

# Глубокое обучение и вообще

Ульянкин Филипп

6 марта 2023 г.

**Посиделка 5:** Свёрточные нейронки

# Image recognition



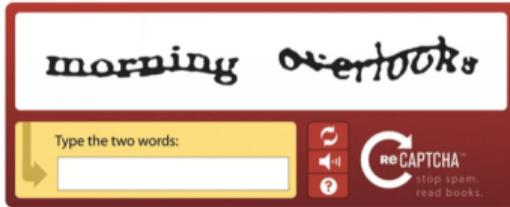
# Image recognition



# Image recognition



# Asirra captcha (2006)



Please click on all the images that show cats:

The image displays a grid of 16 small photographs used in the Asirra CAPTCHA. The top row contains four images of dogs, each with a blue "adopt me" link below it. The bottom row contains four images of cats, also with blue "adopt me" links. The images are arranged in a 4x4 grid. A blue border surrounds the entire grid, and a small circular icon with a plus sign is located in the top right corner of the border.

## Asirra captcha (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.
- У нас есть 12 картинок, робот нагнёт нас с вероятностью  $0.6^{12}$ .
- Пул картинок пополнялся фотографиями из приютов.

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

⟳ Refresh

#	△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

# Agenda

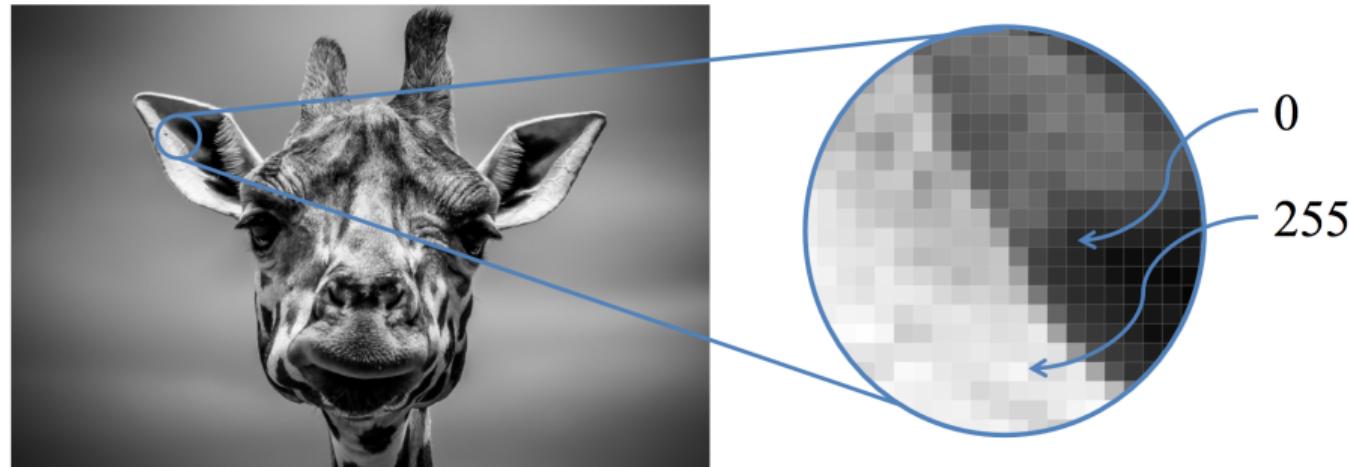
- Как видит компьютер
- Свёртка
- Свёрточный слой
- Типичная свёрточная архитектура
- Что видит свёрточная нейросеть

# Как видит компьютер



# Картина – тензор

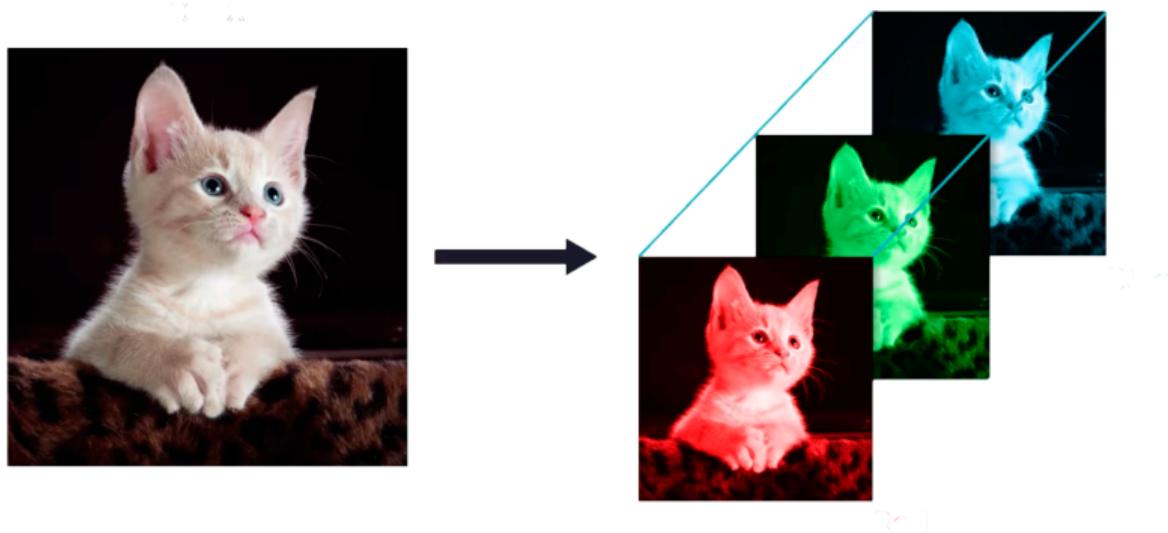
- Каждая картинка – это матрица из пикселей
- Каждый пиксель обладает яркостью по шкале от 0 до 255



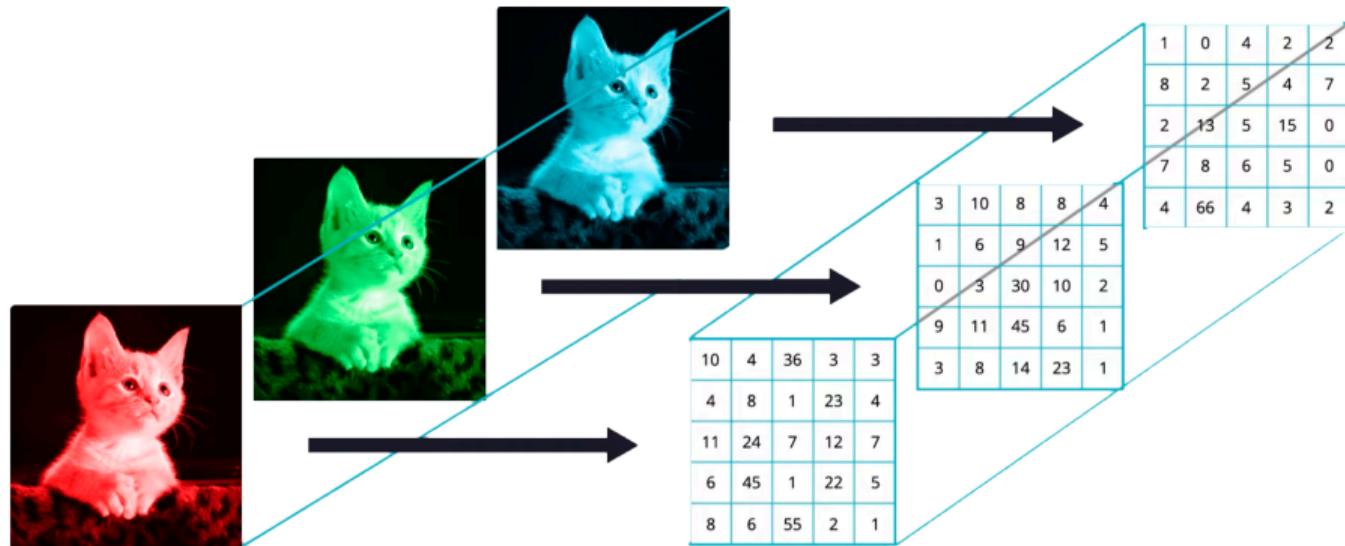
Многие слайды я нагло украл у Андрея Зимовнова:  
<https://github.com/ZEMUSHKA/mml-minor>

# Картина – тензор

- Цветное изображение имеет три канала яркости: красный, зелёный и синий (rgb)



# Картина – тензор



# Картина – тензор

$5 \times 5 \times 3$



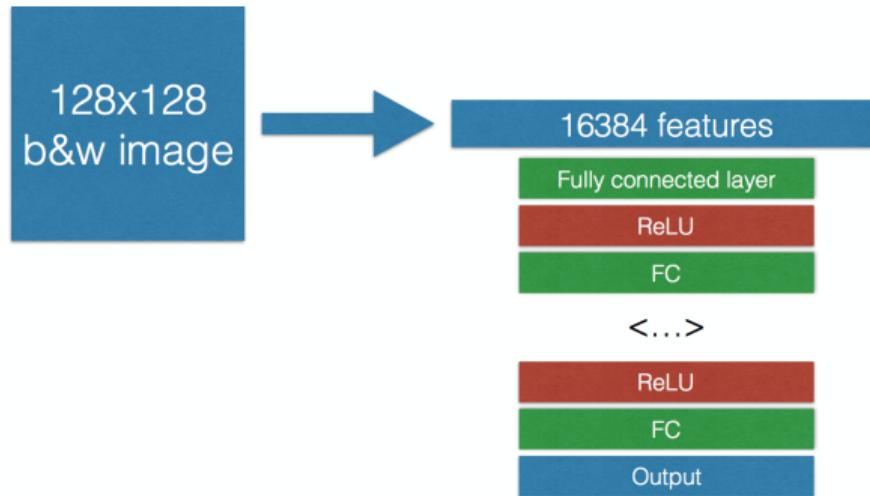
3D Array

7	0	4	2	2	2	3
3	10	36	3	3	9	1
2	4	1	23	4	5	0
20	11	24	7	12	7	0
20	6	45	1	22	5	1
20	8	6	55	2	1	1

5      R      G      B

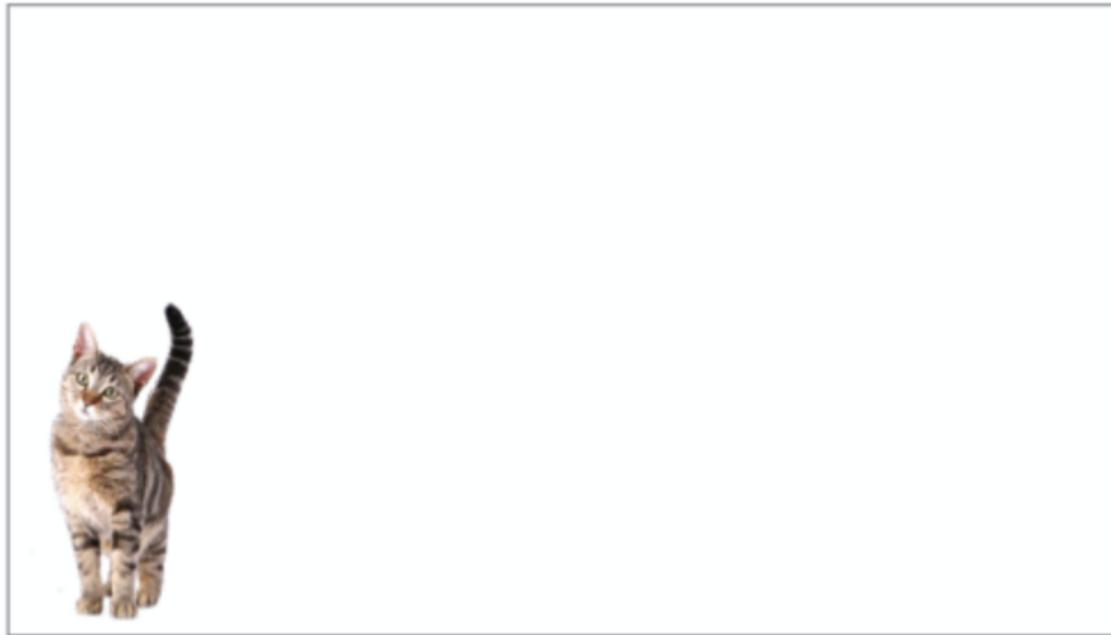
5

# Обычная сеть



- Развернём картинку в вектор  $\Rightarrow$  **очень много весов**
- Если изображение немного сдвинуть, то нейрон уже не будет на него реагировать





# Полносвязные слои для изображений

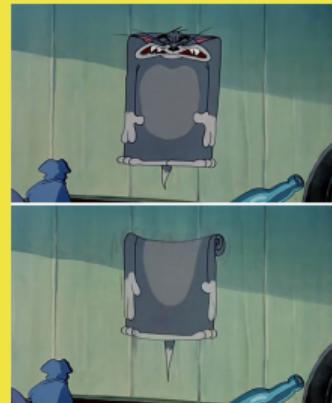
- Очень много параметров
- Легко могут переобучиться
- Не учитывают специфику изображений (сдвиги, небольшие изменения формы и т.д.)
- Изображение в разных местах картинки даёт разные веса
- Теряется информация о взаимном расположении пикселей

# Мы хотим, чтобы...

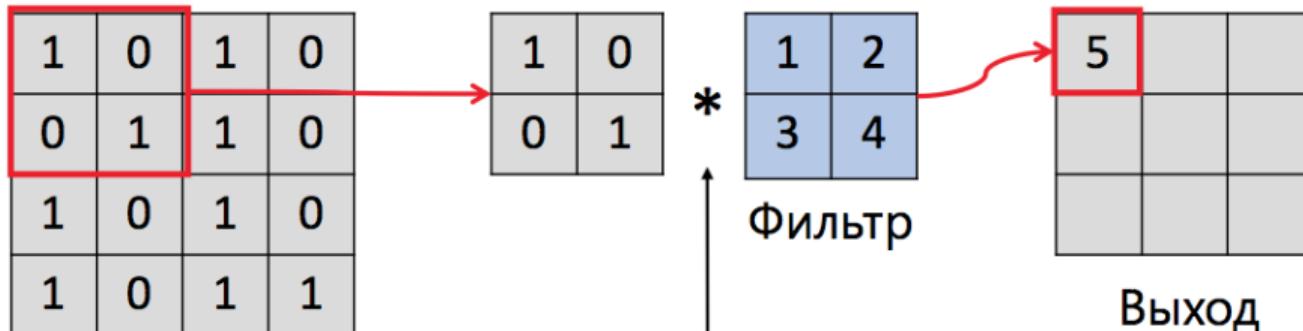
- Хотим меньше параметров, избежать переобучения
- Хотим, чтобы информация не терялась
- Хотим, чтобы модель была нечувствительна к сдвигам картинки в новые места

⇒ свёртка

# Свёртка



# Свёртка

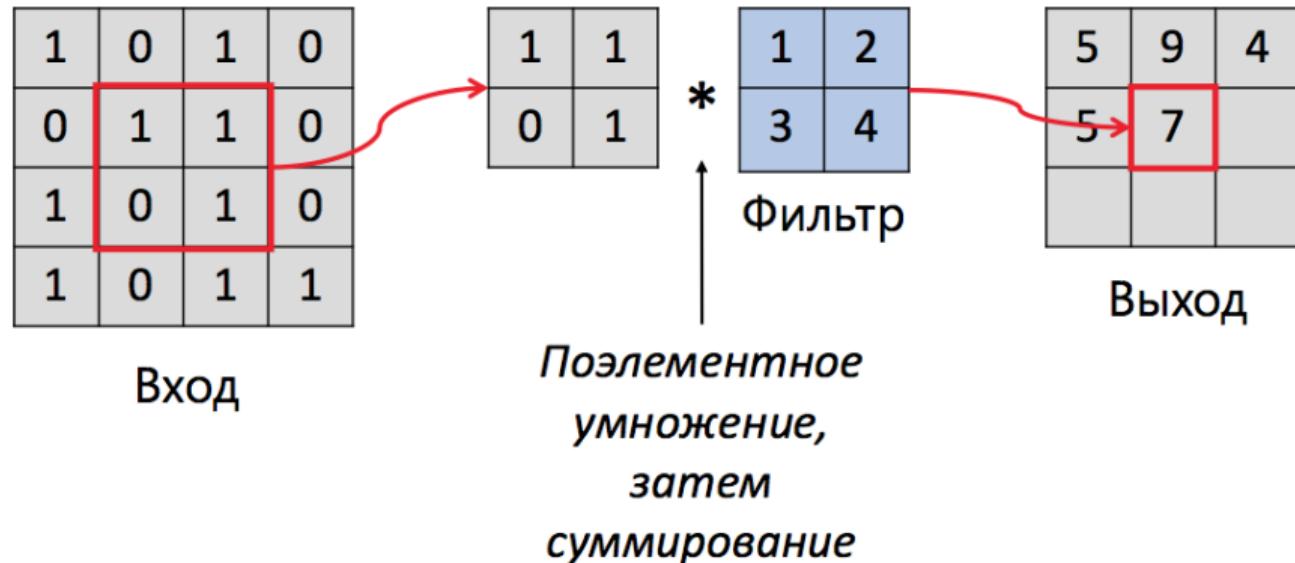


Вход

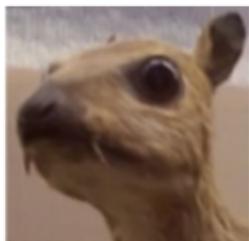
*Поэлементное  
умножение,  
затем  
суммирование*

Выход

# Свёртка



## Фильтр



Входная  
картинка

$$* \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$



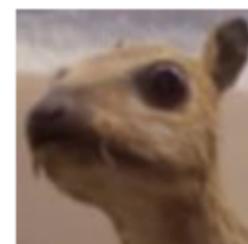
Поиск краев

$$* \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$



Повышение  
резкости

$$* \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

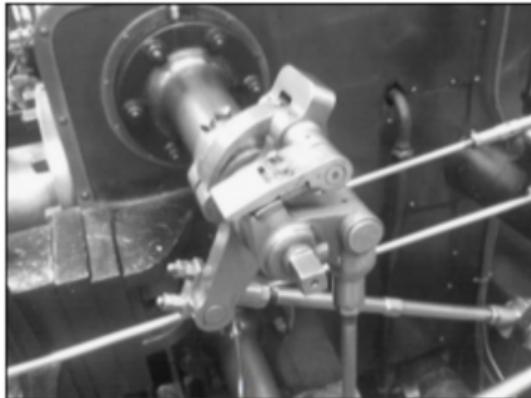


Размытие

# Выделение границ

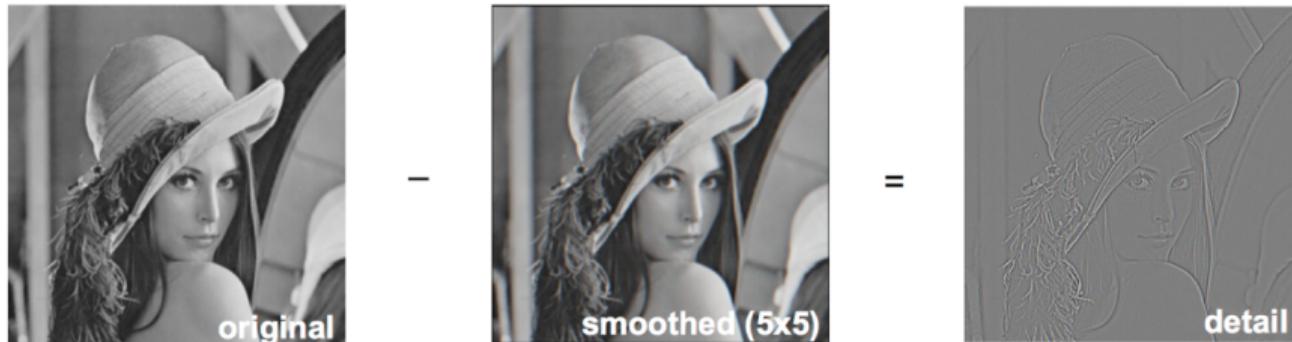
$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

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



# Можно комбинировать ядра

Step 1: Original - Smoothed = "Details"



•0	•0	•0
•0	•1	•0
•0	•0	•0

-

$\frac{1}{9}$	•1	•1	•1
•1	•1	•1	•1
•1	•1	•1	•1

=

# Можно комбинировать ядра

Step 2: Original + "Details" = Sharpened



$$\begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & \bullet & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & \bullet & 2 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & \bullet & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

# Свёртка

- Операция свёртки выявляет наличие на изображении паттерна, который задаётся фильтром
- Результат операции свёртки – новое изображение
- Чем сильнее на участке изображения представлен паттерн, тем больше будет значение свёртки
- **Идея:** дать нейросети возможность самостоятельно придумать ядро для поиска нужных ей паттернов

## Классификатор слэшей

Input                      Kernel                      Output

# Классификатор слэшей

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

Input

\*

1	0
0	1

Kernel

=

0	0	0
0	1	0
0	0	2

Output

Max = 2



Simple  
classifier

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

Input

\*

1	0
0	1

Kernel

=

0	0	0
0	0	1
0	1	0

Output

Max = 1

# 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

\*

1	0
0	1

Kernel

=

0	0	0
0	1	0
0	0	2

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

\*

1	0
0	1

Kernel

=

2	0	0
0	1	0
0	0	0

Output

# Размерность изображения падает

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1	0
0 <sub>2</sub>	0 <sub>2</sub>	1 <sub>0</sub>	3	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3 <sub>0</sub>	2 <sub>1</sub>	1 <sub>2</sub>	0
0	0 <sub>2</sub>	1 <sub>2</sub>	3 <sub>0</sub>	1
3	1 <sub>0</sub>	2 <sub>1</sub>	2 <sub>2</sub>	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2 <sub>0</sub>	1 <sub>1</sub>	0 <sub>2</sub>
0	0	1 <sub>2</sub>	3 <sub>2</sub>	1 <sub>0</sub>
3	1	2 <sub>0</sub>	2 <sub>1</sub>	3 <sub>2</sub>
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>2</sub>	3	1
3 <sub>2</sub>	1 <sub>2</sub>	2 <sub>0</sub>	2	3
2 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0 <sub>0</sub>	1 <sub>1</sub>	3 <sub>2</sub>	1
3	1 <sub>2</sub>	2 <sub>2</sub>	2 <sub>0</sub>	3
2	0 <sub>0</sub>	0 <sub>1</sub>	2 <sub>2</sub>	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1 <sub>0</sub>	3 <sub>1</sub>	1 <sub>2</sub>
3	1	2 <sub>2</sub>	2 <sub>2</sub>	3 <sub>0</sub>
2	0	0 <sub>0</sub>	2 <sub>1</sub>	2 <sub>2</sub>
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2	3
2 <sub>2</sub>	0 <sub>2</sub>	0 <sub>0</sub>	2	2
2 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3 <sub>1</sub>	1 <sub>0</sub>	2 <sub>1</sub>	2 <sub>2</sub>	3
2 <sub>0</sub>	0 <sub>2</sub>	0 <sub>2</sub>	2 <sub>0</sub>	2
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2 <sub>0</sub>	2 <sub>1</sub>	3 <sub>2</sub>
2	0	0 <sub>2</sub>	2 <sub>2</sub>	2 <sub>0</sub>
2	0	0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>2</sub>

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

## Дополнение изображения (zero padding)

0 <sub>2</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0	0	0	0
0 <sub>1</sub>	2 <sub>0</sub>	2 <sub>0</sub>	3	3	3	0
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>1</sub>	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

## Дополнение (Padding)

- Если применять свёртку по формуле, итоговое изображение будет меньше исходного
- Из-за этого мы можем терять информацию на краях изображения
- Нужно заполнить края рамочкой: zero padding, replication padding, reflection padding
- Есть риск, что модель научится понимать, где на изображении края — можем потерять инвариантность
- Разные типы паддингов допускают разные способы переобучения под края

# Свёртка в виде формулы

Чёрно-белый случай:

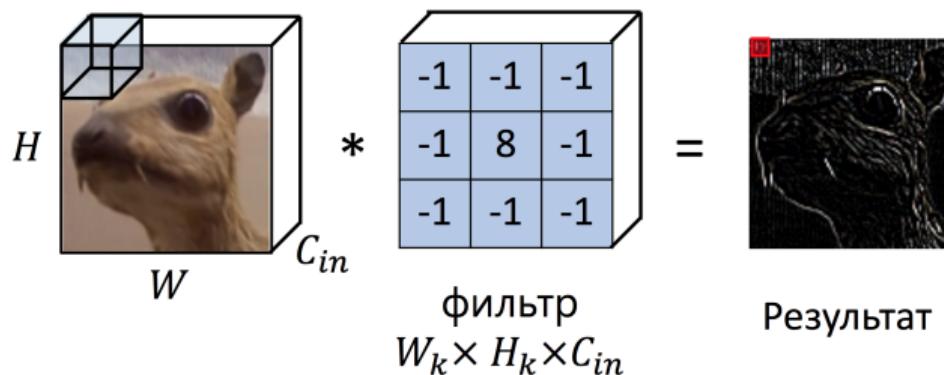
$$out[x, y] = \sum_{i=-d}^d \sum_{j=-d}^d K[i, j] \cdot Image[x + i, y + j]$$

Цветной случай:

$$out[x, y] = \sum_{i=-d}^d \sum_{j=-d}^d \sum_{c=1}^C K[i, j, c] \cdot Image[x + i, y + j, c]$$

# Свёртка для цветной картинки

- Цветная картинка — это тензор  $W \times H \times C_{in}$
- $W$  — высота,  $H$  — ширина,  $C_{in}$  — число каналов



## Резюме по свёрткам

- 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>

# Свёрточный слой



# Свёрточный слой

**Идея:** дать нейросети возможность самостоятельно придумать ядро для поиска нужных ей паттернов

# Свёрточный слой

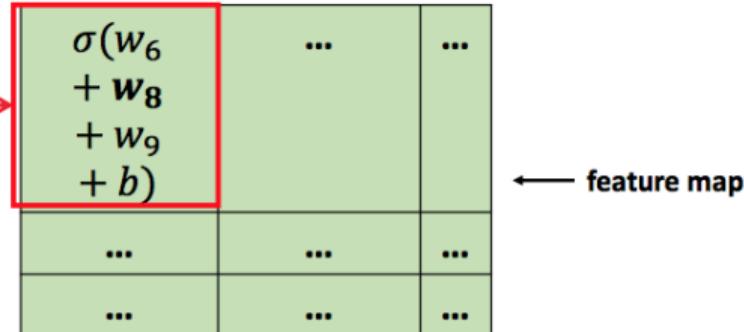
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
0	1	0	1	0
0	0	0	0	0

Входное 3x3  
изображение с  
**нулевой добавкой**  
(padding, серая  
рамка)

Сдвиг:  
 $b$

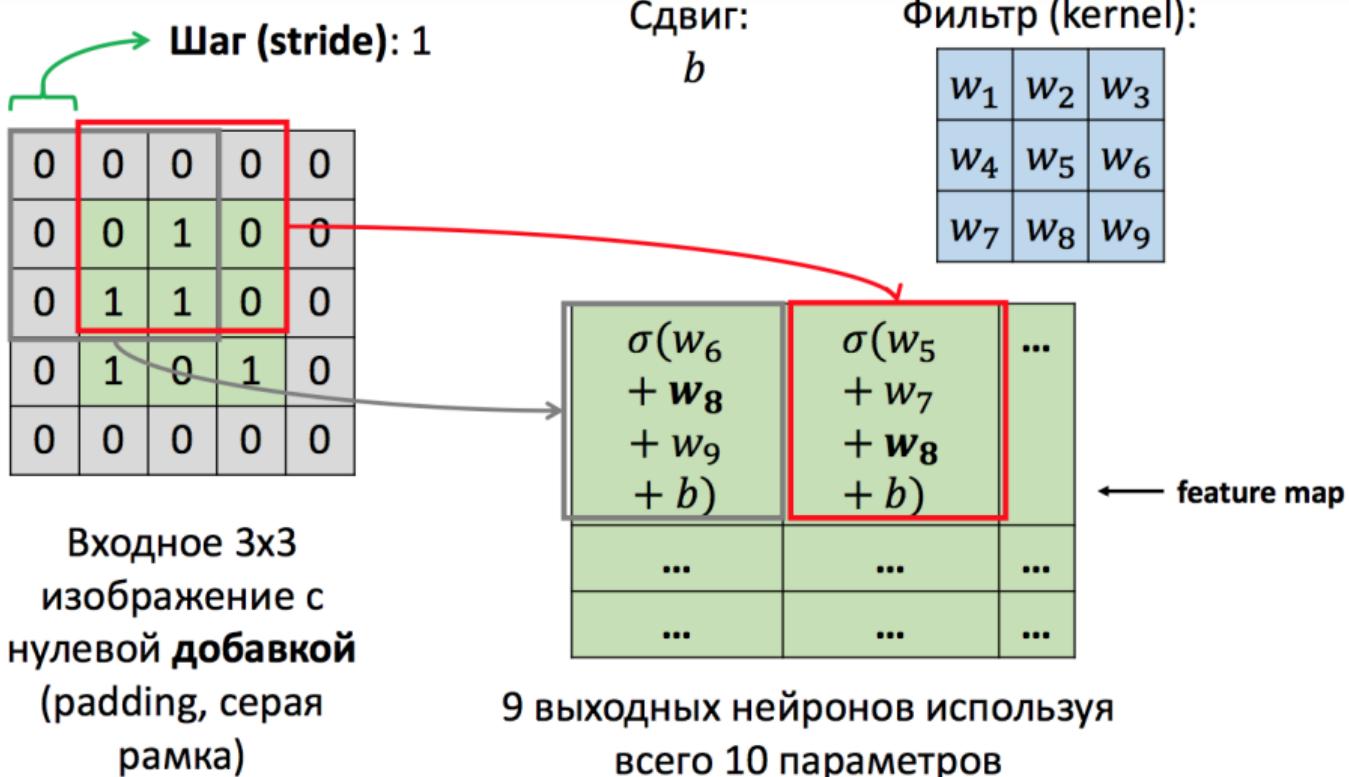
Фильтр (kernel):

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

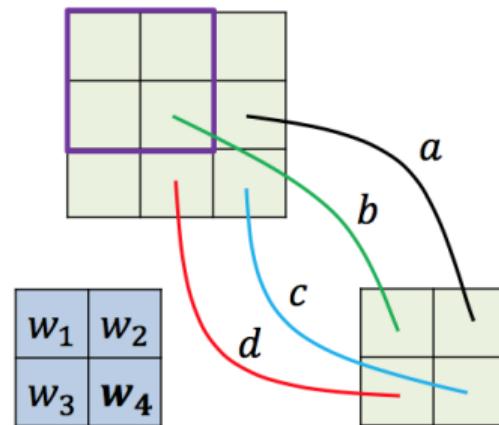


9 выходных нейронов используя  
всего 10 параметров

# Свёрточный слой



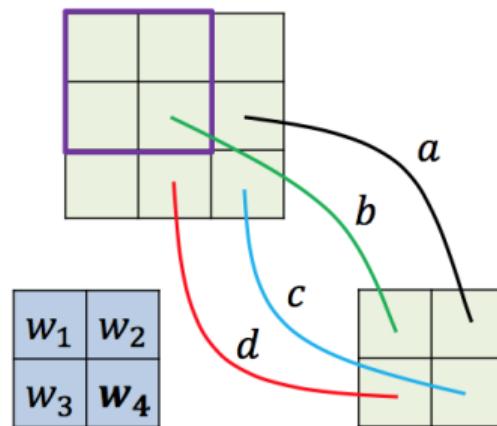
# Как считать градиенты



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

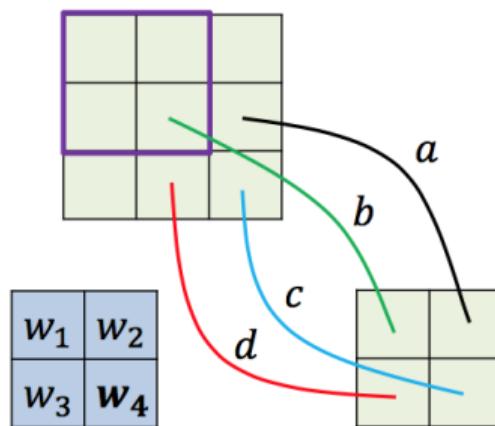
$$a = w_1 x_{12} + w_2 x_{13} + w_3 x_{22} + w_4 x_{23}$$

# Как считать градиенты



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

# Как считать градиенты



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

Суммируем градиенты по всем использованием веса  $w_4$

# Свёрточный слой

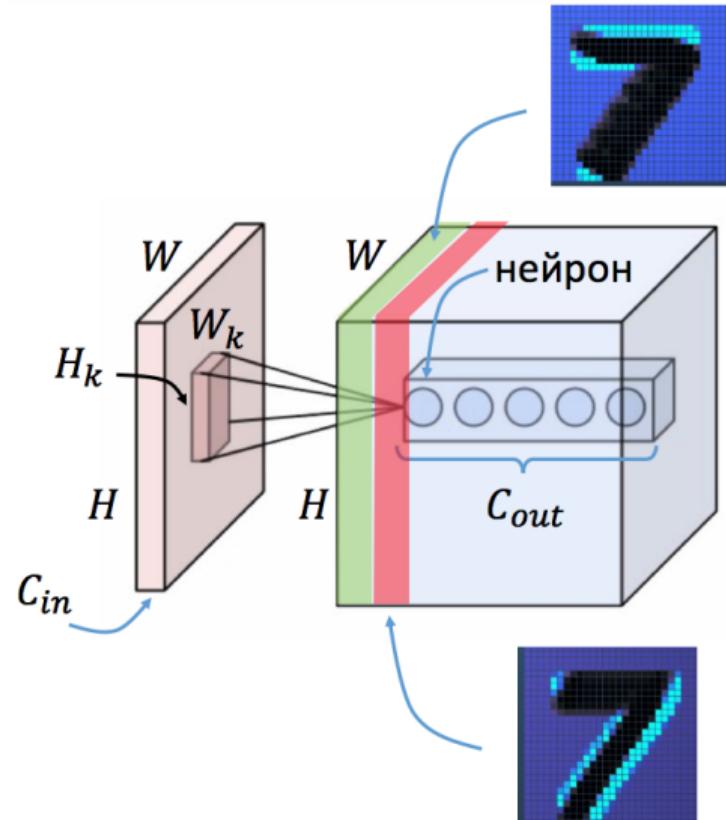
- Слой действует одинаково для каждого участка картинки, в отличие от полносвязного
- Нужно оценивать меньшее количество параметров
- Слой учитывает взаимное расположение пикселей
- Можно учить тем же самым алгоритмом обратного распространения ошибки
- Свёрточный слой — это полносвязный слой с ограничениями

## В свёрточном слое меньше параметров

- Картинка  $300 \times 300$
- $300 \times 300$  нейронов
- Свертка  $5 \times 5$  – 26 параметров
- В полносвязном слое –  $8.1 \cdot 10^9$  параметров

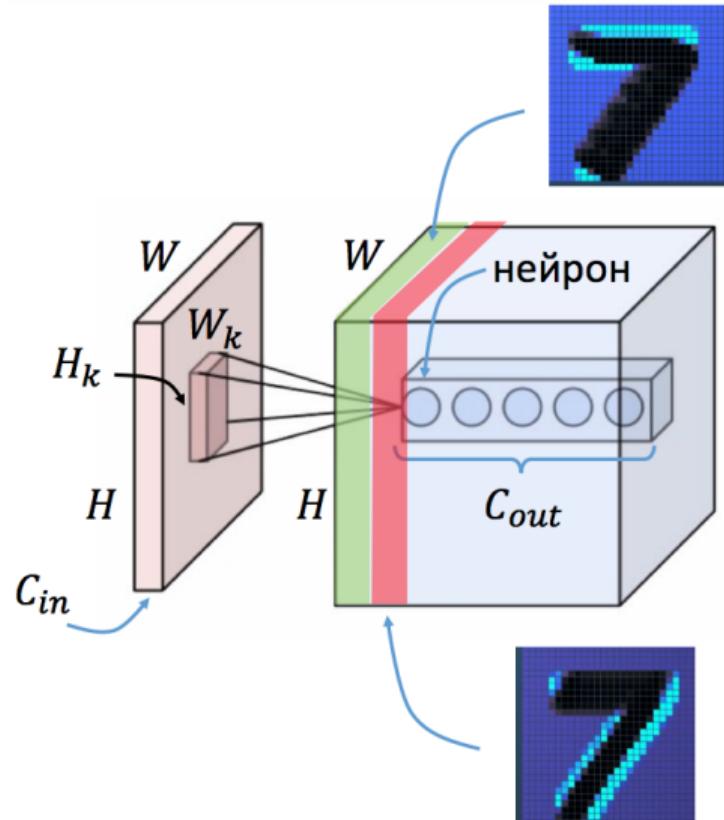
# Одного фильтра мало!

- На входе  $W \times H \times C_{in}$
- На выходе  $W \times H \times C_{out}$
- В одном пикселе появляется глубина с разными характеристиками пикселя изображения
- Сколько параметров надо оценить?

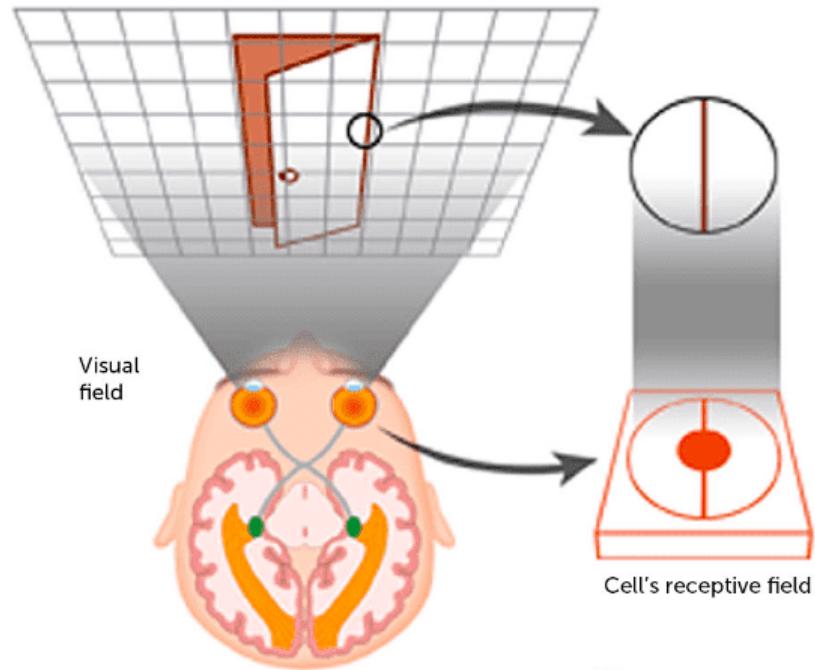


# Одного фильтра мало!

- На входе  $W \times H \times C_{in}$
- На выходе  $W \times H \times C_{out}$
- В одном пикселе появляется глубина с разными характеристиками пикселя изображения
- $(W_k \cdot H_k \cdot C_{in} + 1) \cdot C_{out}$

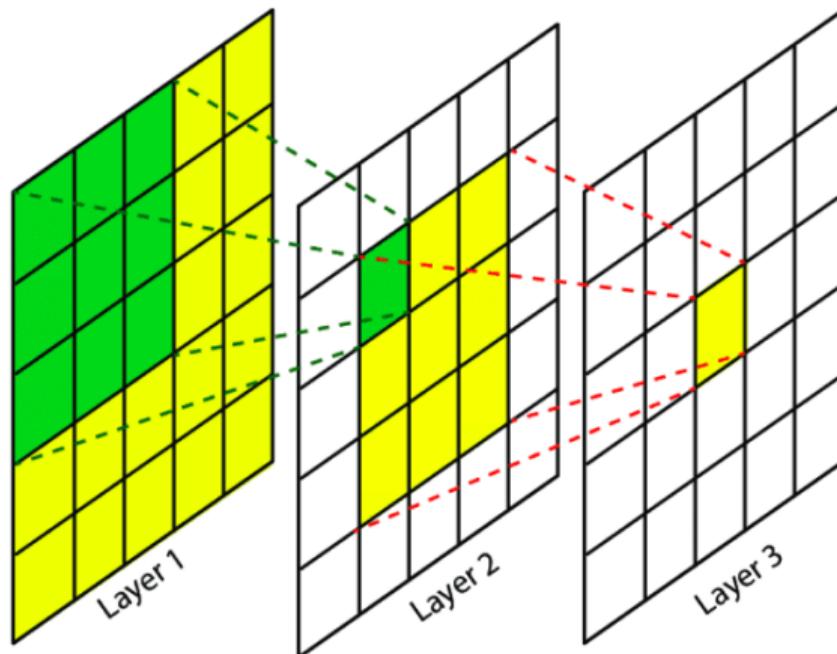


# Поле видимости (receptive field)



<https://theaisummer.com/receptive-field/>

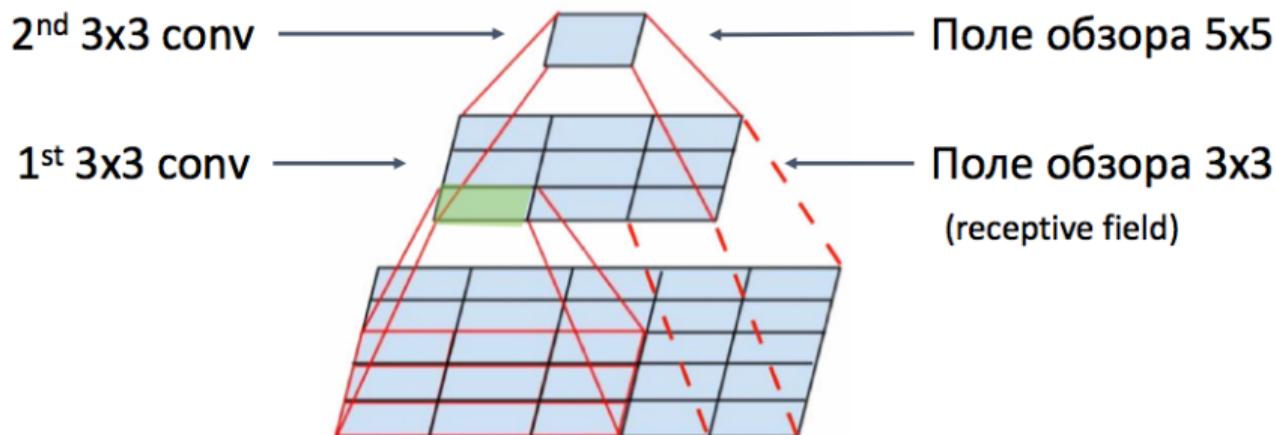
# Поле видимости (receptive field)



<https://theaisummer.com/receptive-field/>

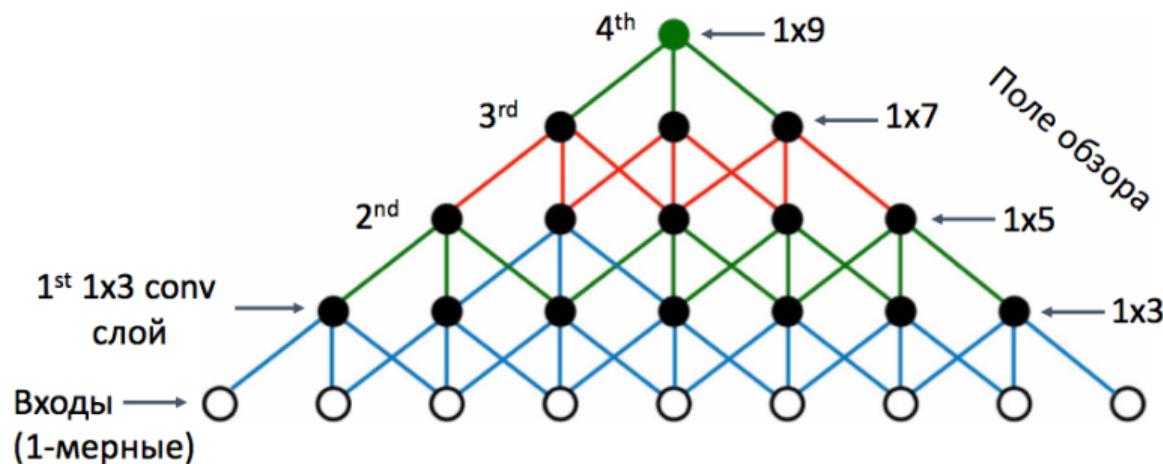
# Одного свёрточного слоя недостаточно

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



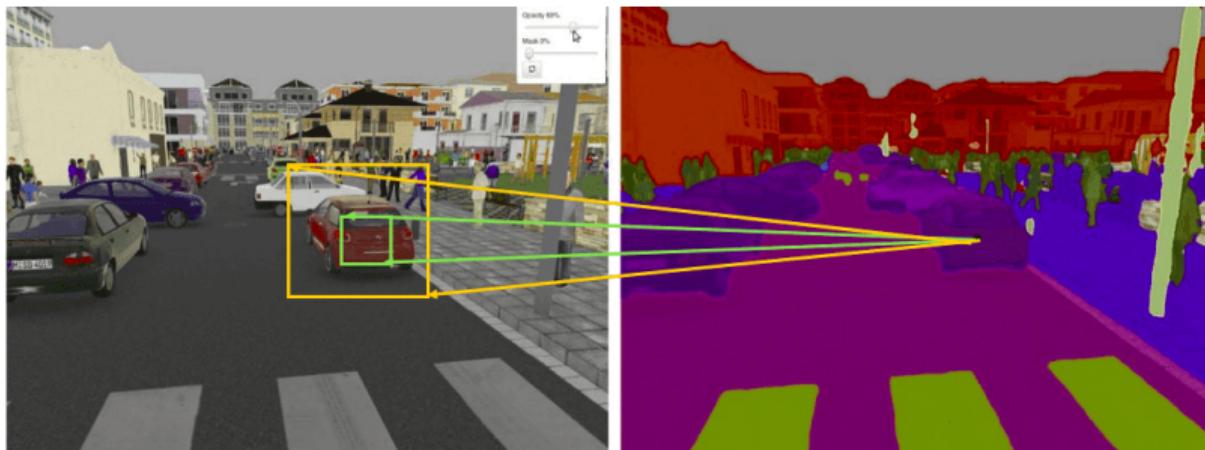
# Одного свёрточного слоя недостаточно

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



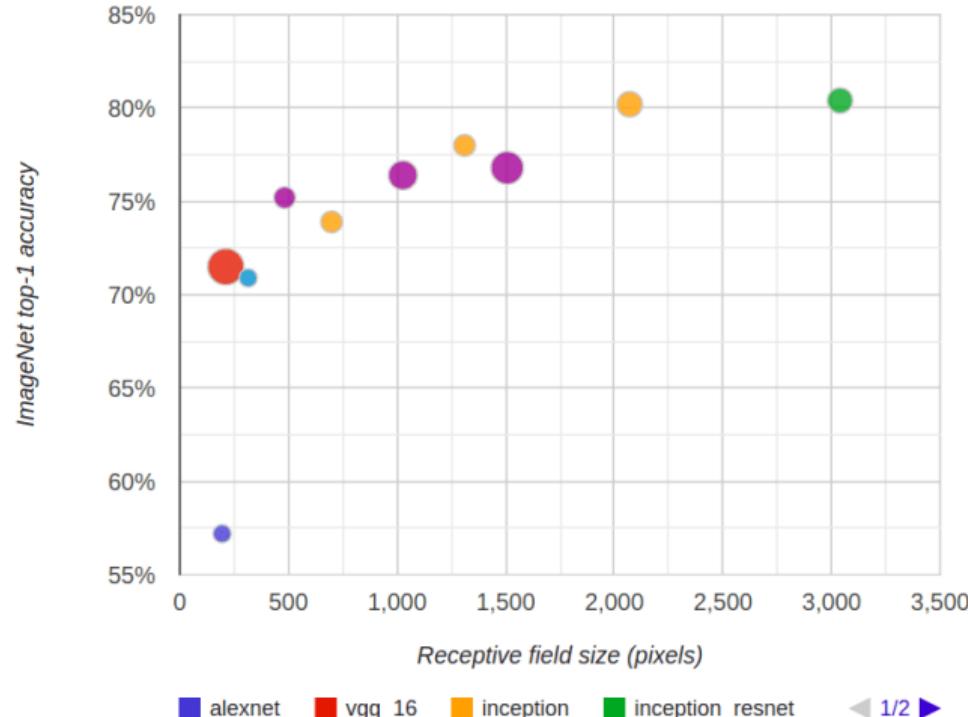
# Помоги CNN найти машину

- Наша цель - детекция автомобилей
- Оранжевым и зелёным выделены два разных поля видимости
- Сеть с маленьким полем видимости не сможет распознать машину



<https://theaisummer.com/receptive-field/>

# Зависимость точности сети от поля обзора

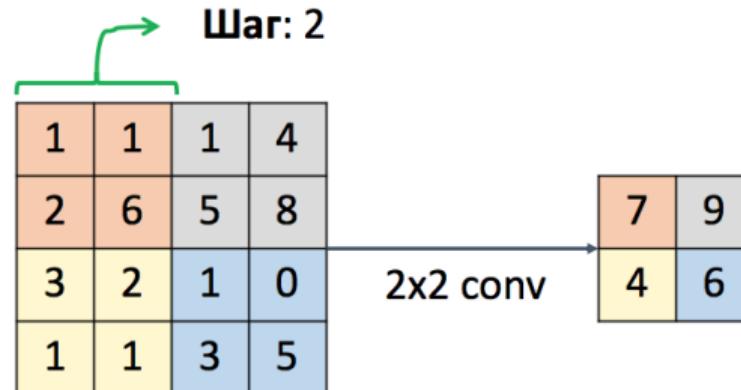


<https://theaisummer.com/receptive-field/>

<https://distill.pub/2019/computing-receptive-fields/>

# Нужно растить поле обзора быстрее!

Можно увеличить шаг свёртки

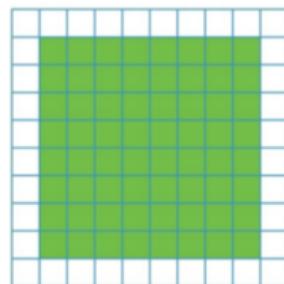


$$* \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}$$

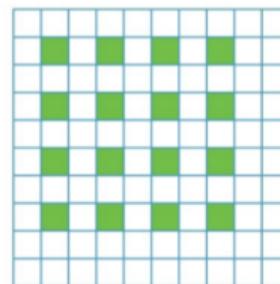
Фильтр

# Свёртки с пропусками (strides)

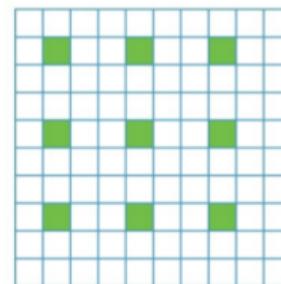
- Пиксели локально скоррелированы – соседние пиксели, как правило, не сильно отличаются друг от друга
- Если будем делать свёртку с каким-то шагом, не потеряем много информации
- Очень агрессивная стратегия снижения размерности картинки



Stride = 1



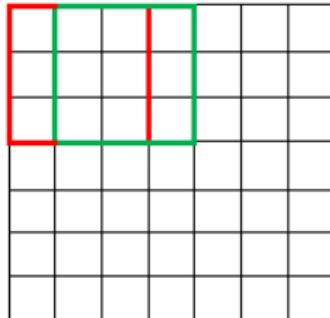
Stride = 2



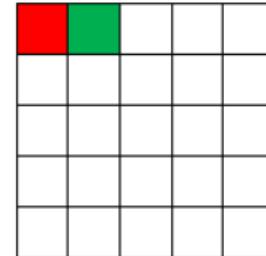
Stride = 3

# Свёртки с пропусками (strides)

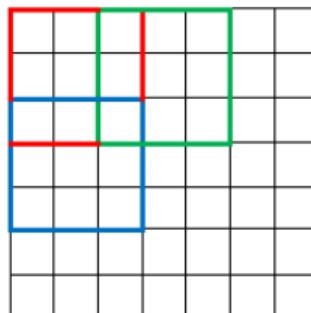
7 x 7 Input Volume



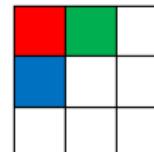
5 x 5 Output Volume



7 x 7 Input Volume



3 x 3 Output Volume



# Пуллинг слой (Pooling)

- Внутри какого-то окна ищем максимум или среднее, размер изображения существенно сокращается, а поле восприятия растёт

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

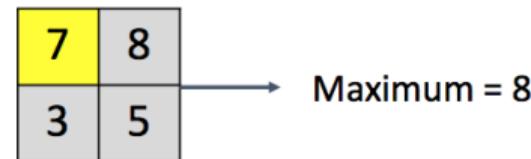
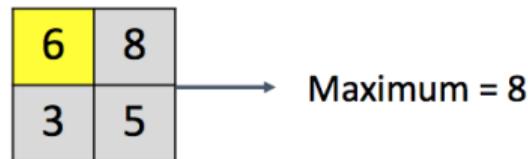
4	7	9
1	6	7
4	5	3

Average  
Pooling

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

# Как считать градиент для пулинга?

- строго говоря: максимум не дифференцируемая функция
- Градиент 0 по немаксимальным входам, потому что при их изменении не меняется выход (максимум)
- Для максимального входа градиент 1.



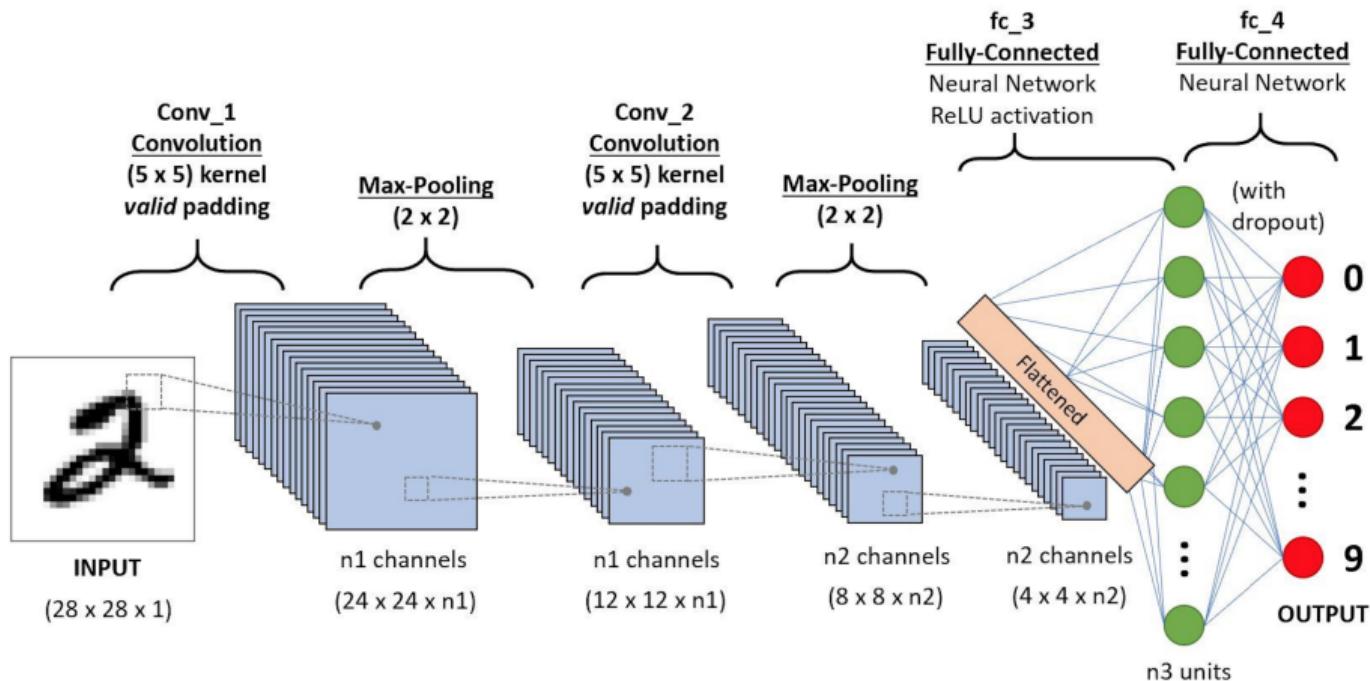
## Зачем это всё?

- Важно следить за тем, чтобы последние свёрточные слои имели размер поля восприятия, сравнимый со всей картинкой
- Нужно понимать, что для более сложных архитектур (батч-норм, скип-конекшн) поле обзора вычисляется сложнее, у пикселей есть больше маршрутов в рамках нейросетки

# Простейшие свёрточные сети



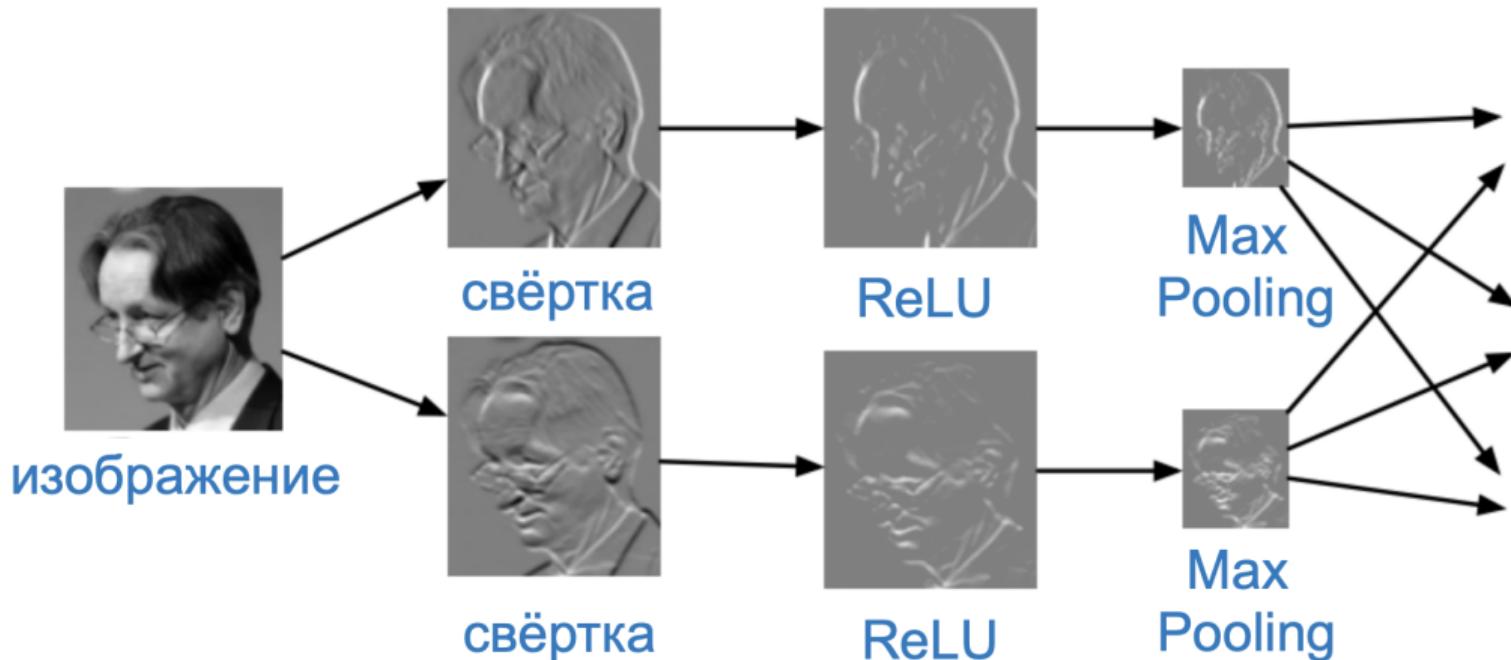
# Типичная архитектура



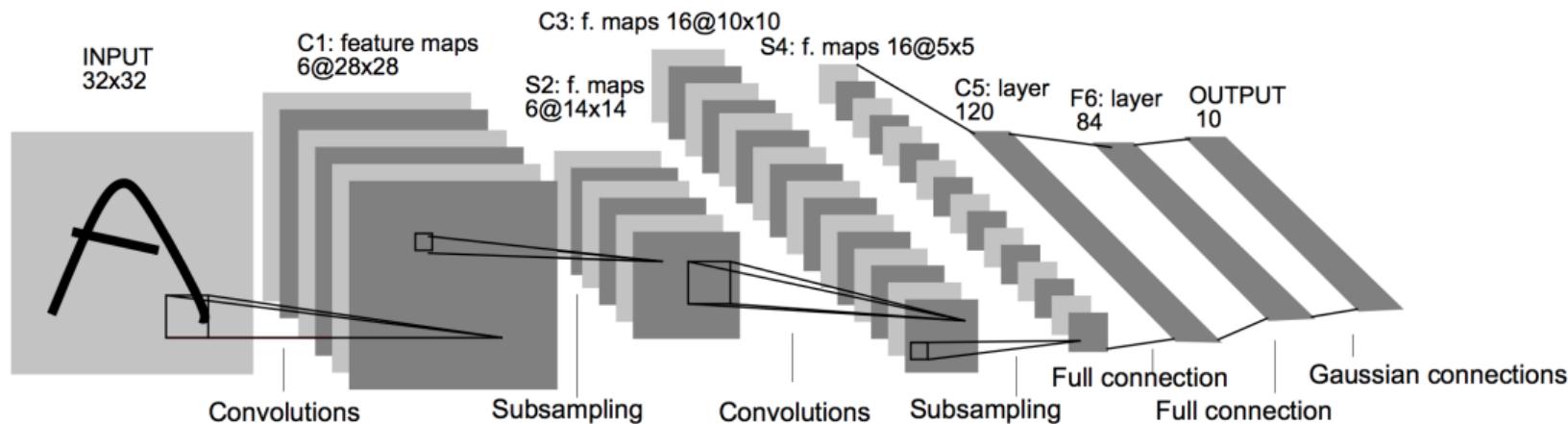
# Типичная архитектура

- Последовательное применение комбинаций вида: свёрточный слой ⇒ нелинейность ⇒ pooling
- Выпрямление (flattening) выхода очередного слоя
- Серия полносвязных слоёв

# Типичная архитектура

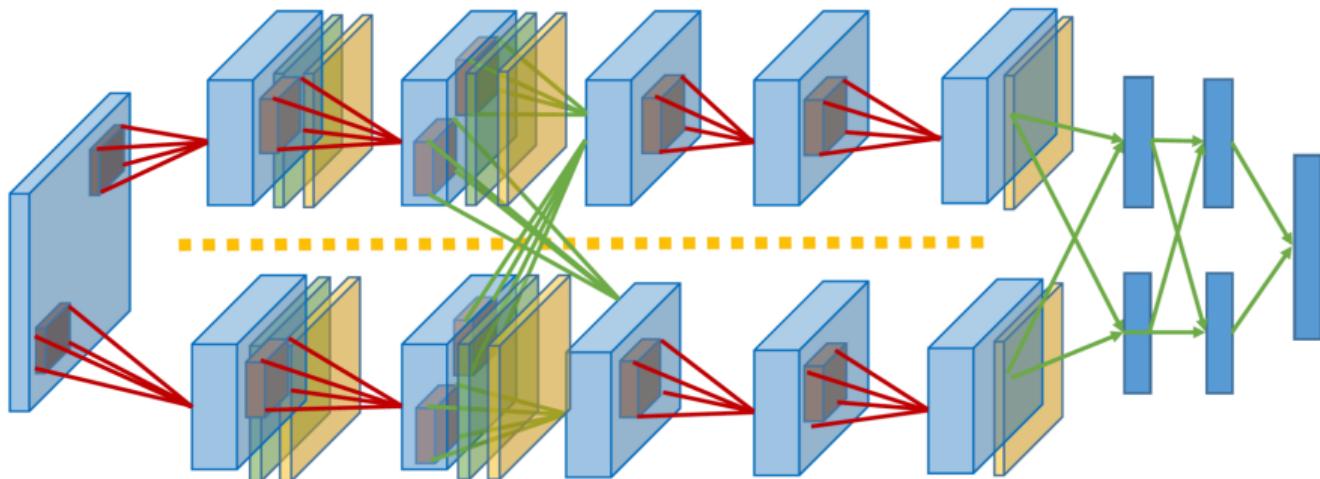
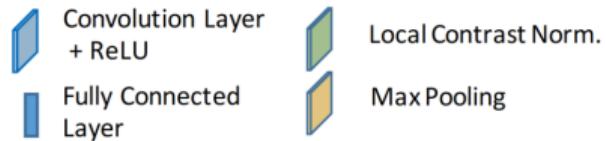


# LeNet (1998)



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

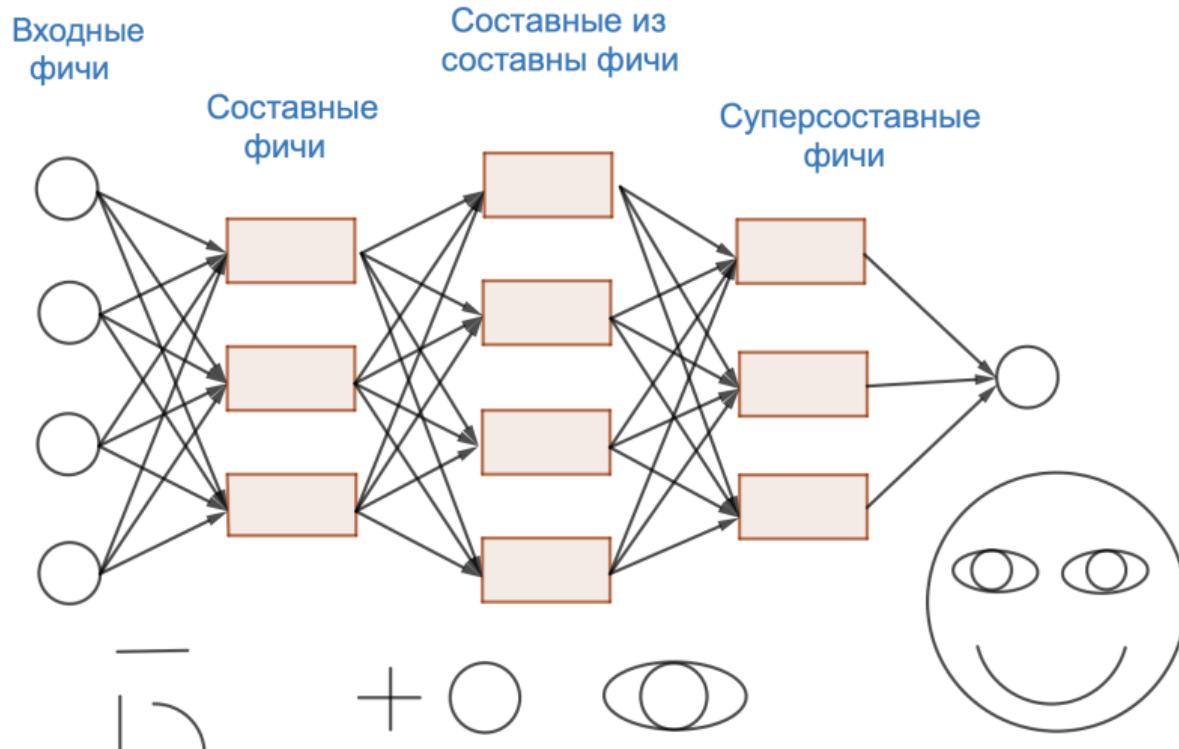
# AlexNet (2012)



# Что выучивают нейросети



# Что выучивают нейросети



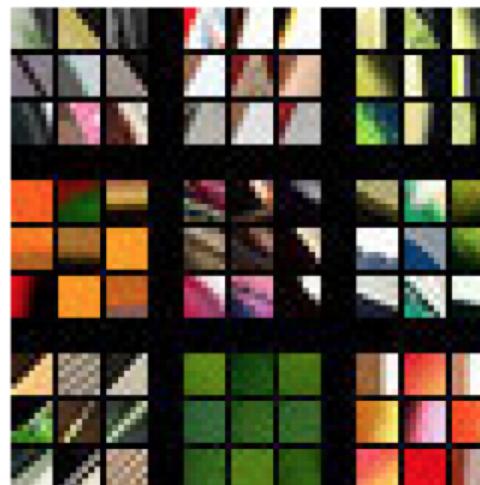
# Что выучивают нейросети

- Первые слои выучивают какие-то низкоуровневые вещи по типу углов/краёв/линий
- Последние слои реагируют уже на целые части изображений
- Нейросеть учит иерархические шаблоны

# Что выучивают нейросети

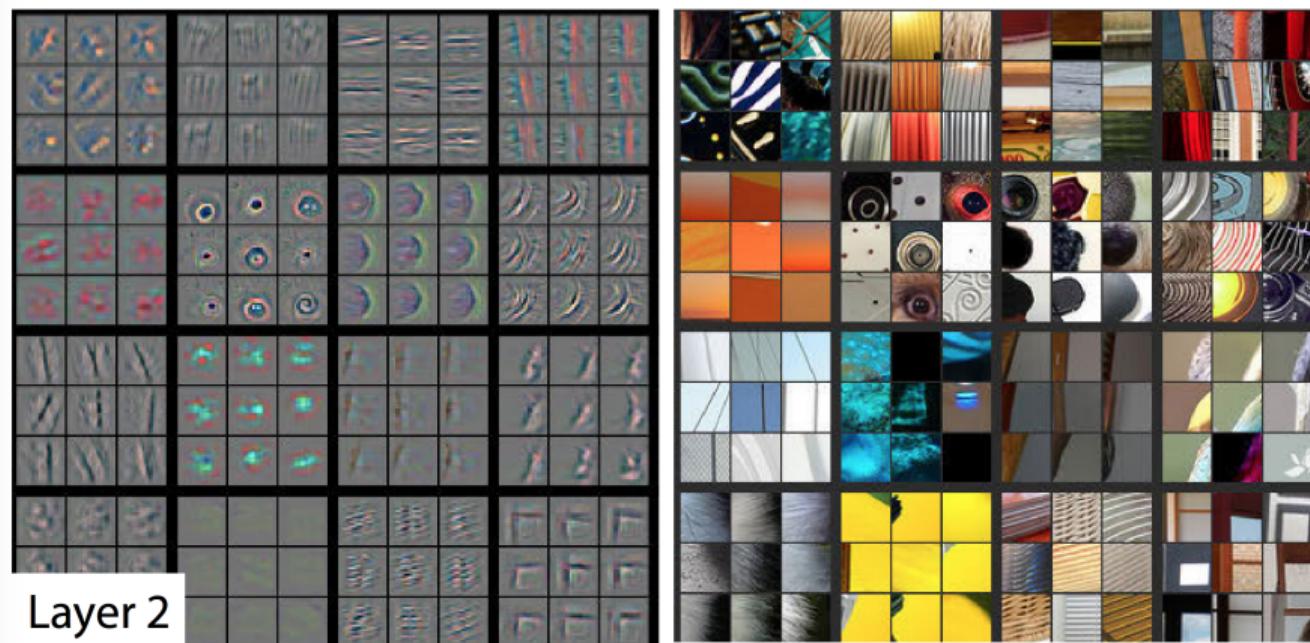


Layer 1



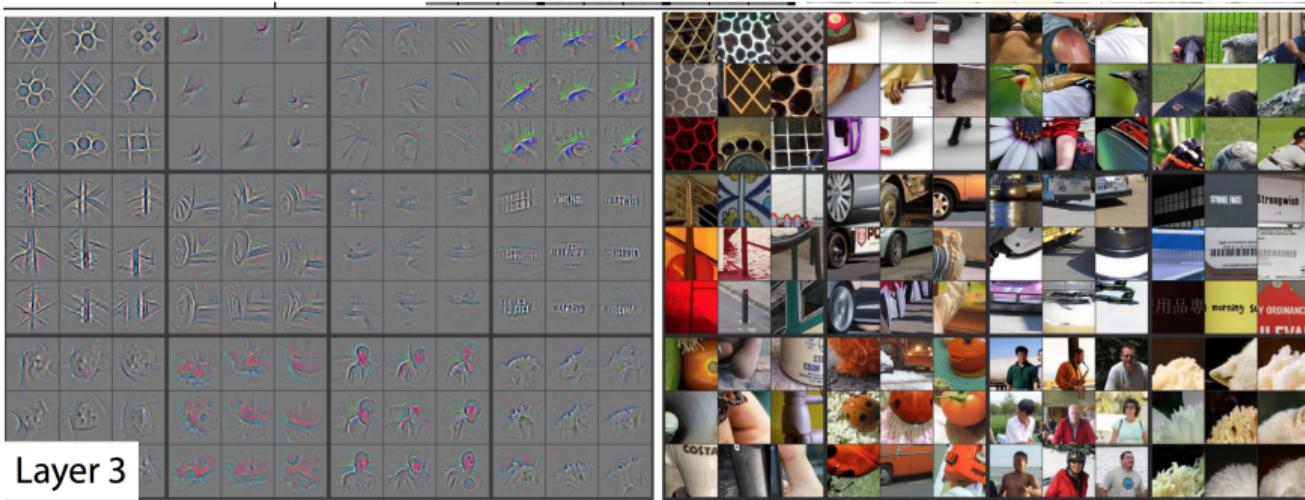
<https://arxiv.org/pdf/1311.2901.pdf>

# Что выучивают нейросети



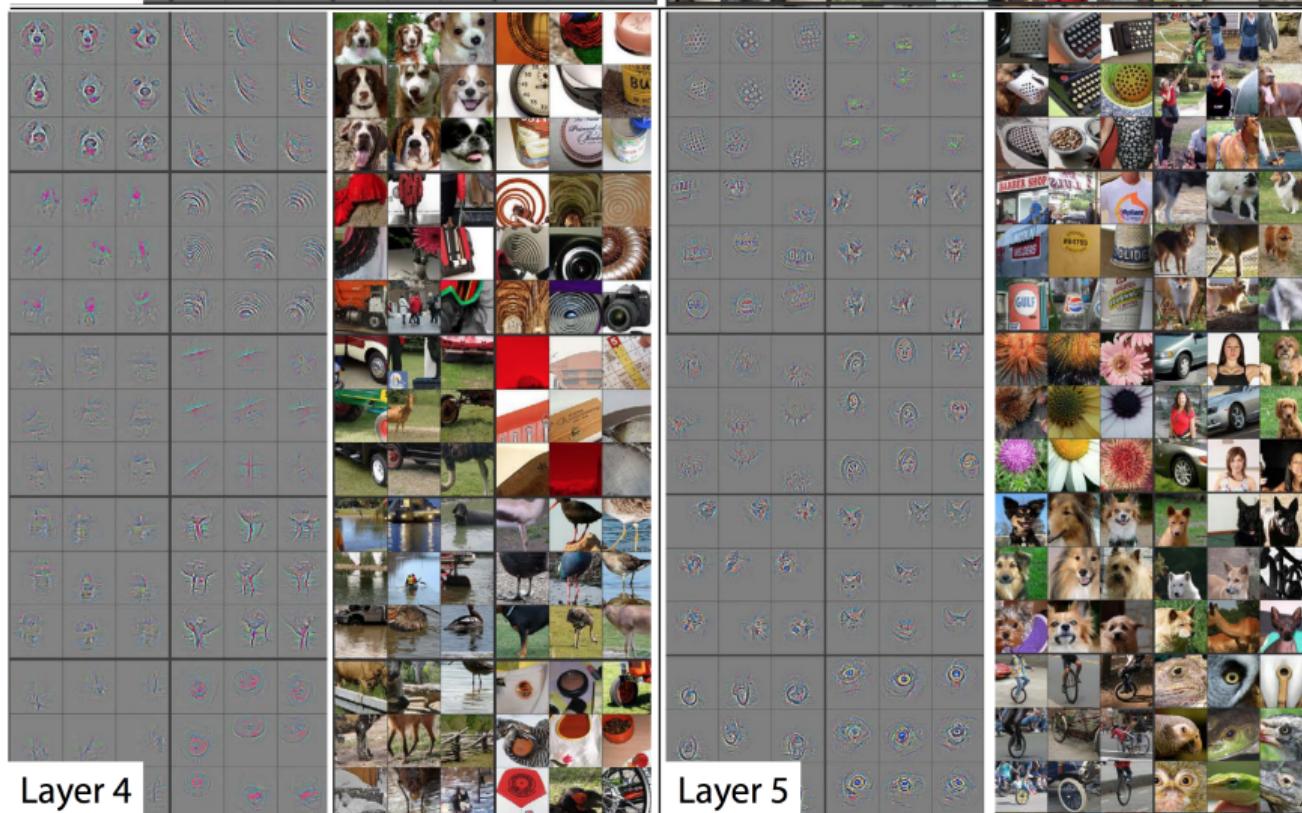
<https://arxiv.org/pdf/1311.2901.pdf>

# Что выучивают нейросети

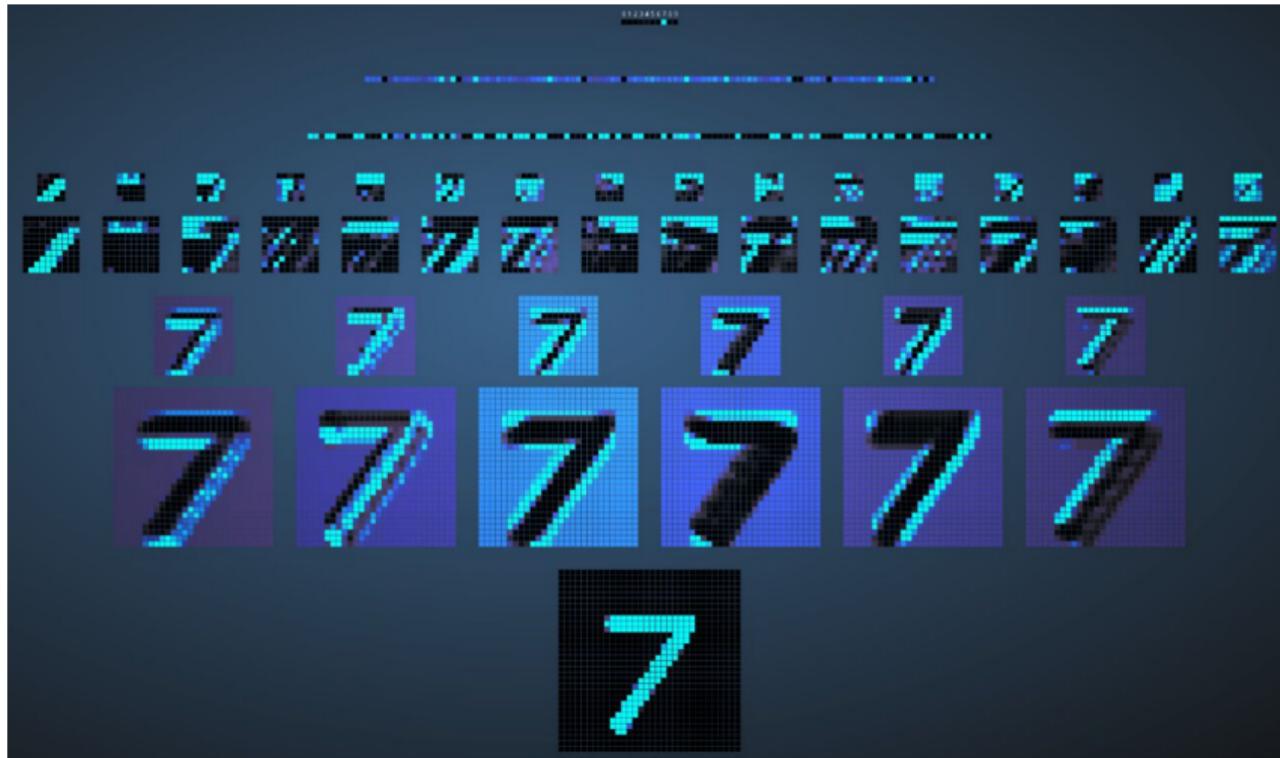


<https://arxiv.org/pdf/1311.2901.pdf>

# Что выучивают нейросети



# Демо: визуализация обученной сети для MNIST



[https://adamharley.com/nn\\_vis/](https://adamharley.com/nn_vis/)

# Представления с последних слоёв

- Выходы с последних слоёв свёрточных нейросетей являются хорошими признаковыми описаниями изображений
- Такие векторные представления называют **эмбеддингами (embeddings)**
- У эмбеддингов нет чёткой интерпретации, цифры в них говорят о наличии каких-то паттернов, на которые настроилась нейросетка
- Эмбеддинги картинок оказываются полезными во многих задачах

Сняться ли нейросетям  
электрические овцы?



# Найдите на картинке овцу



A herd of sheep grazing on a lush green hillside  
Tags: grazing, sheep, mountain, cattle, horse

<https://www.aiweirdness.com/do-neural-nets-dream-of-electric-18-03-02/>

Найдите на картинке овцу



A close up of a lush green field

Tags: grass, field, sheep, standing, rainbow, man

# Найдите на картинке цветы



A group of orange flowers in a field

*Image credit: Richard Leeming @RM\_Leeming - CC-BY license*

# Найдите на картинке птиц



NeuralTalk2: A flock of birds flying in the air

Microsoft Azure: A group of giraffe standing next to a tree

*Image: Fred Dunn, <https://www.flickr.com/photos/gratapictures> - CC-BY-NC*

# Собираем свою собственную CNN