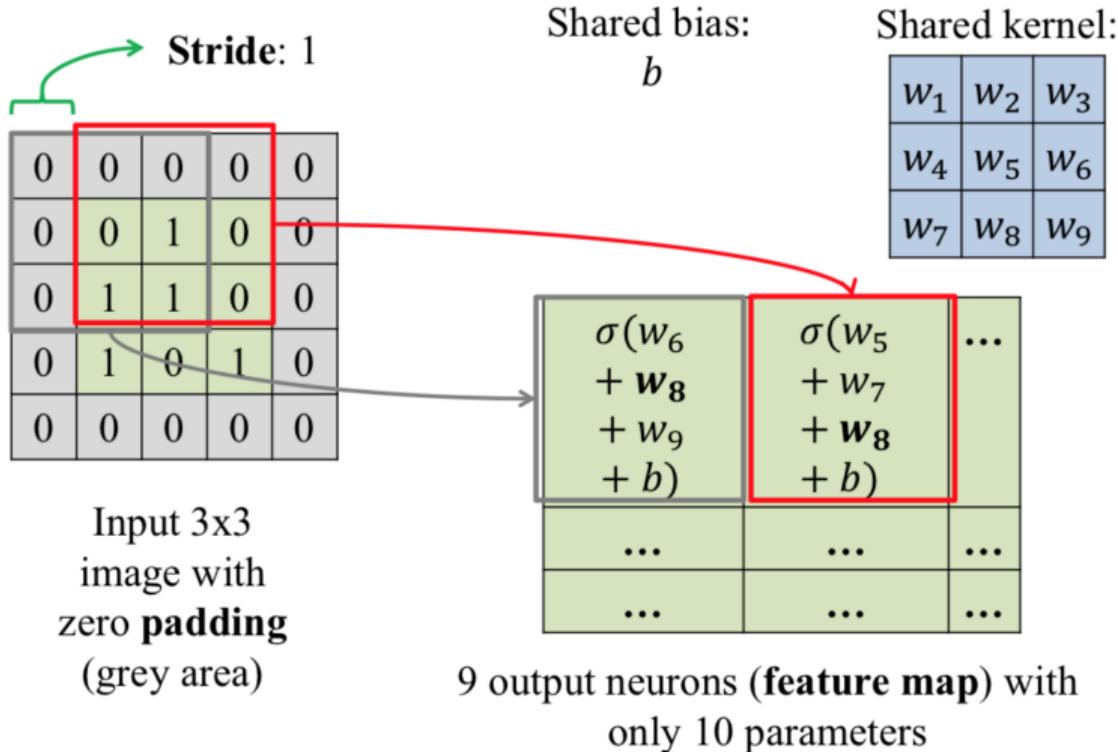
Shared kernel:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

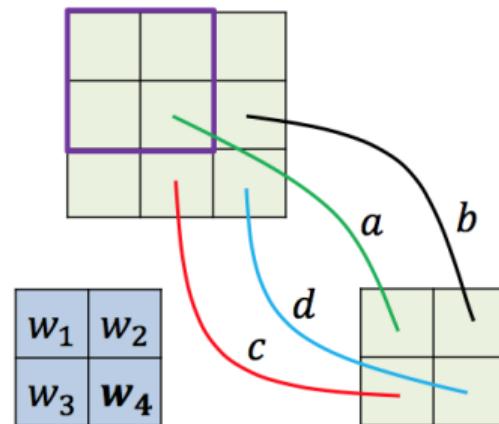
$\sigma(w_6$ $+ w_8$ $+ w_9$ $+ b)$
...
...

9 output neurons (**feature map**) with
only 10 parameters

Свёрточный слой



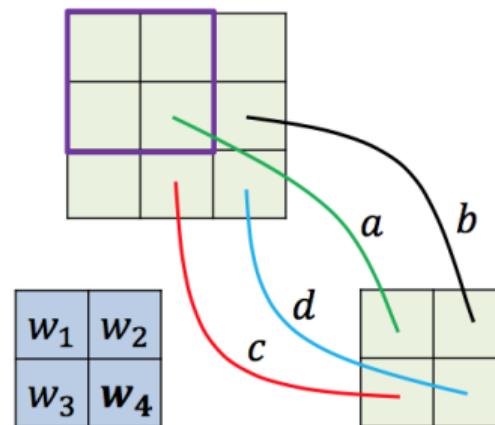
Backpropagation для свёрточного слоя



$$a = w_1 x_{11} + w_2 x_{12} + w_3 x_{21} + w_4 x_{22}$$

$$b = w_1 x_{12} + w_2 x_{13} + w_3 x_{22} + w_4 x_{23}$$

Backpropagation для свёрточного слоя



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \cdot x_{11} + \frac{\partial L}{\partial b} \cdot x_{12} + \frac{\partial L}{\partial c} \cdot x_{21} + \frac{\partial L}{\partial d} \cdot x_{22}$$

Свёртка — это просто умножение на матрицу

$$\begin{array}{|c|c|} \hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 6 & 5 & 3 \\ \hline 1 & 4 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 22 & 21 \\ \hline 22 & 20 \\ \hline \end{array}$$

2x2 Kernel 3x3 Input

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

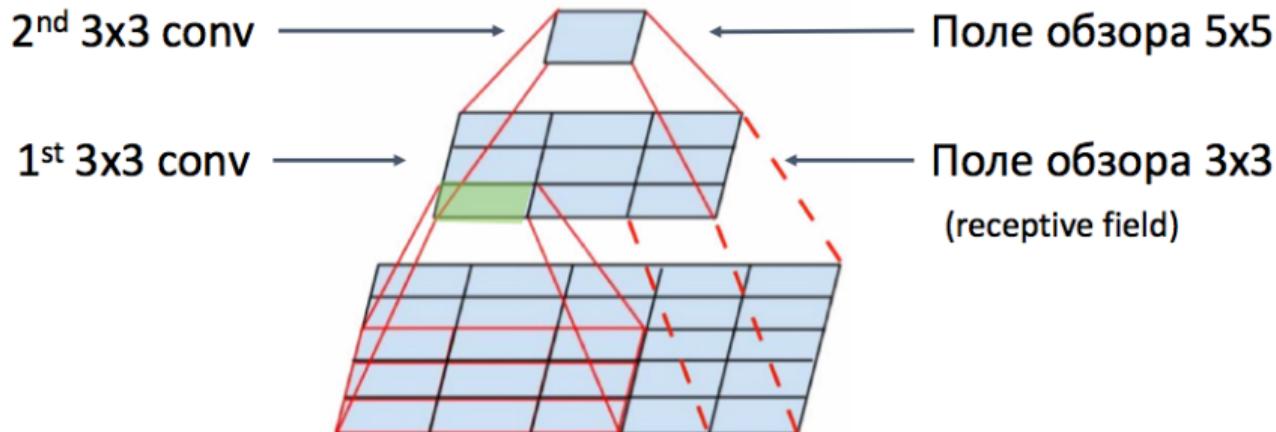
$$x = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline 6 \\ \hline 5 \\ \hline 3 \\ \hline 1 \\ \hline 4 \\ \hline 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|} \hline 22 \\ \hline 21 \\ \hline 22 \\ \hline 20 \\ \hline \end{array}$$

Свёрточный слой

- Слой действует одинаково для каждого участка картинки, в отличие от полносвязного
- Нужно оценивать меньшее количество параметров
- Слой учитывает взаимное расположение пикселей
- Можно учить тем же самым backpropagation
- Свёрточный слой - это полносвязный слой с ограничениями

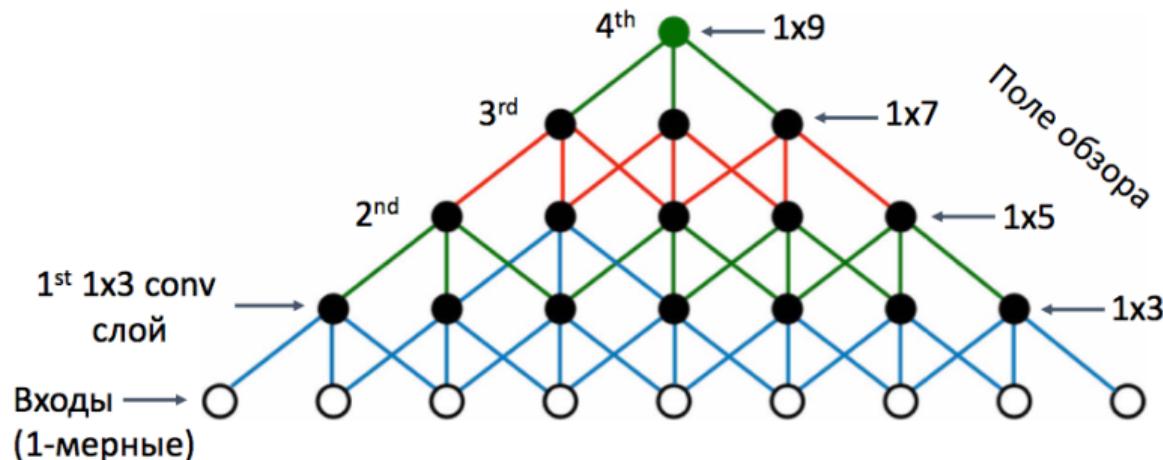
Одного свёрточного слоя недостаточно

- Нейроны первого слоя смотрят на поле 3×3
- Если интересующий нас объект больше, нам нужна вторая свёртка



Одного свёрточного слоя недостаточно

- N слоёв со свёртками 3×3
- На N -ом слое поле обзора $(2N + 1) \times (2N + 1)$
- Если наш объект размера 300, надо 150 слоёв... \Rightarrow нужно растить поле обзора быстрее

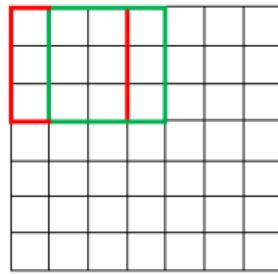


Нужно растить поле обзора быстрее!

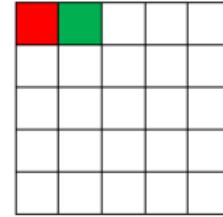
- Пиксели локально скоррелированы — соседние пиксели, как правило, не сильно отличаются друг от друга
- Если будем делать свёртку с каким-то шагом, сэкономим мощности компьютера и не потеряем в информации

Сдвиг (Stride)

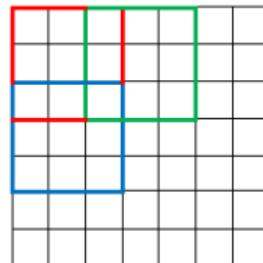
7 x 7 Input Volume



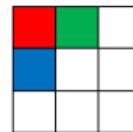
5 x 5 Output Volume



7 x 7 Input Volume



3 x 3 Output Volume



Пуллинг слой (Pooling)

- Будем считать внутри какого-то окна максимум или среднее и сворачивать размерность, пользуясь локальной коррелированностью

2	4	5	7	3	-2
-2	0	0	4	9	9
1	0	-1	2	1	1
1	1	6	3	7	2
3	4	0	-2	3	0
3	0	5	1	0	0

Feature Map

2	4	5	7	3	-2
-2	0	0	4	9	9
1	0	-1	2	1	1
1	1	6	3	7	2
3	4	0	-2	3	0
3	0	5	1	0	0

Pool size=2

Max
Pooling

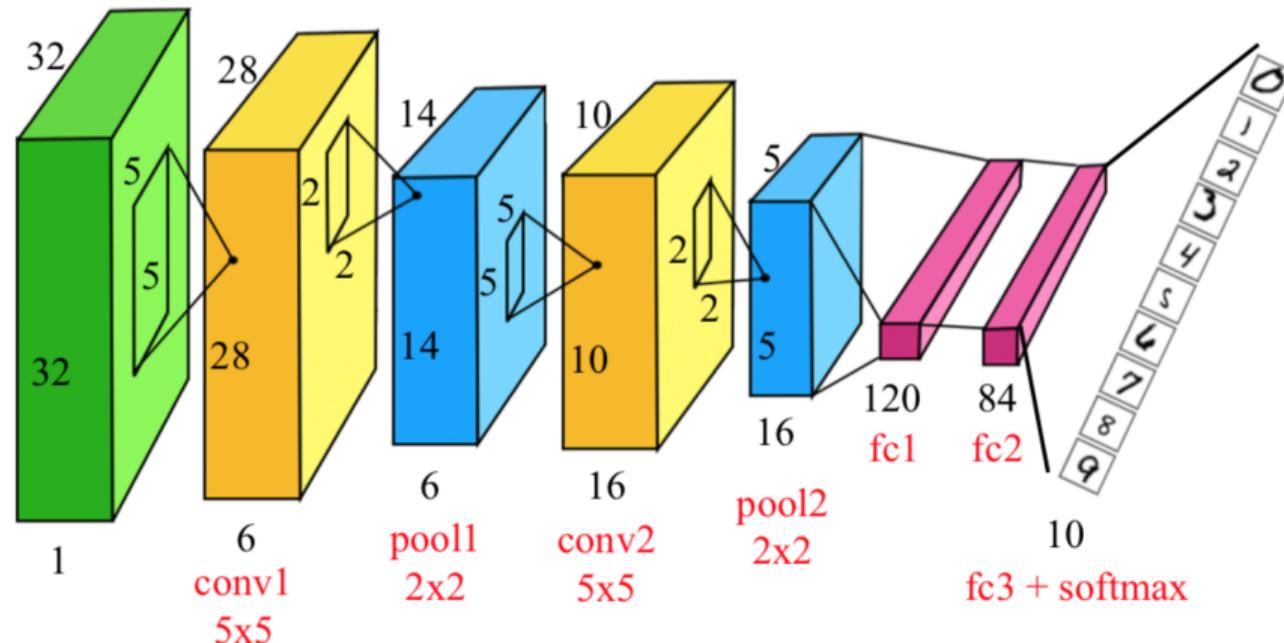
4	7	9
1	6	7
4	5	3

Average
Pooling

1	4	4,8
0,8	2,5	2,8
2,5	1	0,8

Простейшая CNN

Нейросеть LeNet-5 (1998) для распознавания рукописных цифр



<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>