

Глубокое обучение и вообще

Ульянкин Филипп

7 февраля 2021 г.

Посиделка 7: Свёрточные нейронки

Agenda

- Как видит компьютер
- Свёртка и пулинг
- Data augmentation
- Собираем свою свёрточную сетку

Затравка (2006)



Please click on all the images that show cats:

adopt me	adopt me	adopt me	adopt me
adopt me	adopt me	adopt me	adopt me
adopt me	adopt me	adopt me	adopt me
adopt me	adopt me	adopt me	adopt me

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.
- 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. Refresh

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

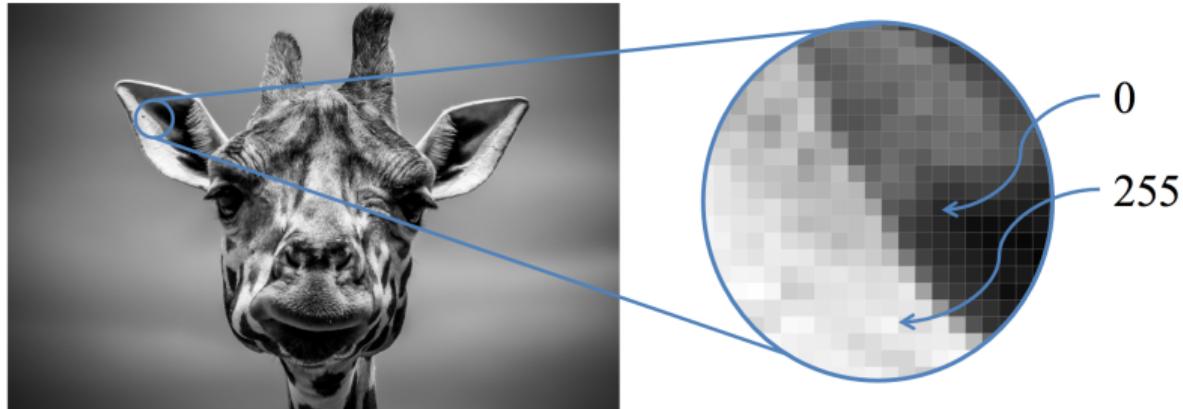
Gold Silver Bronze

#	△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

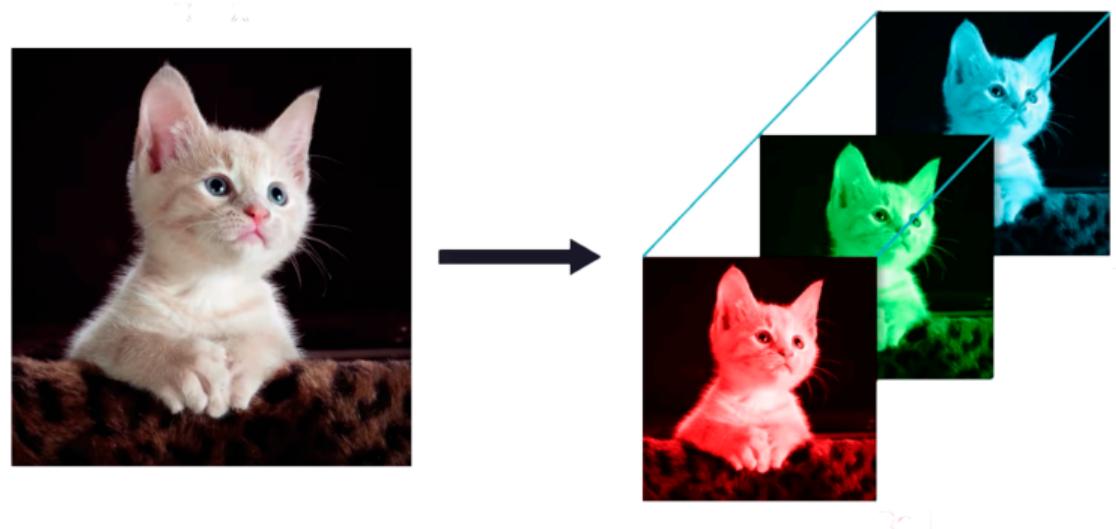
Картина – тензор

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

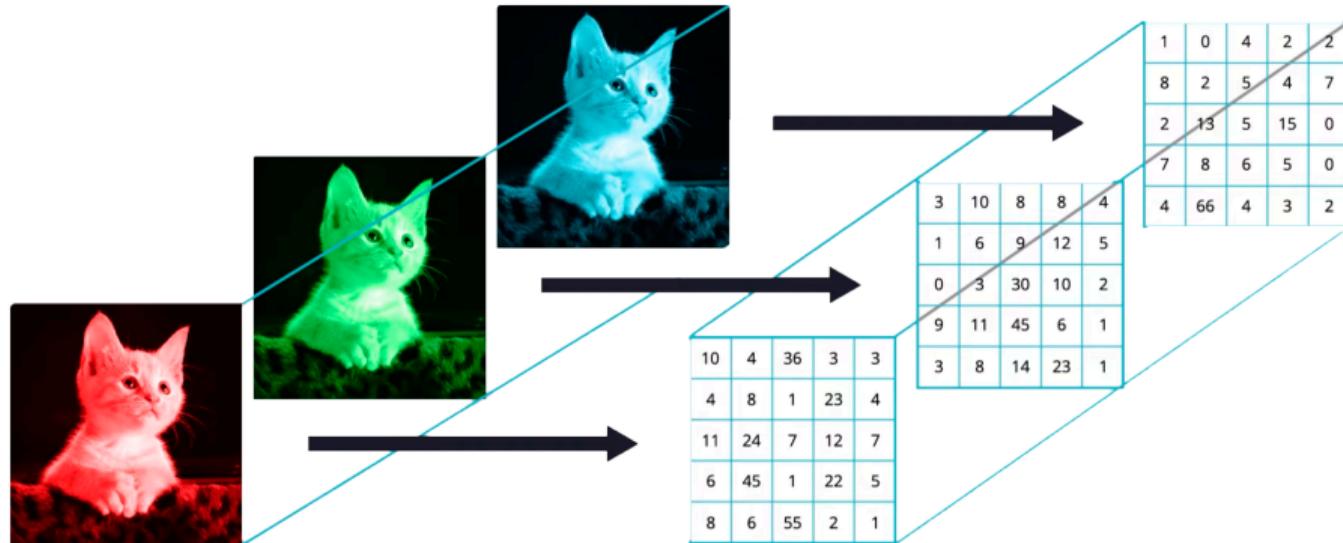


Картина – тензор

- Цветное изображение имеет три канала пикселей: красный, зелёный и синий (rgb), размерность изображения $5 \times 5 \times 3$



Картина – тензор



Картина – тензор

$5 \times 5 \times 3$

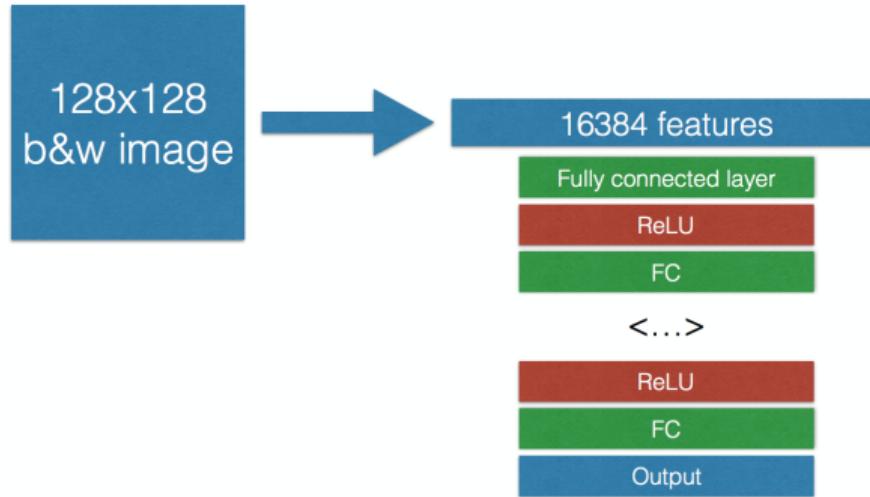


3D Array

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

5

Обычная сеть

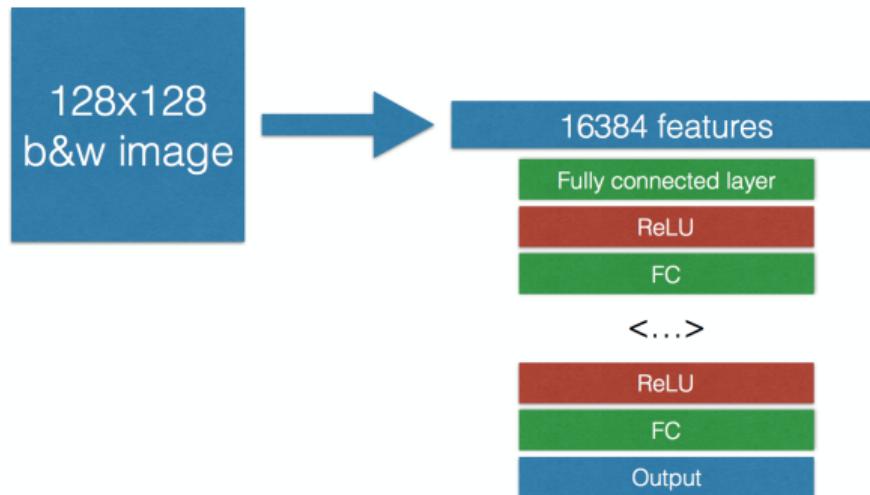


- Развернём картинку в вектор \Rightarrow **очень много весов**
- Сетка может легко переобучиться
- Лучшая борьба с переобучением - уменьшение числа параметров





Обычная сеть



- Изображение в разных местах картинки даёт разные веса
- Теряется информация о взаимном расположении пикселей

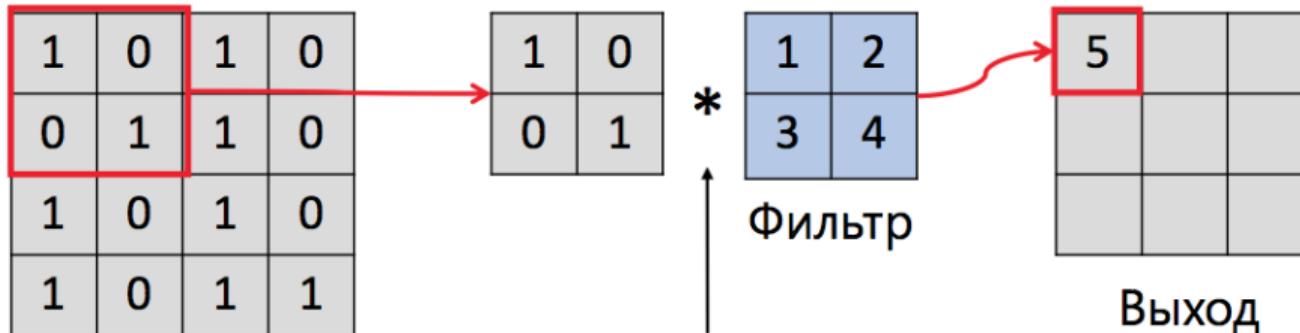
Мы хотим, чтобы...

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

⇒ свёртка

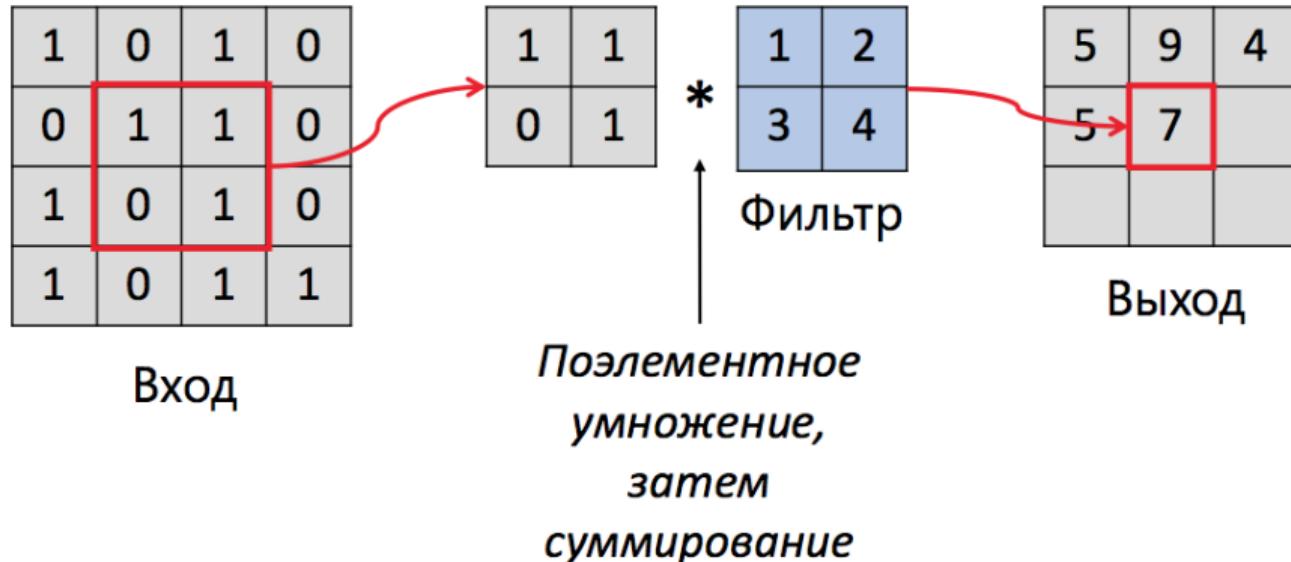
Свёртка

Свёртка



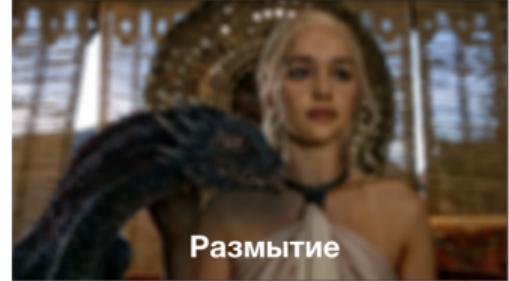
*Поэлементное
умножение,
затем
суммирование*

Свёртка

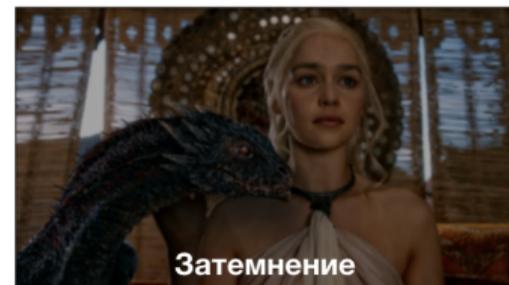




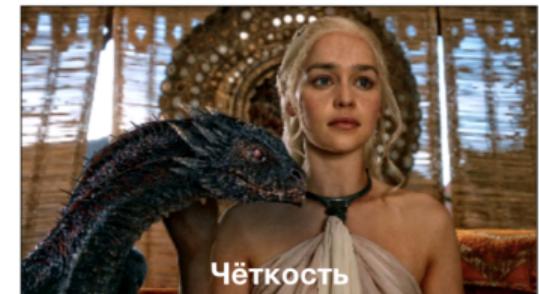
$$\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



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



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



```
[[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

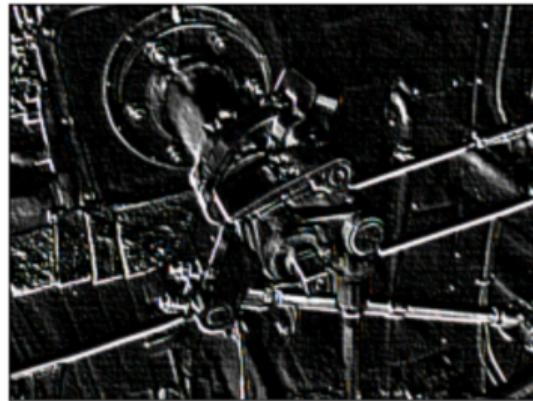
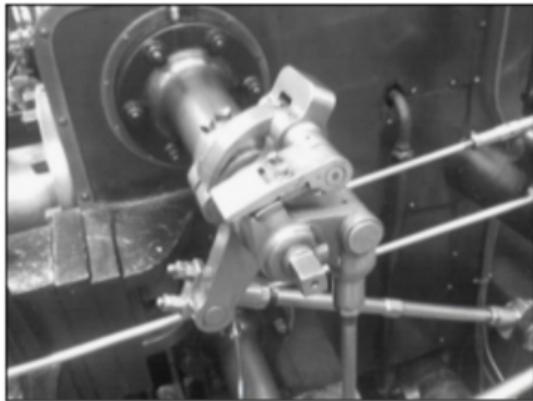
Зеленый

Синий

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

Kernel

Output

The diagram shows a 4x4 input matrix and a 3x3 kernel matrix being multiplied to produce a 3x3 output matrix. The input matrix has its last three columns highlighted in green. The kernel matrix has its last two rows highlighted in green. The resulting output matrix has its second column highlighted in green.

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

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

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

Input

*

1	0
0	1

=

0	0	0
0	1	0
0	0	2

Kernel

Output

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

Input

*

1	0
0	1

=

2	0	0
0	1	0
0	0	0

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

=

0	0	0
0	1	0
0	0	2

Output

Max = 2

↑
Didn't
change
↓

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

Input

*

1	0
0	1

=

2	0	0
0	1	0
0	0	0

Output

Max = 2

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 в виде формулы

$$\begin{matrix} H \\ \text{ } \\ W \end{matrix} \quad \begin{matrix} C_{in} \\ \text{ } \\ \text{ } \end{matrix} \quad * \quad \begin{matrix} \text{фильтр} \\ W_k \times H_k \times C_{in} \end{matrix} = \quad \begin{matrix} \text{Результат} \\ \text{ } \end{matrix}$$

The diagram illustrates the convolution process. On the left, an input image of a dog's head is shown within a 3D volume cube. The dimensions are labeled H (height), W (width), and C_{in} (input channels). A 3x3 kernel is applied to the input, indicated by a small 3D cube with a grid pattern. The kernel values are displayed as a 3x3 matrix:

-1	-1	-1
-1	8	-1
-1	-1	-1

The result of the convolution is a processed image of the dog's head, where the features have been highlighted. A red square box highlights a specific pixel in the result image.

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>

Дополнение (Padding)

Дополнение (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

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

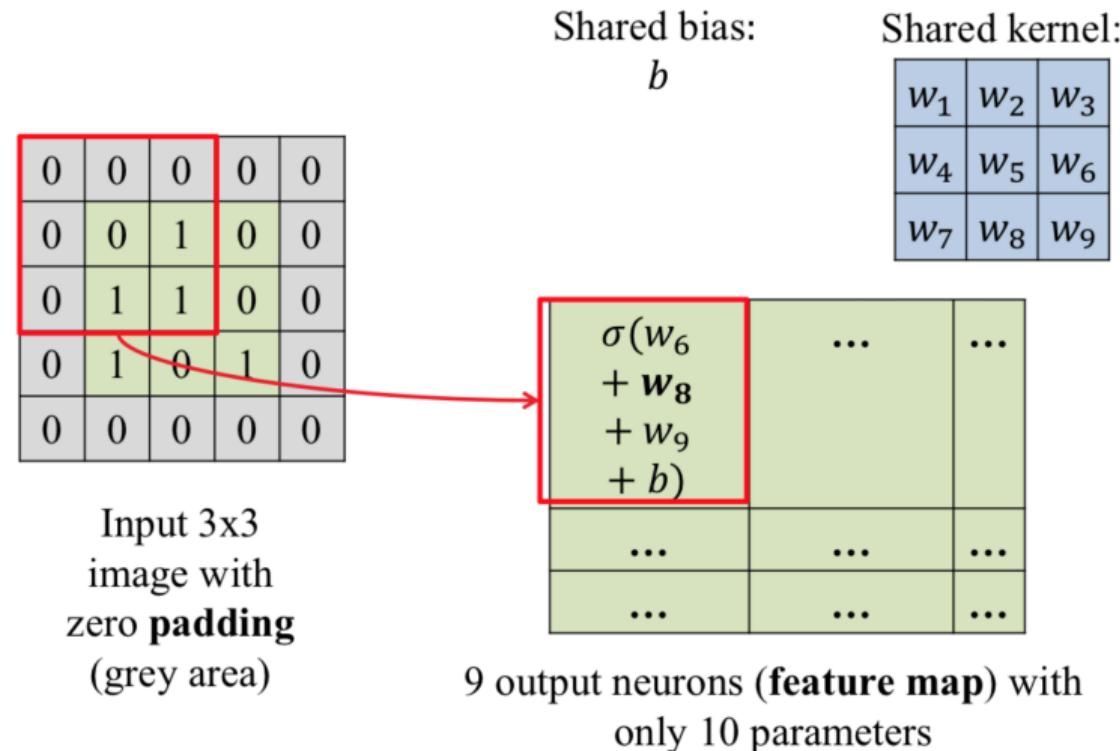
<https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/>

Padding

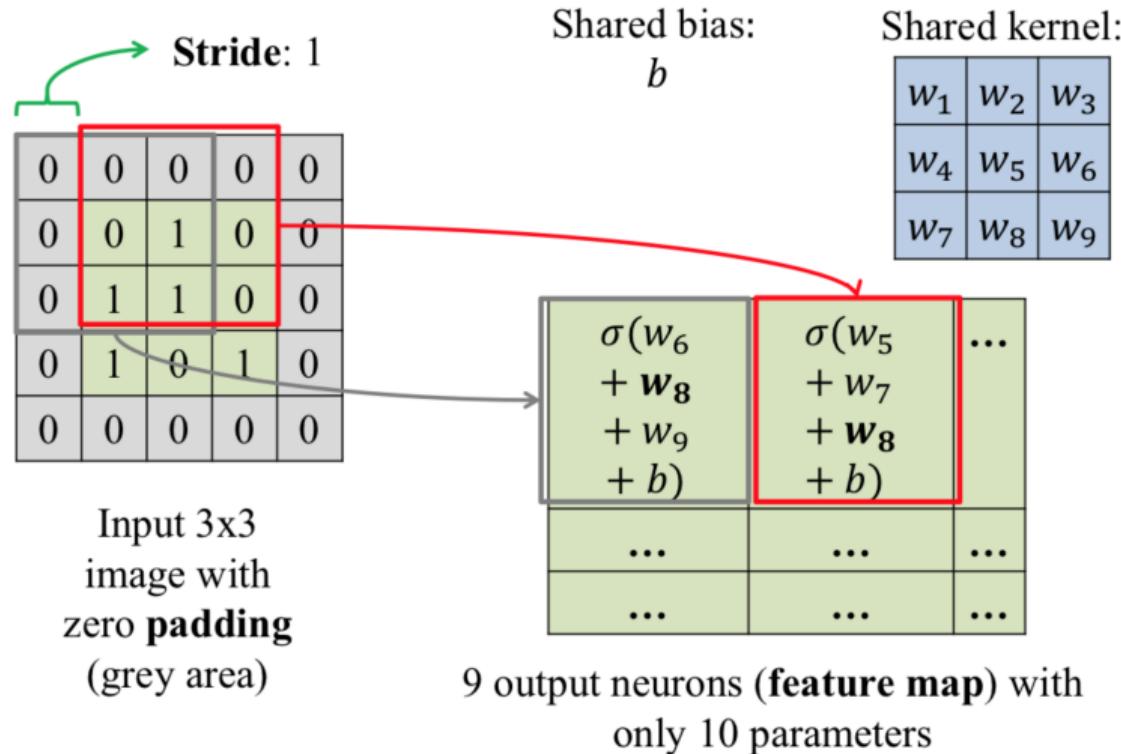
- Паддинг позволяет контролировать размер выходных изображений
- Паддинг позволяет учитывать даже объекты на краях
- Разные типы паддингов допускают разные способы переобучения под края

Свёрточный слой

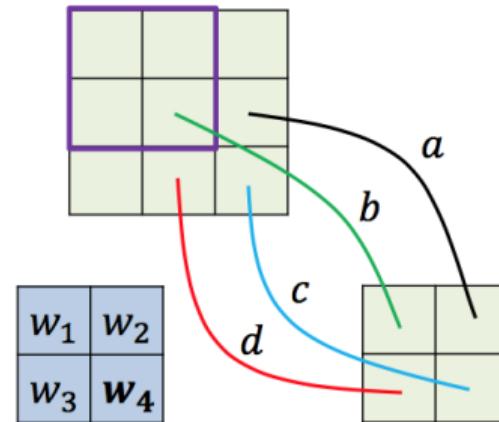
Свёрточный слой



Свёрточный слой



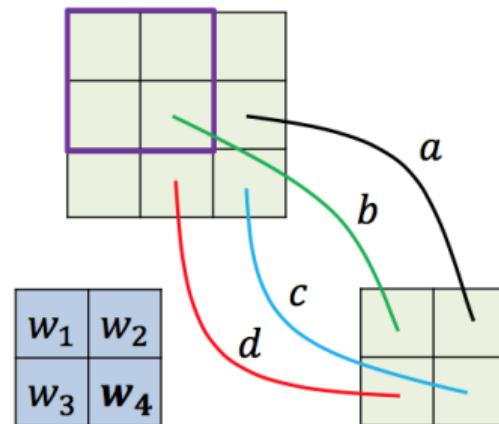
Backpropagation для свёрточного слоя



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

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

Backpropagation для свёрточного слоя

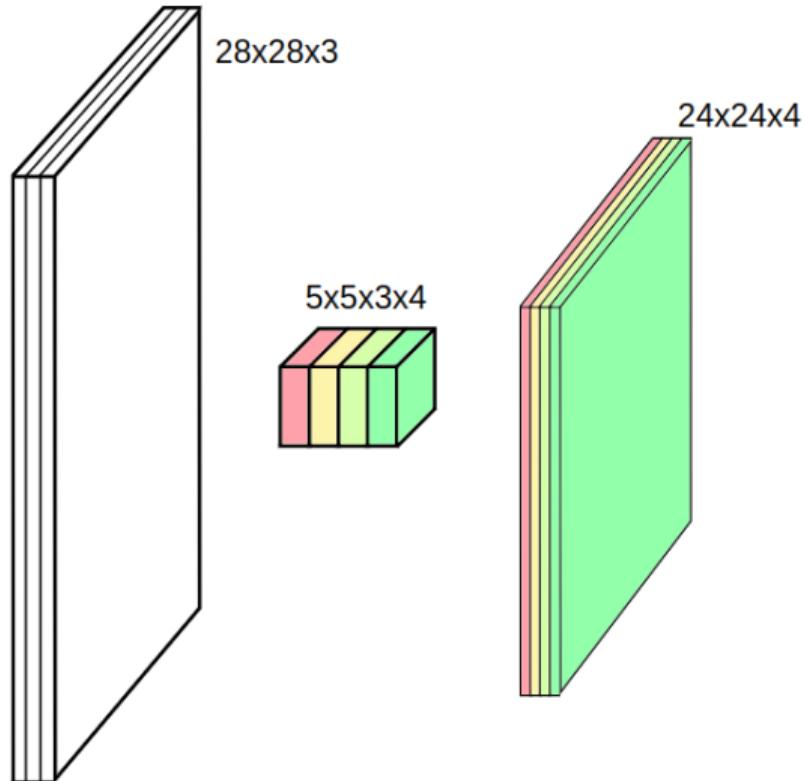


$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \cdot x_{11} + \frac{\partial L}{\partial b} \cdot x_{12} + \frac{\partial L}{\partial c} \cdot x_{22} + \frac{\partial L}{\partial d} \cdot x_{12}$$

Свёрточный слой

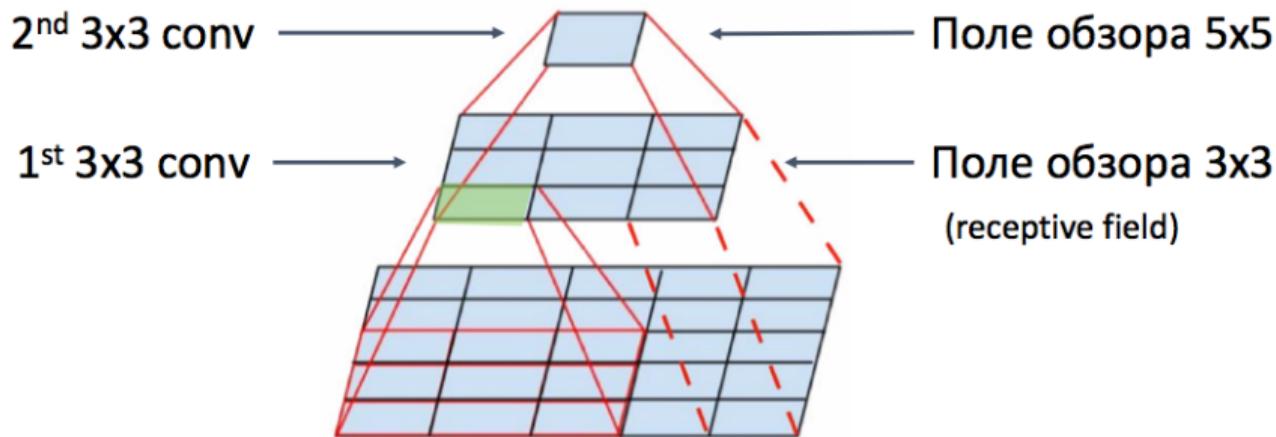
- Слой действует одинаково для каждого участка картинки, в отличие от полносвязного
- Нужно оценивать меньшее количество параметров
- Слой учитывает взаимное расположение пикселей
- Можно учить тем же самым backpropagation
- Свёрточный слой - это полносвязный слой с ограничениями

Одного ядра недостаточно



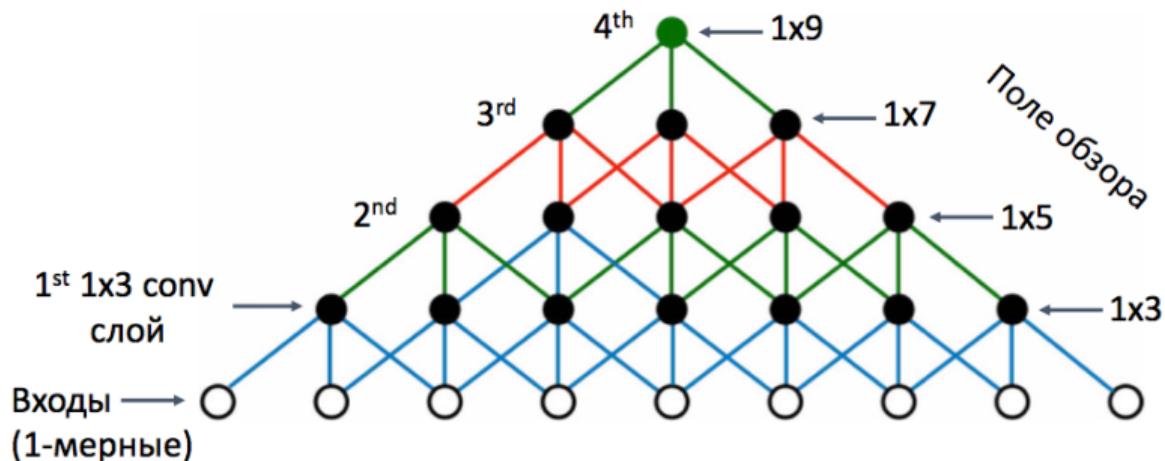
Одного свёрточного слоя недостаточно

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



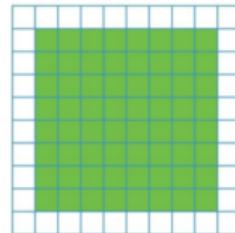
Одного свёрточного слоя недостаточно

- 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

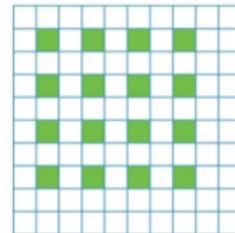


Нужно растить поле обзора быстрее!

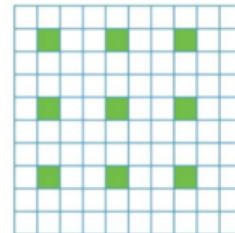
- Пиксели локально скоррелированы – соседние пиксели, как правило, не сильно отличаются друг от друга
- Если будем делать свёртку с каким-то шагом, сэкономим мощности компьютера и не потеряем в информации



Stride = 1



Stride = 2

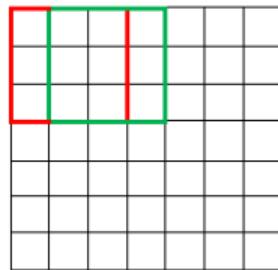


Stride = 3

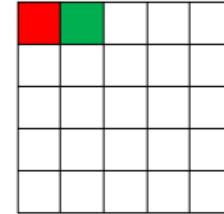
- Очень агрессивная стратегия снижения размерности картинки

Сдвиг (Stride)

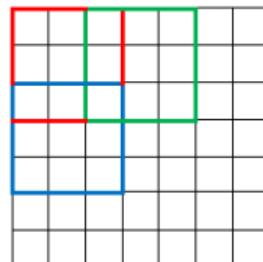
7 x 7 Input Volume



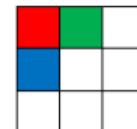
5 x 5 Output Volume



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

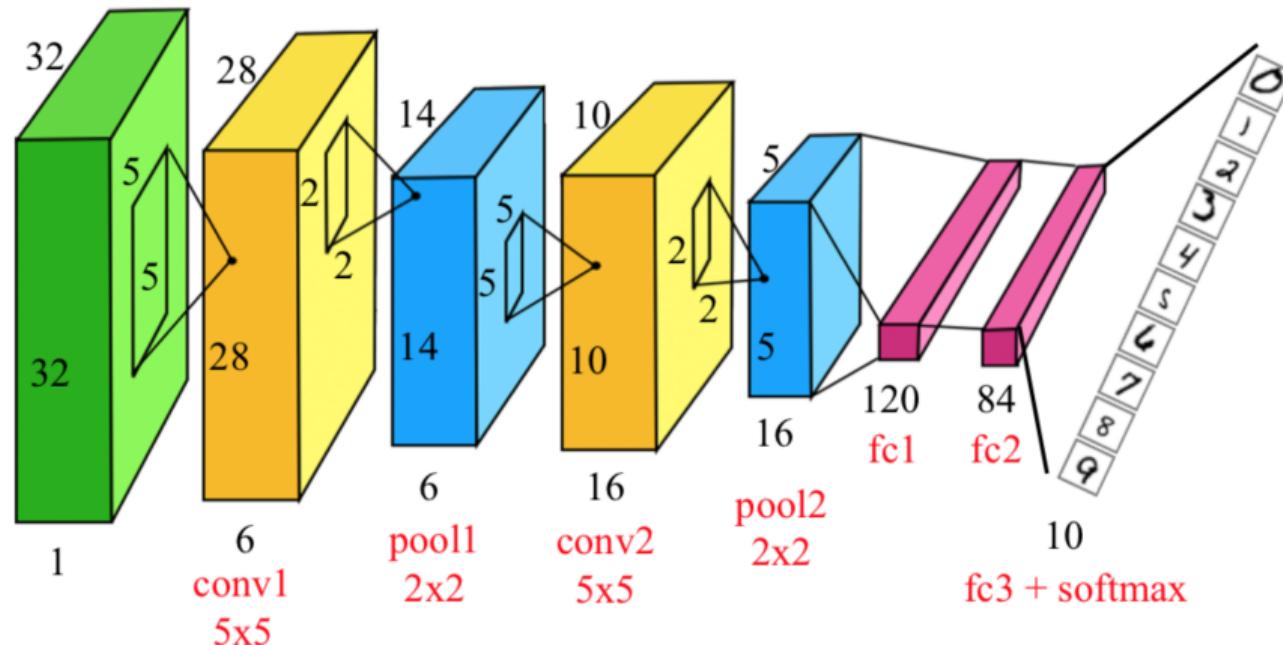
4	7	9
1	6	7
4	5	3

Average
Pooling

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

Простейшая CNN

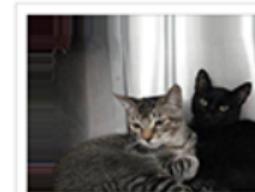
Нейросеть LeNet-5 (1998) для распознавания рукописных цифр



Data augmentation

Data augmentation

- В сетке может быть миллионы параметров
- Естественная регуляризация, дополнительная регуляризация, генерация новых данных (data augmentation)
- Генерируем новые цвета, сдвигаем, искажаем и тп



https://www.tensorflow.org/tutorials/images/data_augmentation

Data augmentation



<https://github.com/albumentations-team/albumentations>

Data augmentation

- Сдвиги (вместо них лучше пулинг)
- Увеличение, уменьшение
- Повороты
- Искажение
- Затенение
- Смена стиля

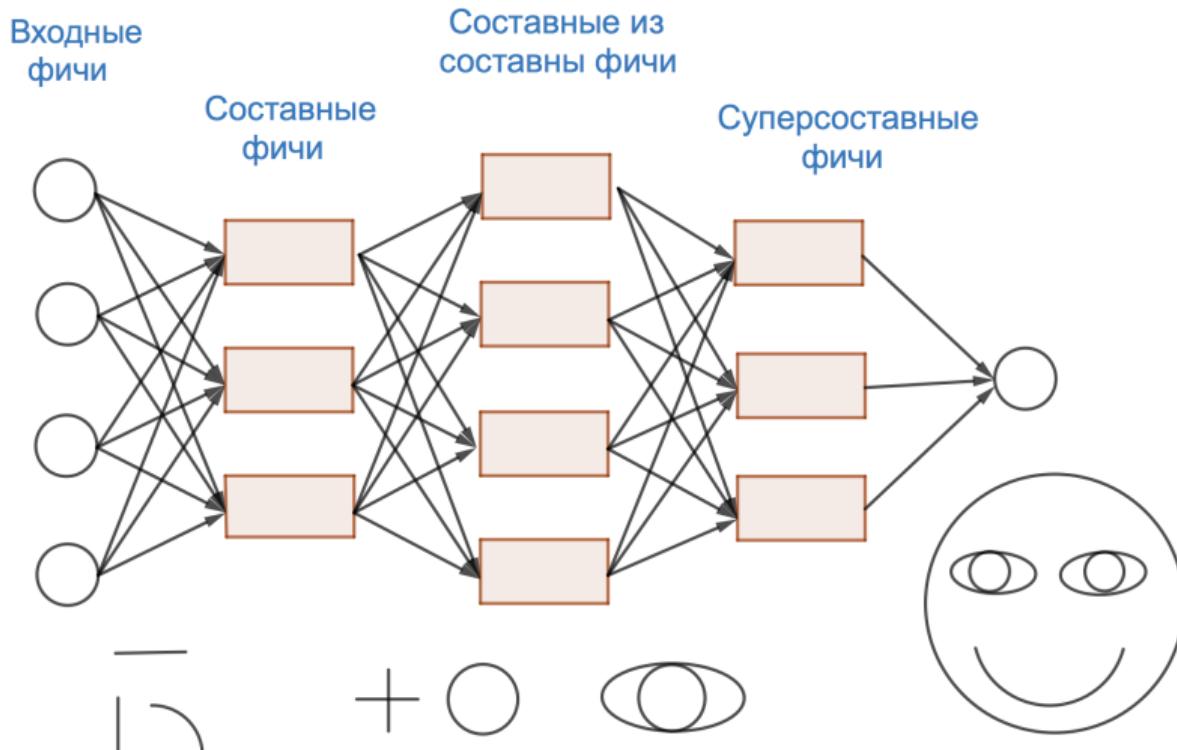


Data augmentation

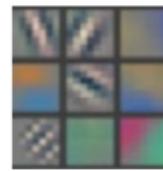
- Бесплатное расширение обучающей выборки
- В некотором смысле регуляризация модели
- Обычно аугментации применяют к случайным картинкам из текущего батча

Что выучивают нейросети

Что выучивают нейросети



Что выучивают нейросети

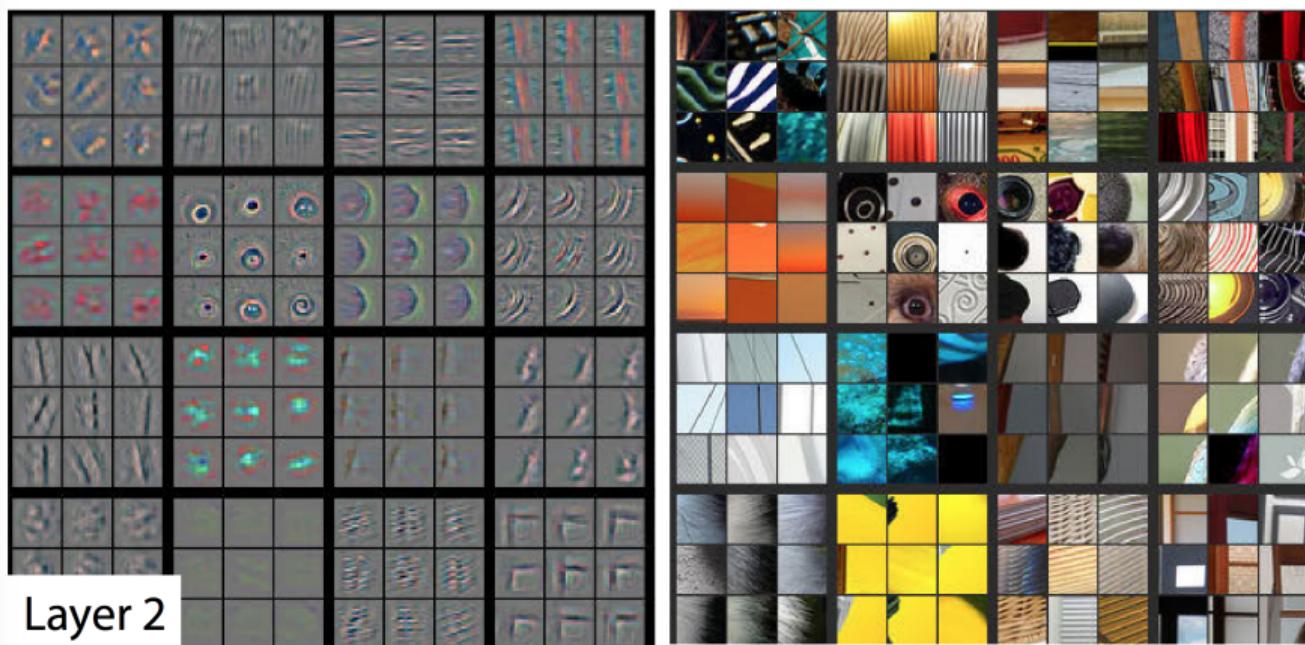


Layer 1



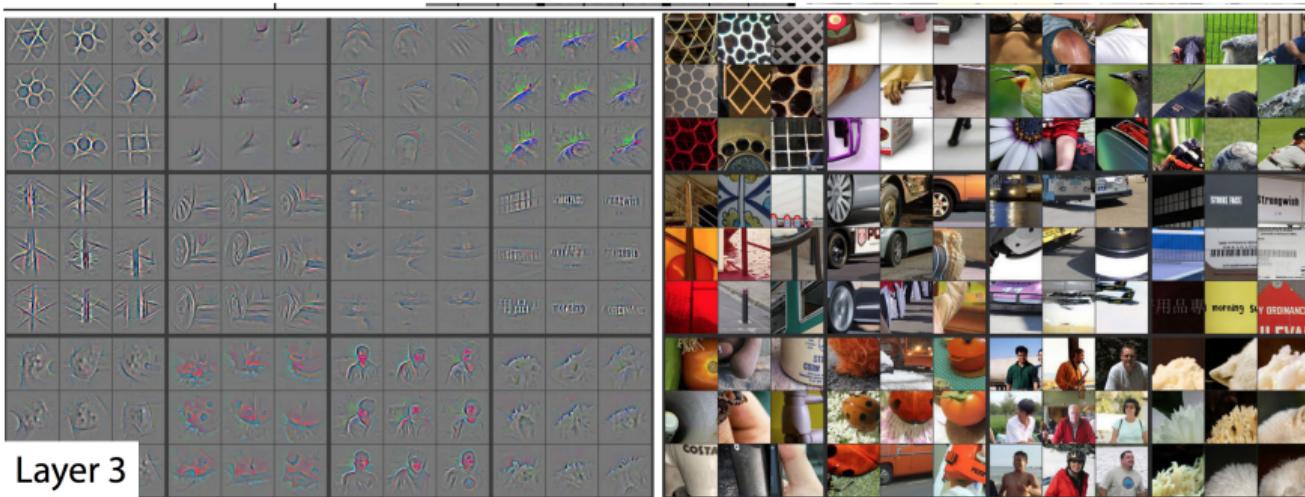
<https://arxiv.org/pdf/1311.2901.pdf>

Что выучивают нейросети



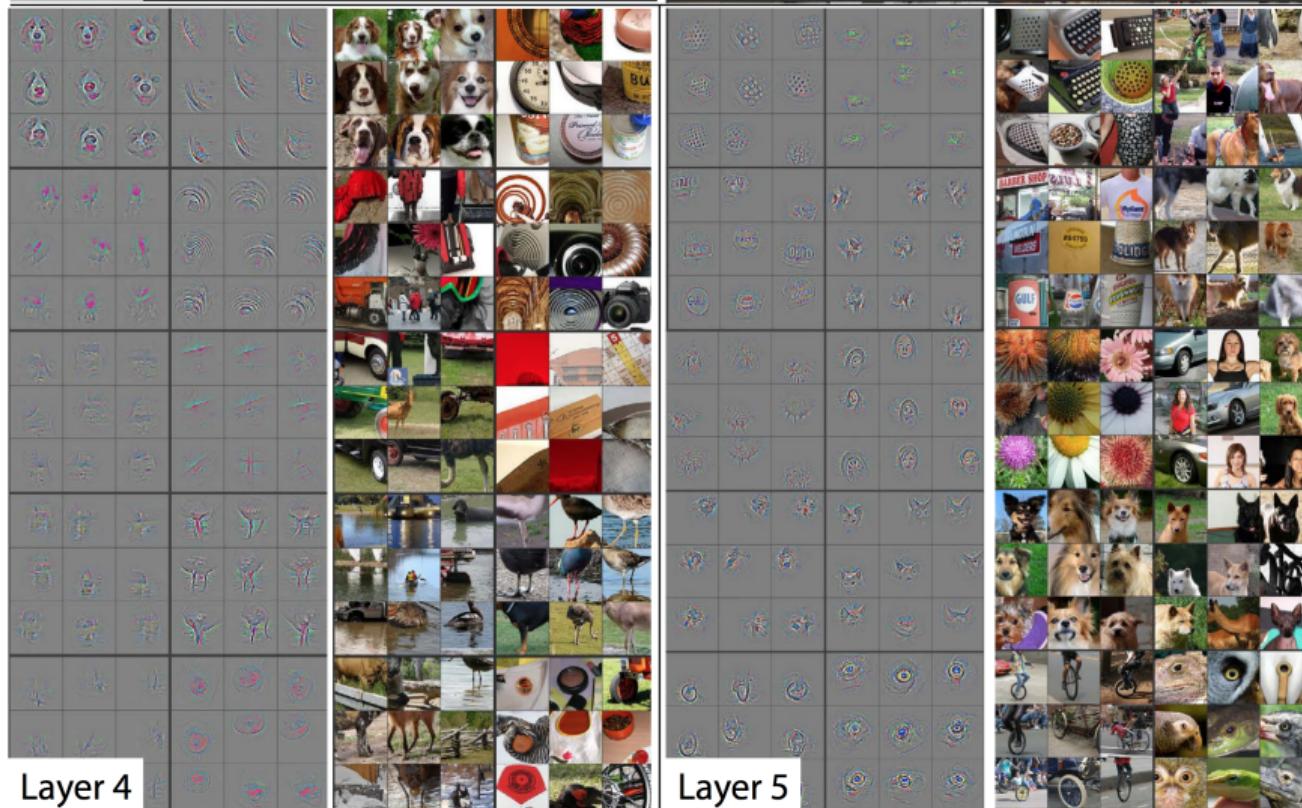
<https://arxiv.org/pdf/1311.2901.pdf>

Что выучивают нейросети



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

Что выучивают нейросети



Представления с последних слоёв

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

Собираем свою собственную CNN