

Глубокое обучение и вообще

Ульянкин Филипп и Соловей Влад

5 февраля 2020 г.

Посиделка 5: Свёрточные сетки

Agenda

- Как видит компьютер
- Свёртка
- Пуллинг
- Собираем свою свёрточную сетку
- Data augmentation

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

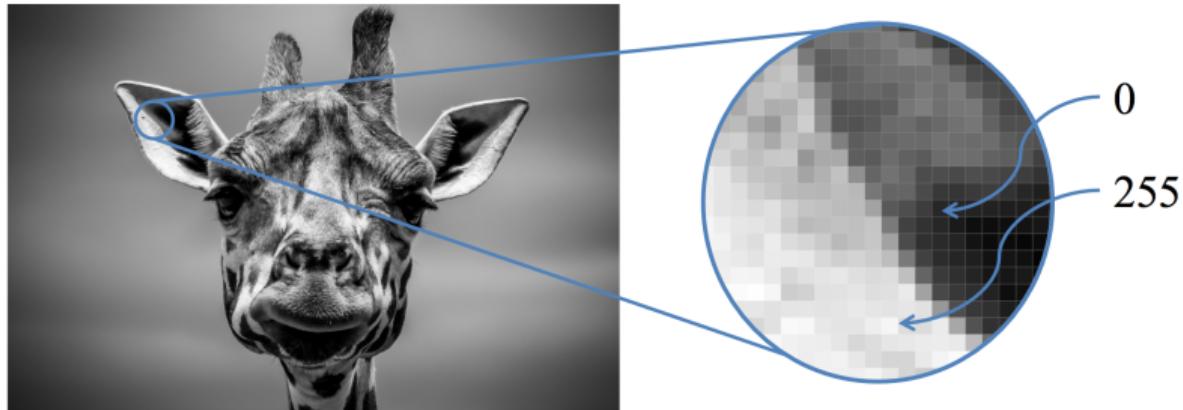
#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	Pierre Sermanet			0.98914	5	6y
2	▲ 4	orchid			0.98308	17	6y
3	—	Owen			0.98171	15	6y
4	—	Paul Covington			0.98171	3	6y

Как видит компьютер



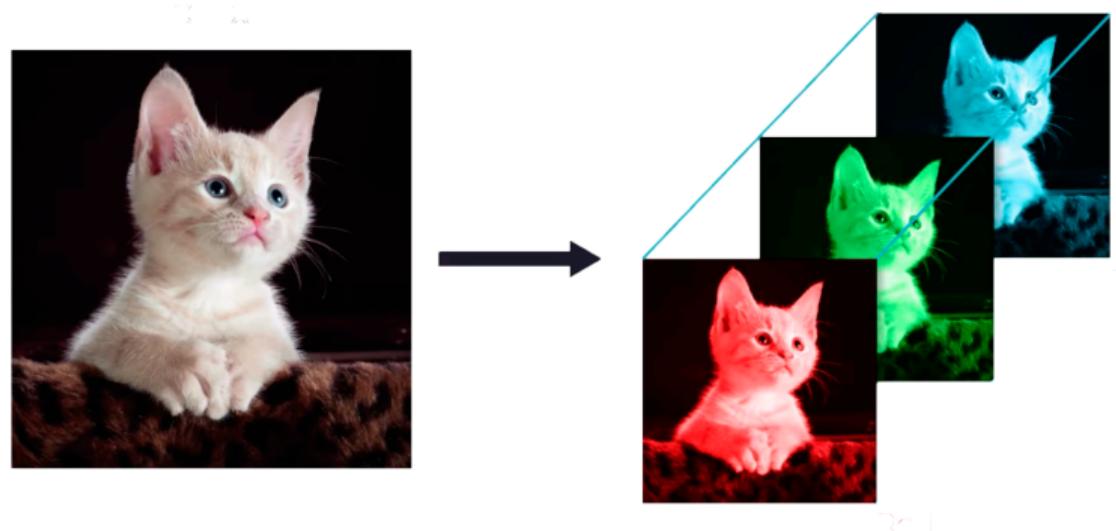
Картина – тензор

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

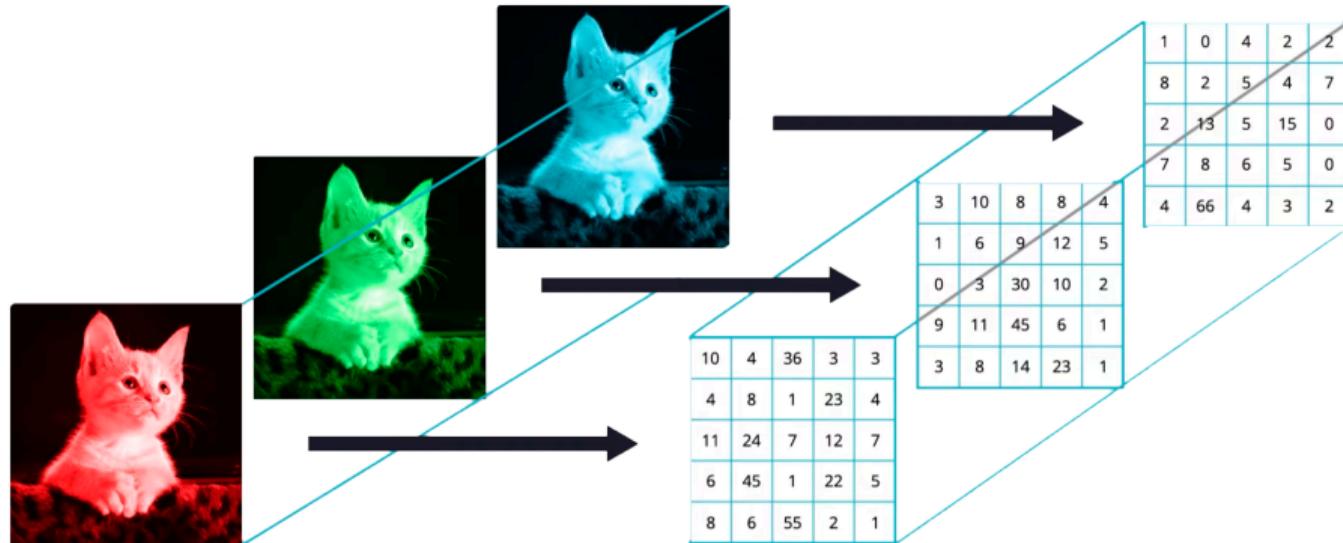


Картина – тензор

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



Картина – тензор



Картинка – тензор

5 x 5 x 3

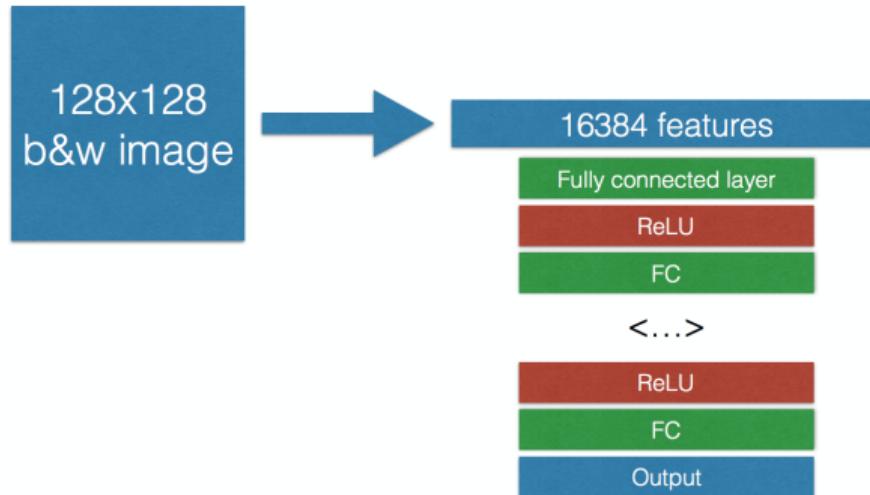


A thick black arrow pointing to the right, indicating the direction of the next section.

3D Array

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

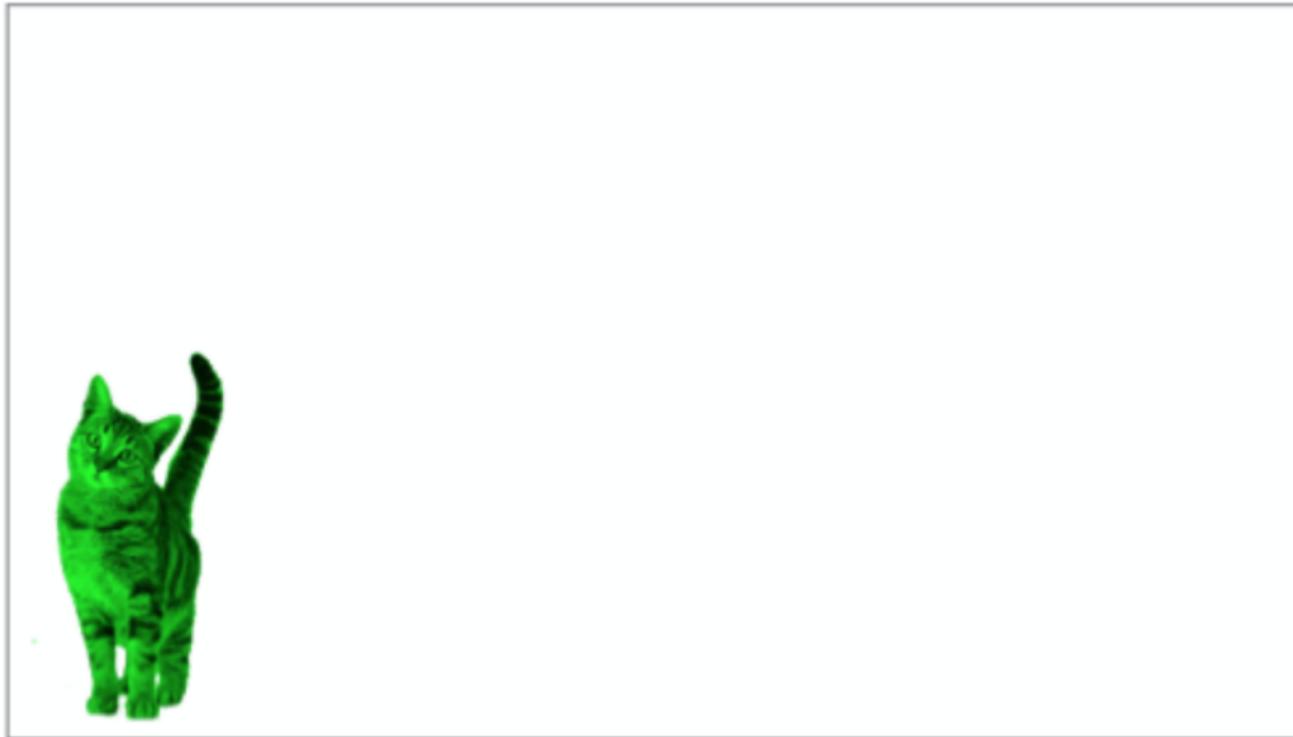
Обычная сеть



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







Обычная сетка

- Хотим, чтобы информация не терялась
- Хотим, чтобы сетка была устойчива к положению картинки
- Хотим одинаковые веса ⇒ свёртка

Свёртка



Свёртка

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	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 ₀	2 ₁	1 ₂	0
0	0 ₂	1 ₂	3 ₀	1
3	1 ₀	2 ₁	2 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	2 ₀	1 ₁	0 ₂
0	0	1 ₂	3 ₂	1 ₀
3	1	2 ₀	2 ₁	3 ₂
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 ₀	0 ₁	1 ₂	3	1
3 ₂	1 ₂	2 ₀	2	3
2 ₀	0 ₁	0 ₂	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 ₀	1 ₄	3 ₂	1
3	1 ₂	2 ₂	2 ₀	3
2	0 ₀	0 ₁	2 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	1 ₀	3 ₁	1 ₂
3	1	2 ₂	2 ₂	3 ₀
2	0	0 ₀	2 ₁	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	1	3	1
3 ₀	1 ₁	2 ₂	2	3
2 ₂	0 ₂	0 ₀	2	2
2 ₀	0 ₁	0 ₂	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 ₁	1 ₀	2 ₁	2 ₂	3
2	0 ₂	0 ₂	2 ₀	2
2	0 ₀	0 ₁	0 ₂	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 ₀	2 ₁	3 ₂
2	0	0 ₂	2 ₂	2 ₀
2	0	0 ₀	0 ₁	1 ₂

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0



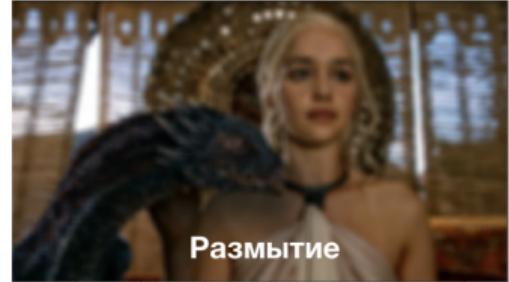
```
[[62, 36, 9], [[62, 36, 9], [[62, 36, 9],  
[62, 36, 9], [61, 35, 8], [60, 34, 7],  
[61, 35, 8], [59, 33, 6], [57, 31, 4],  
..., ..., ...,  
[58, 43, 20], [53, 41, 17], [50, 38, 14],  
[57, 45, 21], [53, 41, 17], [49, 37, 13],  
[57, 45, 21]] [52, 40, 16]] [48, 36, 12]]
```

Красный

Зеленый

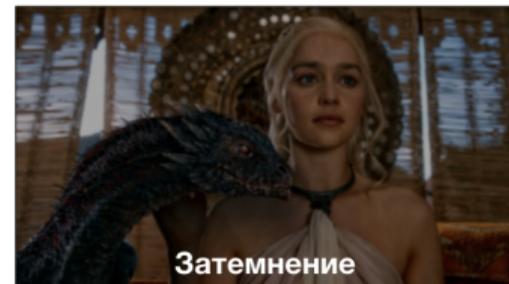
Синий

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



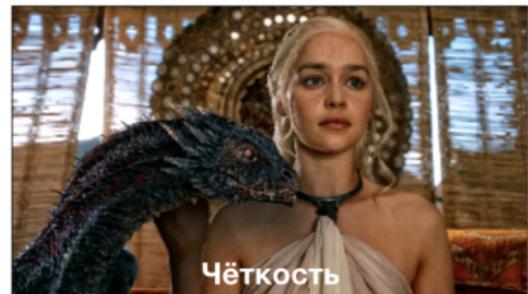
Размытие

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



Затемнение

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



Чёткость

Свёртка

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



Классификатор слэшей

Input

Kernel

Output

The diagram shows a 4x4 input image and a 2x2 kernel being multiplied to produce a 3x3 output. The input image has its bottom-right 2x2 submatrix highlighted in green. The kernel is also highlighted in green. The resulting output matrix has its bottom-right element 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



Max = 1

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

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

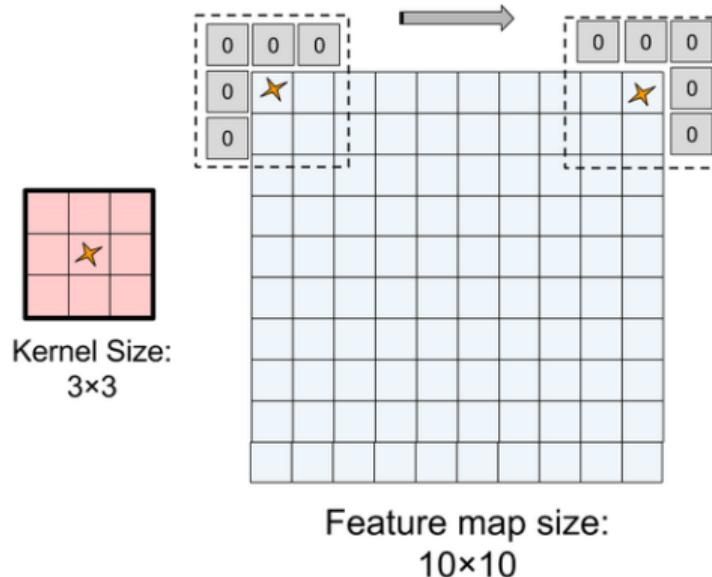
=

2	0	0
0	1	0
0	0	0

Output

Max = 2

Дополнение (Padding)



Padding используют, чтобы пространственная размерность картинки не уменьшалась после применения свёртки, это помогает не терять информацию на краях и избежать проблем с размерностями

Дополнение (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

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:

$$b$$

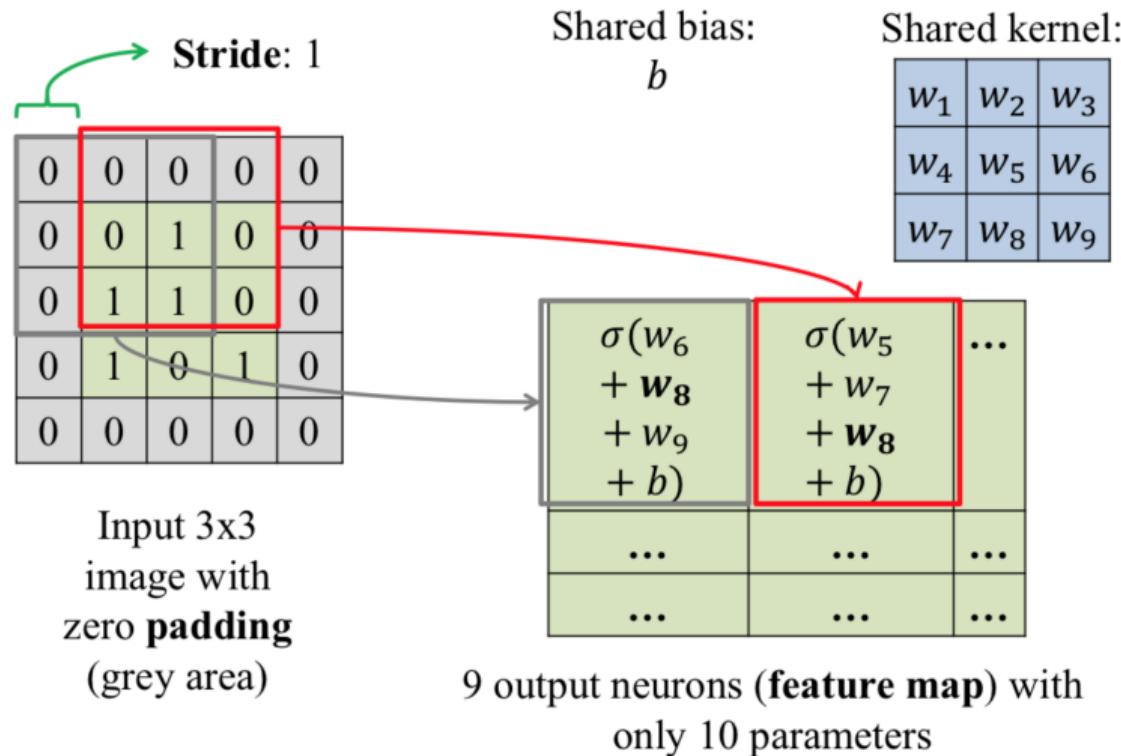
Shared kernel:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

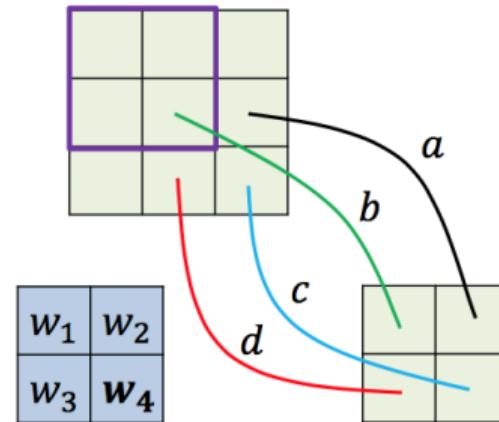
$\sigma(w_6 + w_8 + w_9 + b)$
...
...

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

Свёрточный слой



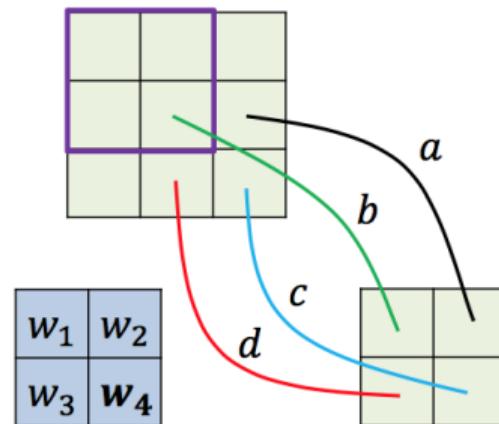
Backpropagation для свёрточного слоя



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

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

Backpropagation для свёрточного слоя

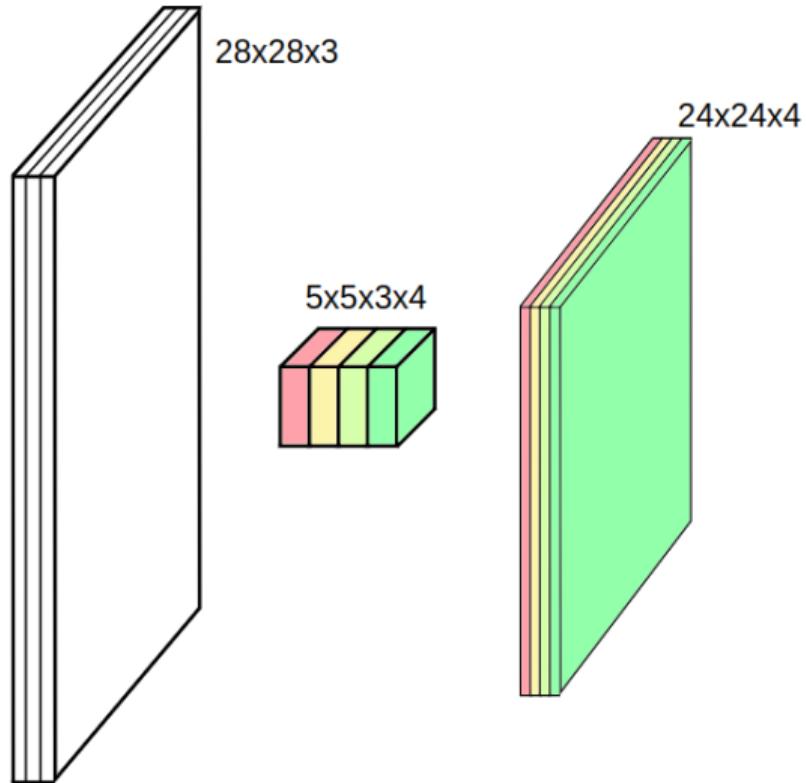


$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \cdot x_{11} + \frac{\partial L}{\partial b} \cdot x_{12} + \frac{\partial L}{\partial c} \cdot x_{22} + \frac{\partial L}{\partial d} \cdot x_{12}$$

Свёрточный слой

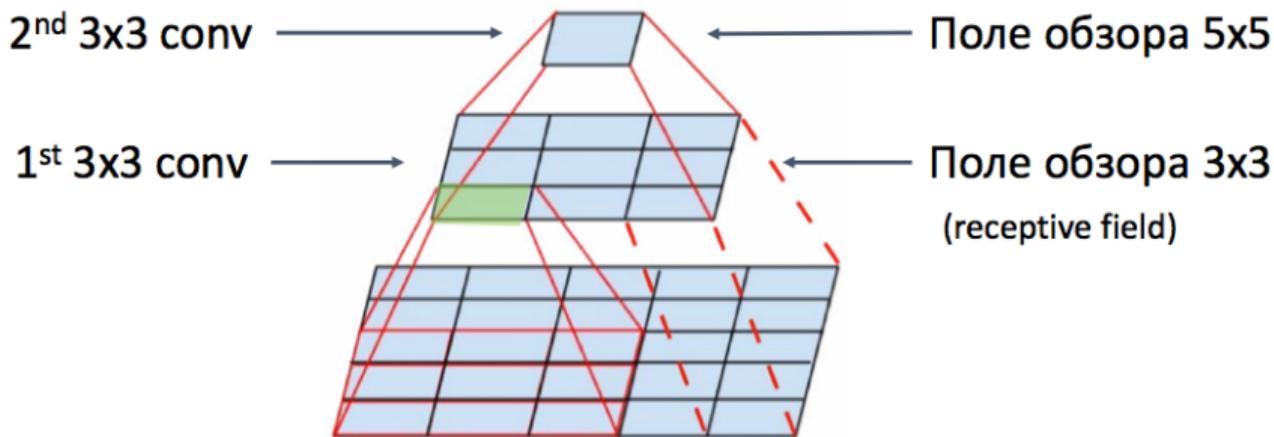
- Слой действует одинаково для каждого участка картинки, в отличие от полносвязного
- Нужно оценивать меньшее количество параметров
- Слой учитывает взаимное расположение пикселей
- Можно учить тем же самым backpropagation
- Свёрточный слой - это полносвязный слой с ограничениями, попробуйте нарисовать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

Одного ядра недостаточно



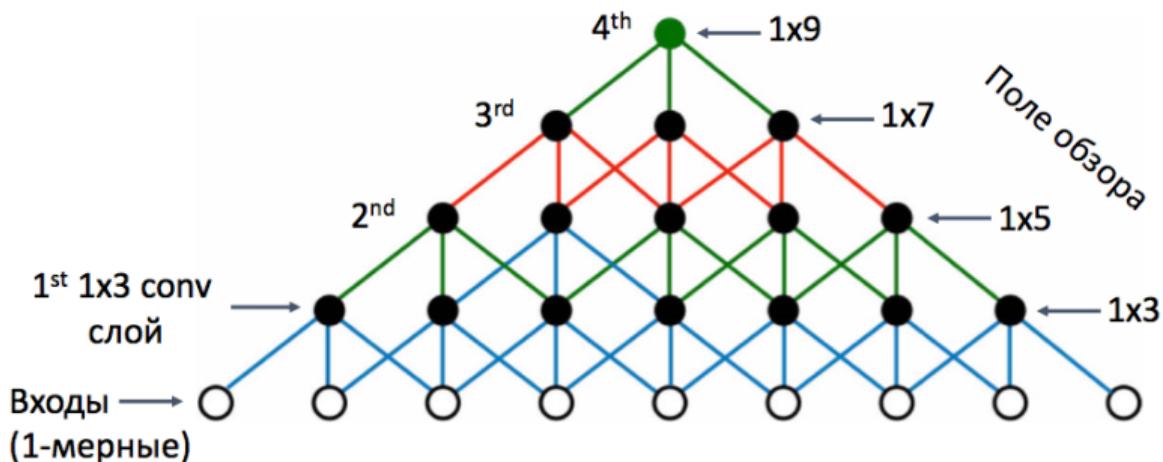
Одного свёрточного слоя недостаточно

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



Одного свёрточного слоя недостаточно

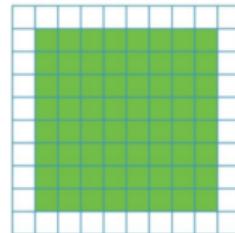
- N слоёв со свёртками 3×3
- На N -ом слое поле обзора $(2N + 1) \times (2N + 1)$
- Если наш объект размера 300, надо 150 слоёв... \Rightarrow нужно растить поле обзора быстрее



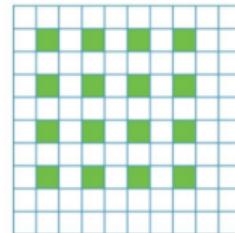
Сдвиги и Пулинг

Нужно растить поле обзора быстрее!

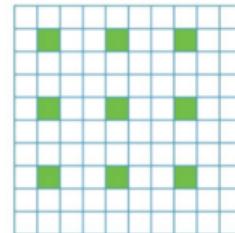
- Пиксели локально скоррелированы – соседние пиксели, как правило, не сильно отличаются друг от друга
- Если будем делать свёртку с каким-то шагом, сэкономим мощности компьютера и не потеряем в информации



Stride = 1



Stride = 2

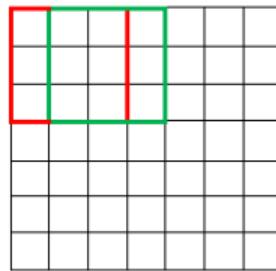


Stride = 3

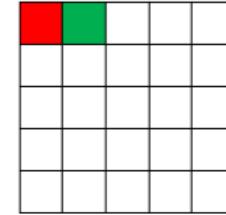
- Очень агрессивная стратегия снижения размерности картинки

Сдвиг (Stride)

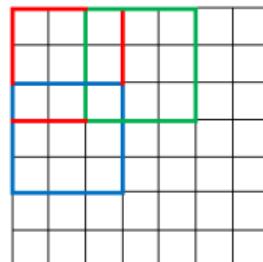
7 x 7 Input Volume



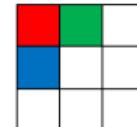
5 x 5 Output Volume



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

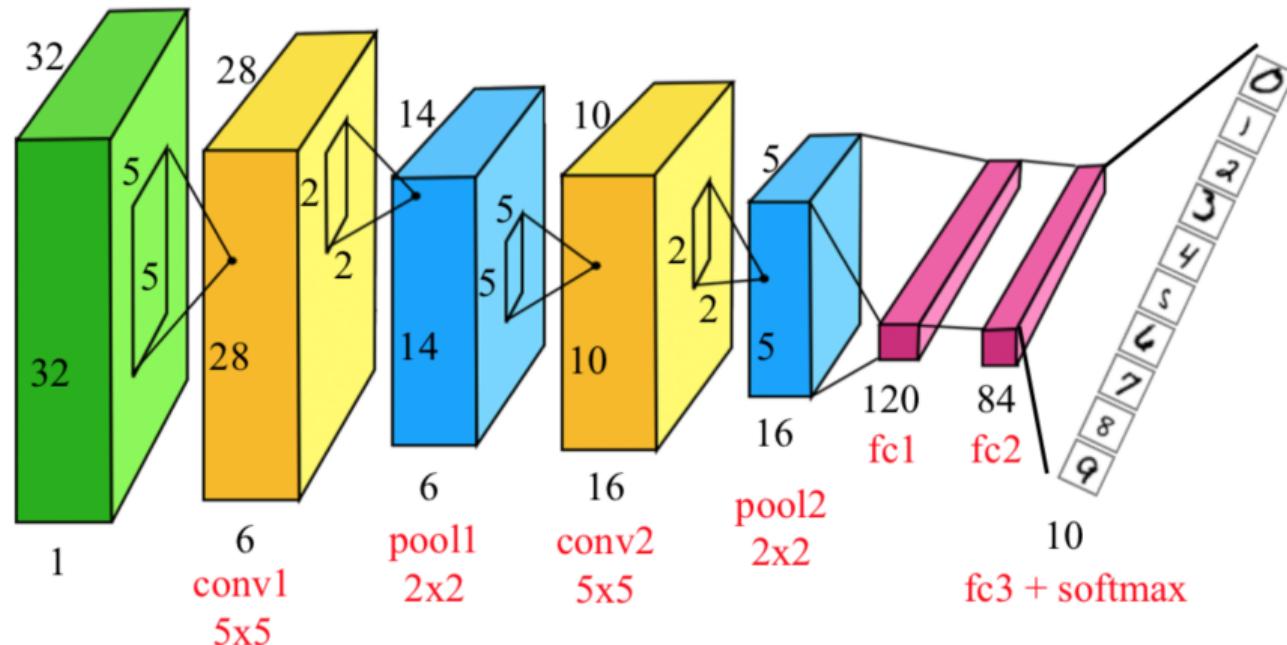
4	7	9
1	6	7
4	5	3

Average
Pooling

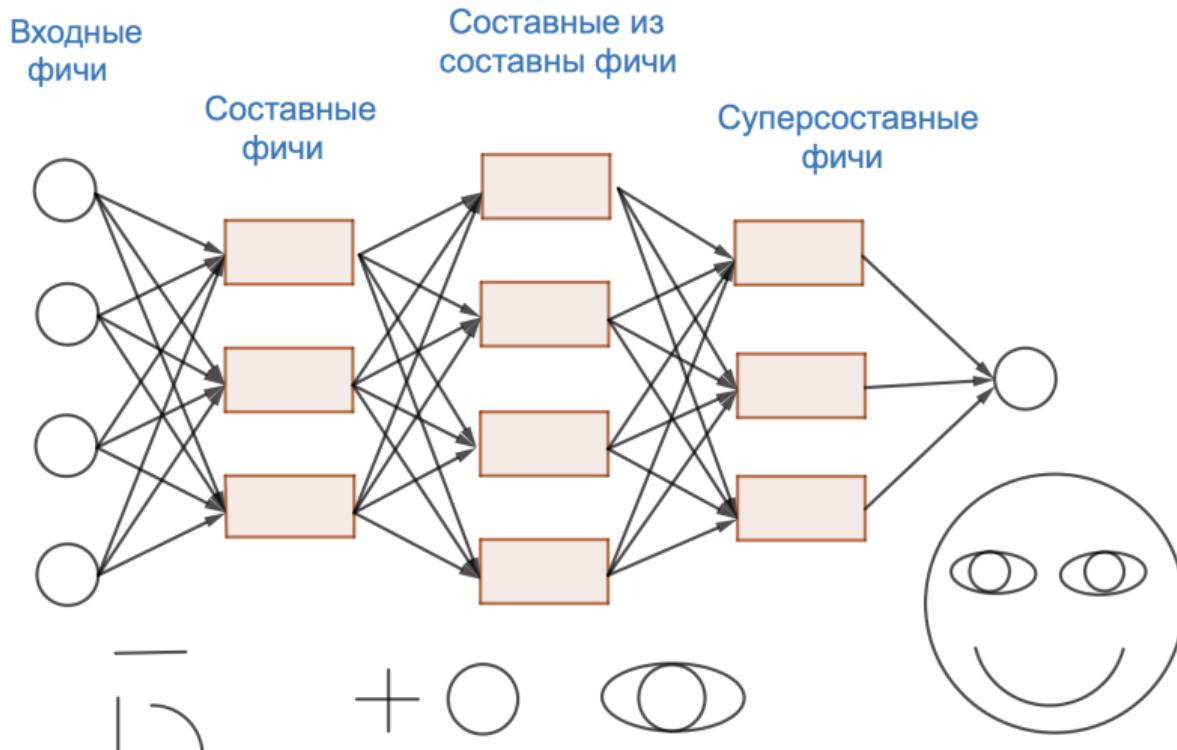
1	4	4,8
0,8	2,5	2,8
2,5	1	0,8

Простейшая CNN

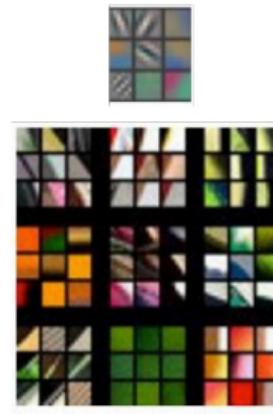
Нейросеть LeNet-5 (1998) для распознавания рукописных цифр



Что выучивают нейросети



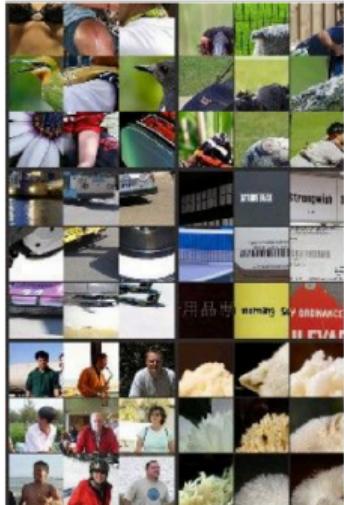
Что выучивают нейросети



Layer 1



Layer 2



Layer 3



Layer 4



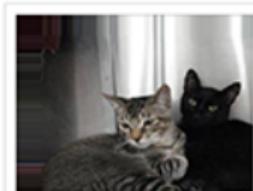
Layer 5

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

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://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>

Data augmentation

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



Делает модель более устойчивой, полезна при маленьких выборках. На больших датасетах также улучшает результаты.

Собираем свою собственную CNN