курс «Глубокое обучение»

Свёрточные нейронные сети

Александр Дьяконов

(ВМК МГУ имени М.В. Ломоносова)

2 марта 2020 года

План

Что такое изображение

Как классифицировать

Свёртка (Convolution)

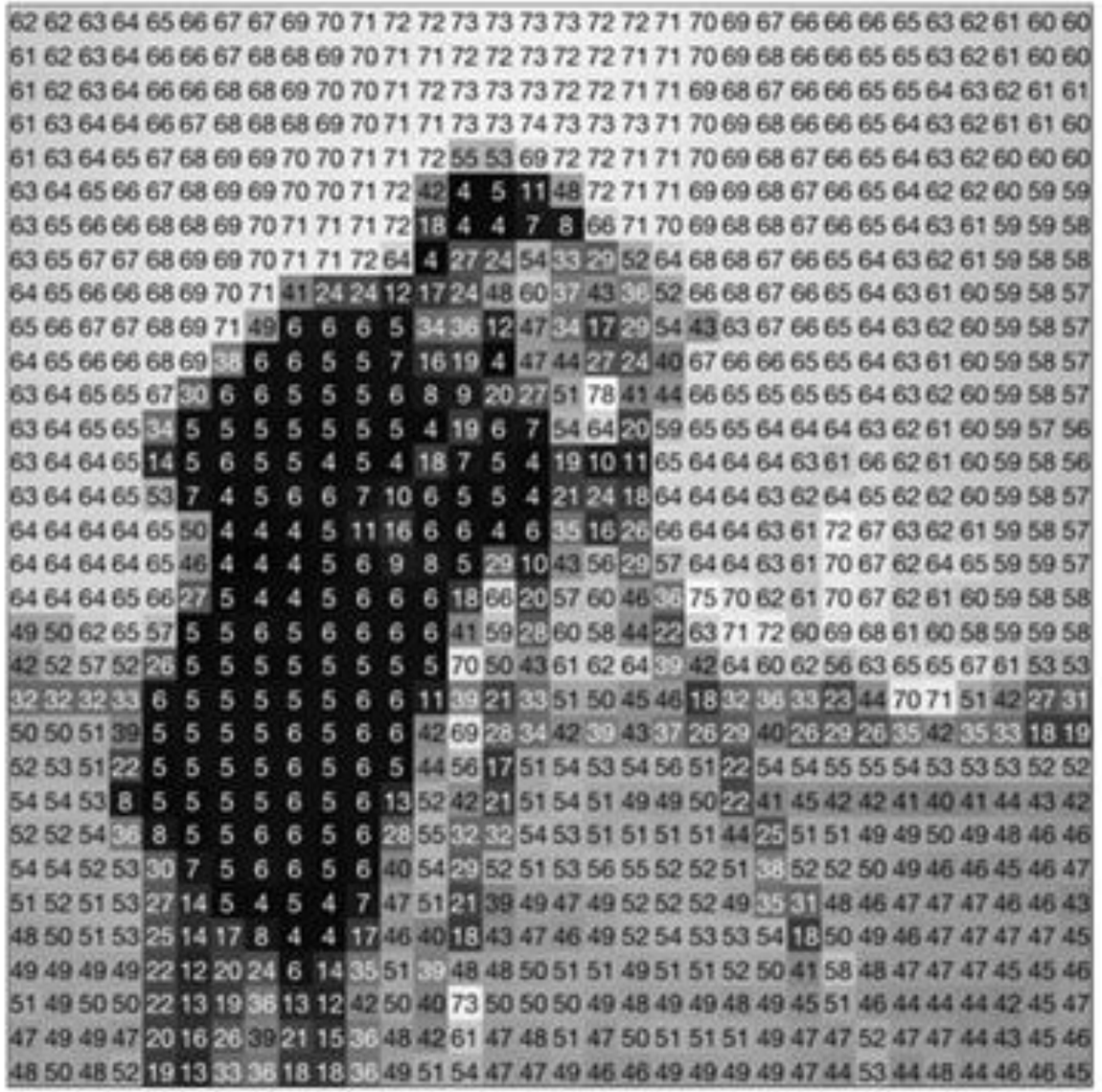
Отступ (Padding) Шаг (stride)

Свёрточные нейронные сети (ConvNet, CNN)

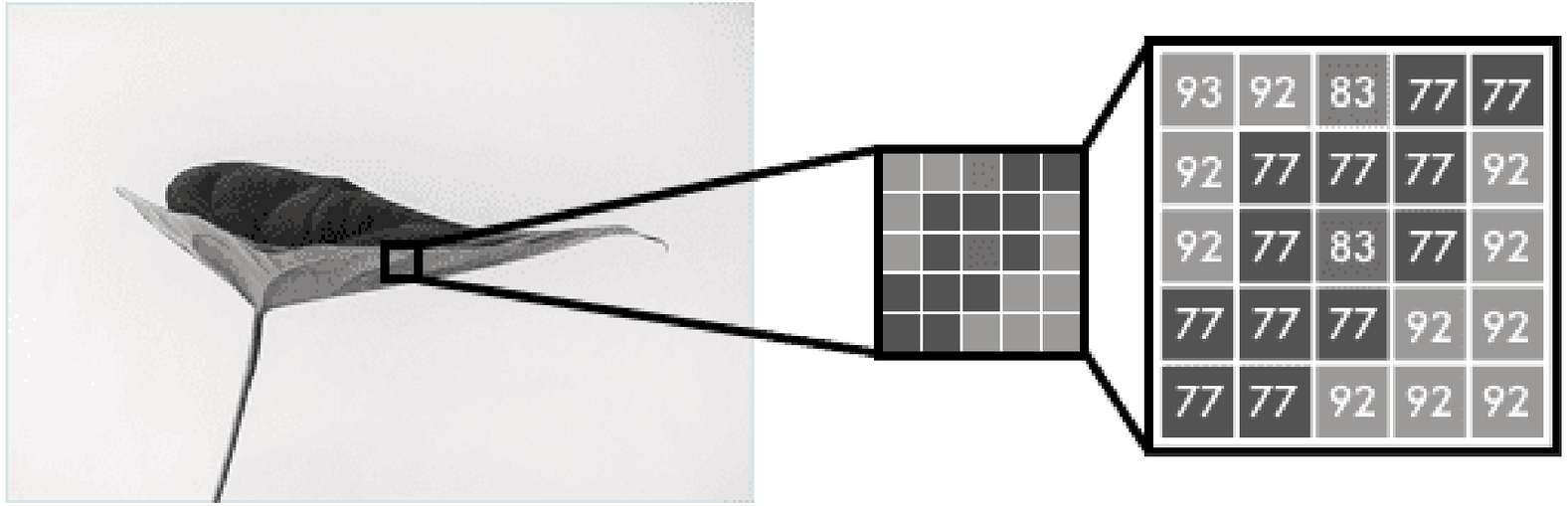
Pooling (агрегация, субдискретизация / subsampling)

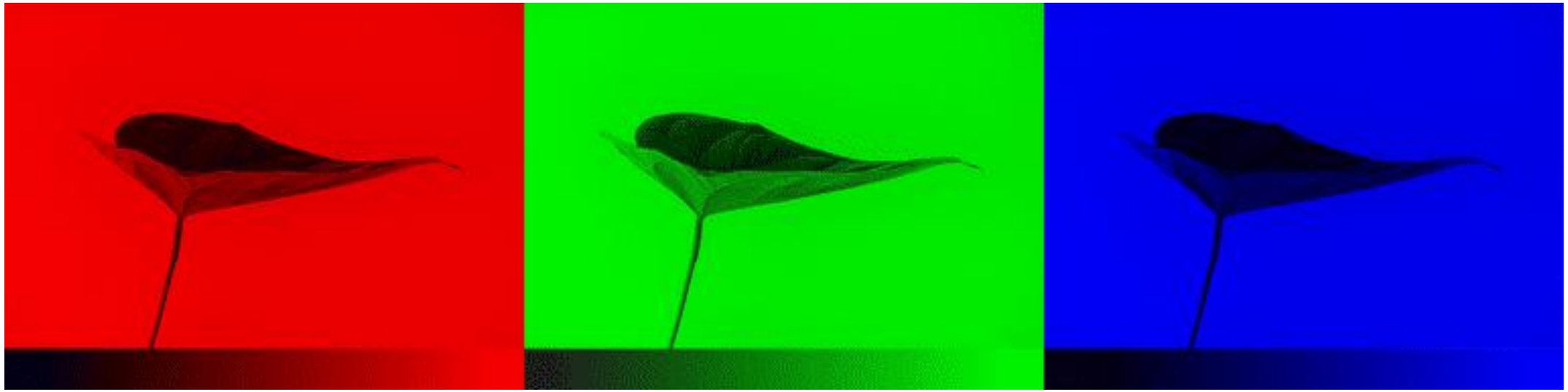
Какие бывают свёртки

Что такое изображение

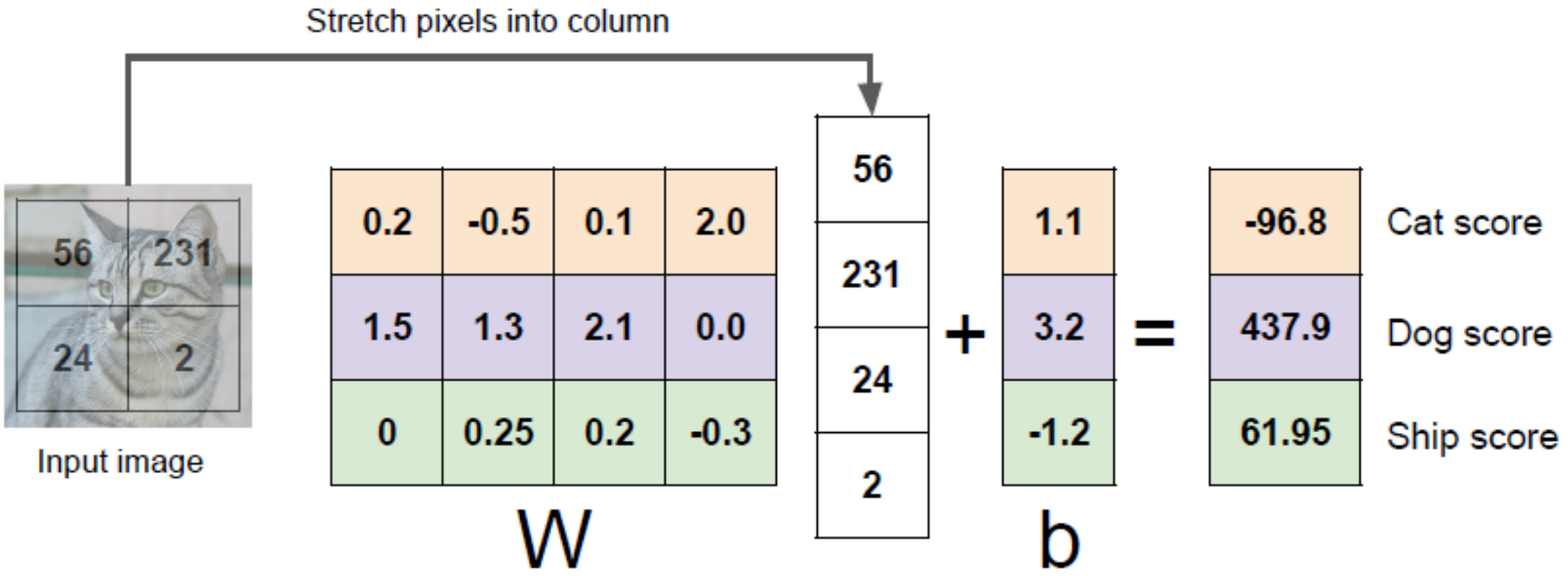


Что такое изображение





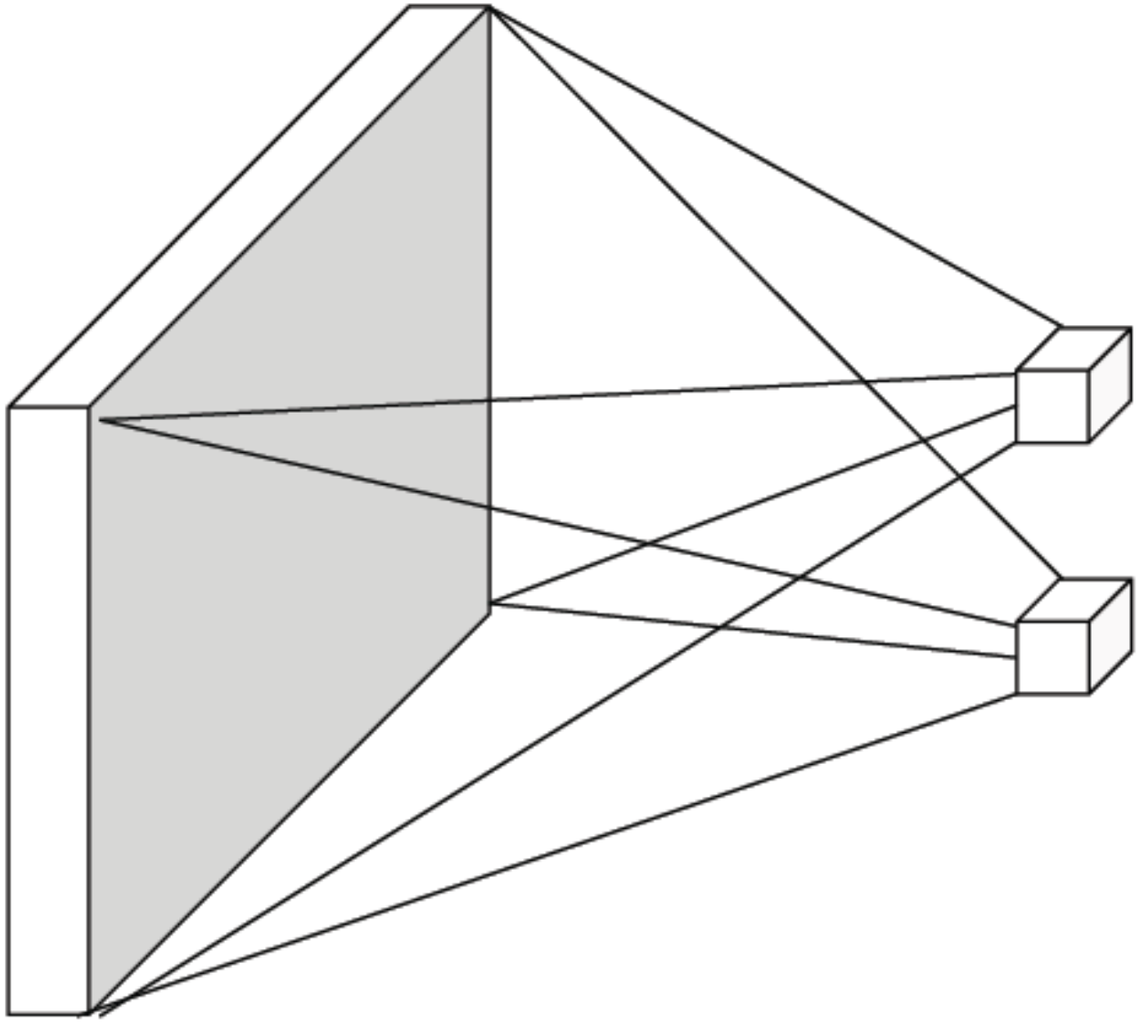
Линейный подход к классификации на несколько классов



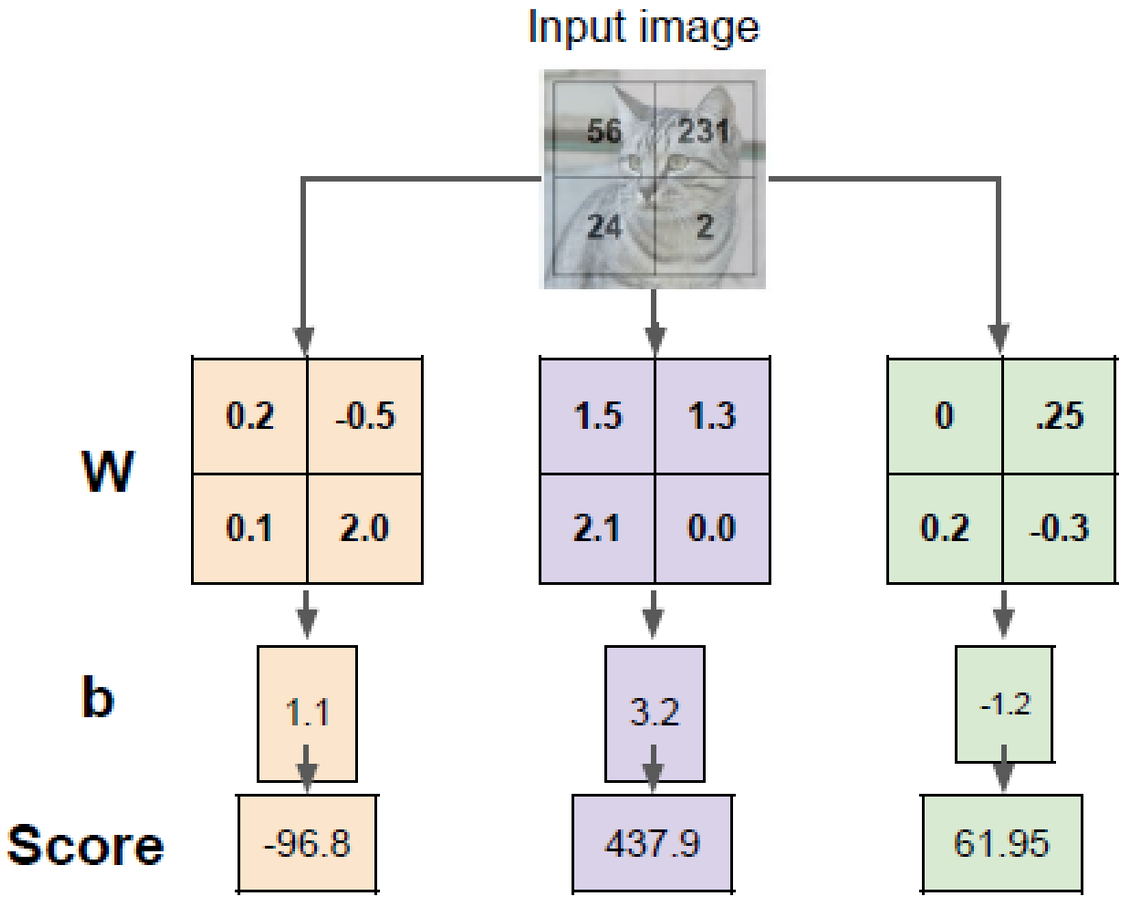
3 класса = 3 вектора признаков

Изображение = вытянуть в вектор

Линейный подход к классификации на несколько классов



Линейный подход к классификации на несколько классов



Вектора весов можно потом опять «прорешейпить» в изображения:

Проблема

Детектирование объекта в одном месте изображения
Примитивность модели

В обычных сетях слишком много параметров!

если изображение 256×256×3 ~ 200k,
то чтобы изображение → изображение
надо $3.9 \cdot 10^9$ параметров!

Как сравнивать изображения

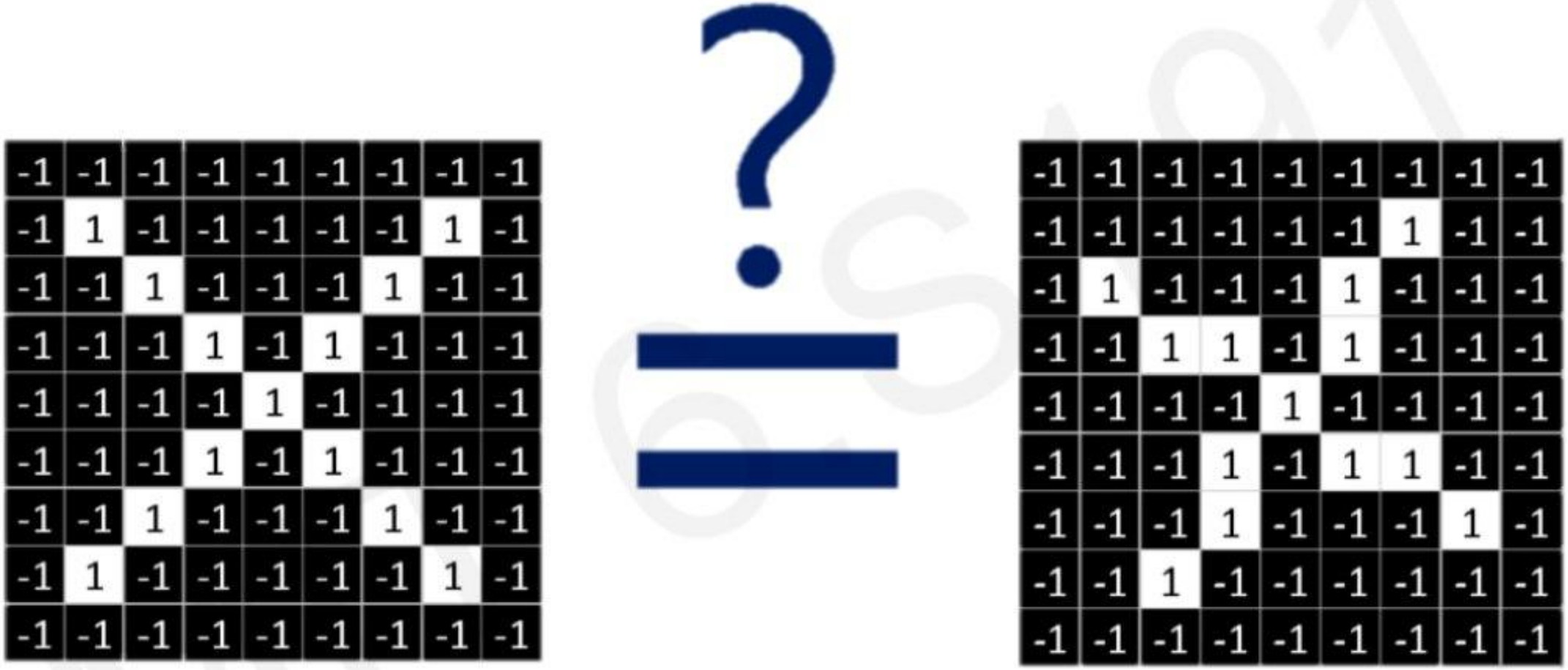
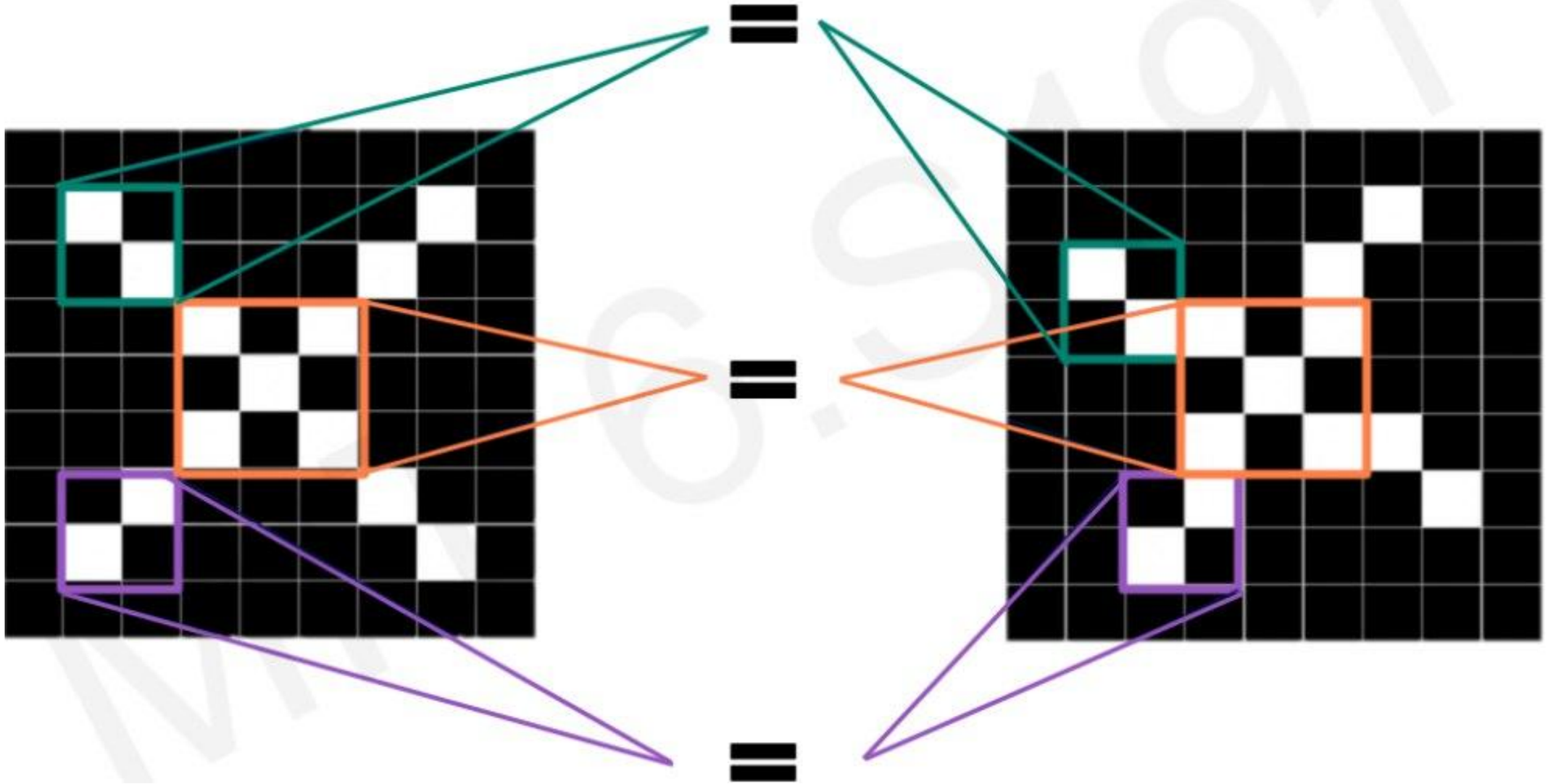


Image is represented as matrix of pixel values… and computers are literal!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

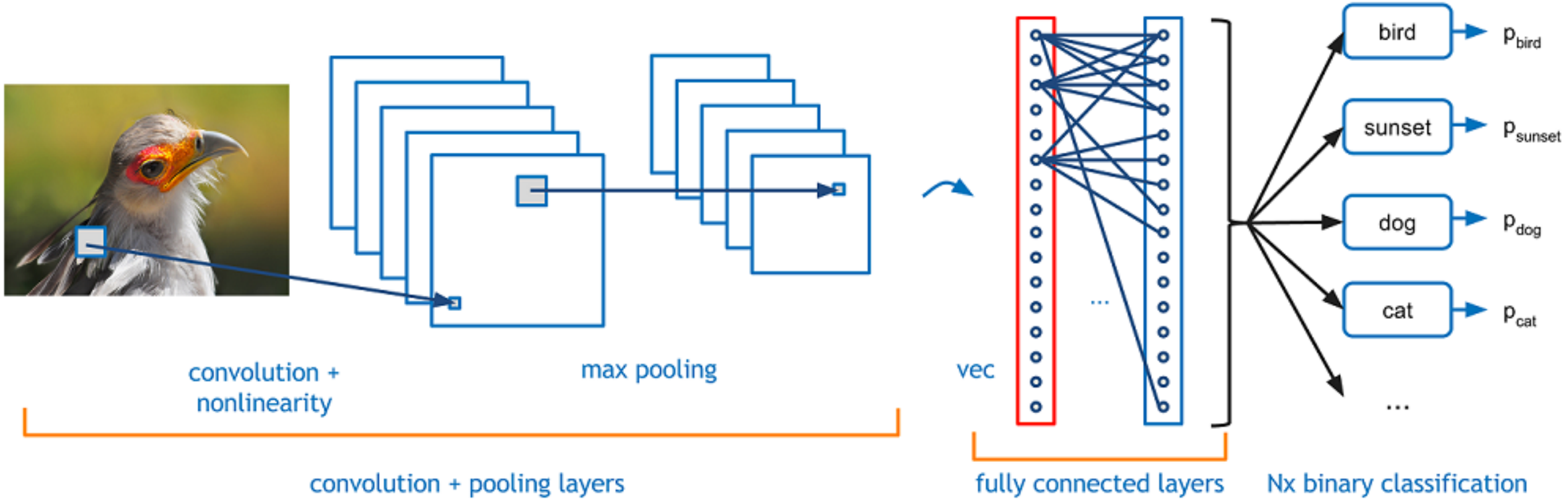
http://introtodeeplearning.com/

Как сравнивать изображения



http://introtodeeplearning.com/

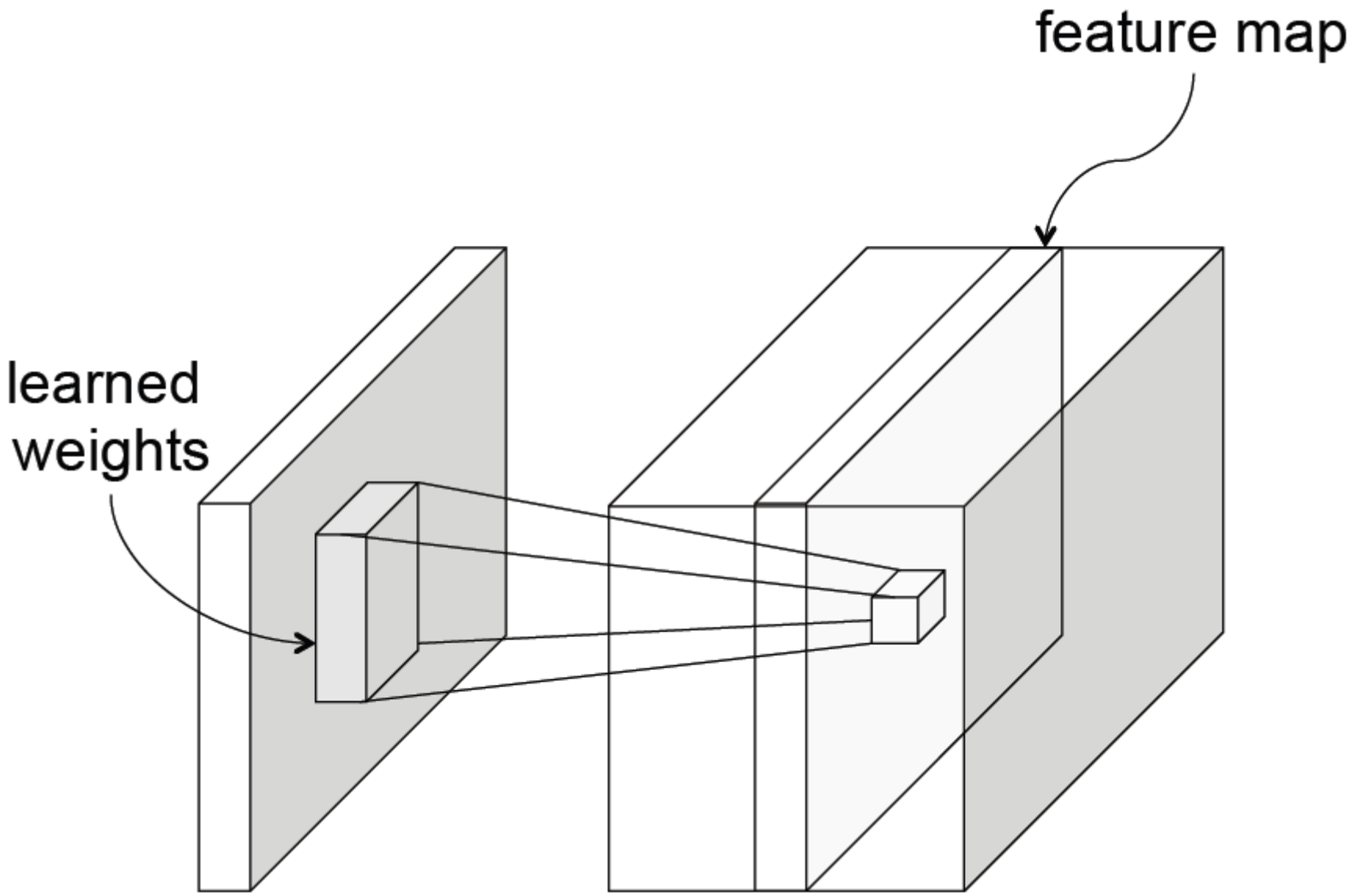
Свёрточные нейронные сети (ConvNet, CNN)



– специальный вид нейронных сетей,
для обработки равномерных сигналов

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

Свёрточные нейронные сети (ConvNet, CNN)



http://slazebni.cs.illinois.edu/spring17/

Что такое 1-D свёртка (Convolution)

пусть $I = (i_1, \ldots, i_n) \in \mathbb{R}^n$ **– сигнал / массив,** $K = (k_1, \ldots, k_r) \in \mathbb{R}^r$ **– ядро свёртки,**

тогда свёртка:

$$I * K = (i_1 k_1 + \ldots + i_r k_r,\ i_2 k_1 + \ldots + i_{r+1} k_r,\ \ldots,\ i_{n-r+1} k_1 + \ldots + i_n k_r) \in \mathbb{R}^{n-r+1}$$

-1	0	1							
1	2	3	4	5	5	4	3	2	1
	3 – 1								

	-1	0	1						
1	2	3	4	5	5	4	3	2	1
	2	4 – 2							

							-1	0	1
1	2	3	4	5	5	4	3	2	1
	2	2	2	1	-1	-2	-2	1 – 3	

Отступ (Padding)

Нулевой

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

Константный

-1	0	2	0	-1				
1	1	1	2	3	4	5	5	5
		-2	-1	0	1	2		

Зеркальный

-1	0	2	0	-1				
2	1	1	2	3	4	5	5	4
		-3	-1	0	1	3		

Циклический

-1	0	2	0	-1				
4	5	1	2	3	4	5	1	2
		-5	-5	0	5	5		

2-D свёртка (Convolution)

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{r} K_{ij} I_{x+i-1, y+j-1}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \right] =$$

$$\begin{bmatrix} 3 & 4 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{bmatrix} 4 & 5 \\ 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$$

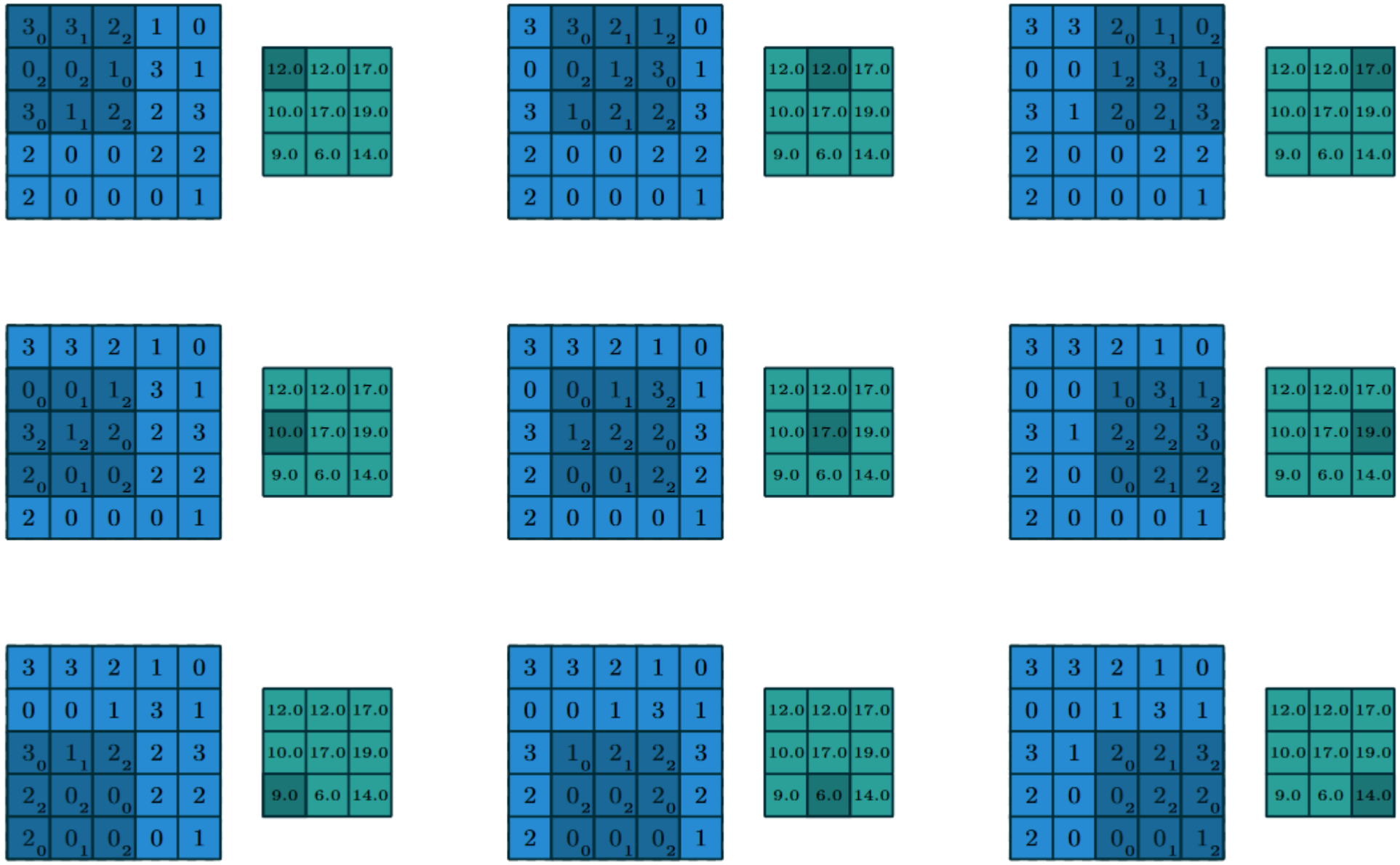
может быть немного другая индексация

хорошее объяснение:

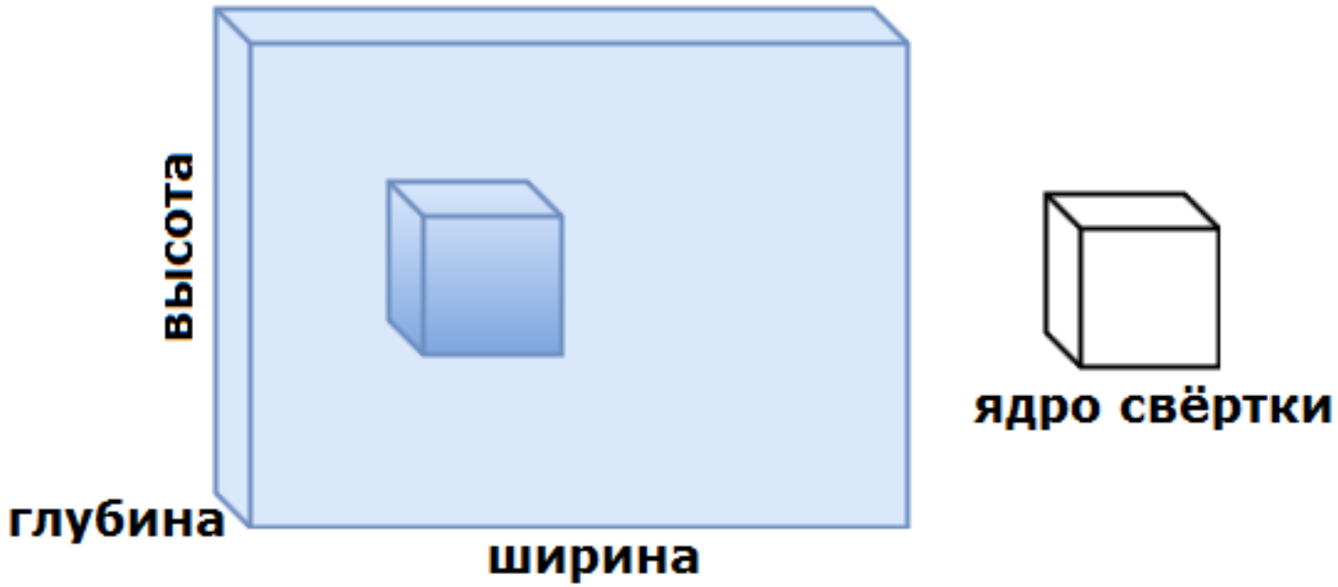
Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning 2018

https://arxiv.org/pdf/1603.07285.pdf

Свёртка (Convolution)



Свёртка (Convolution)



Глубина (depth) / число каналов

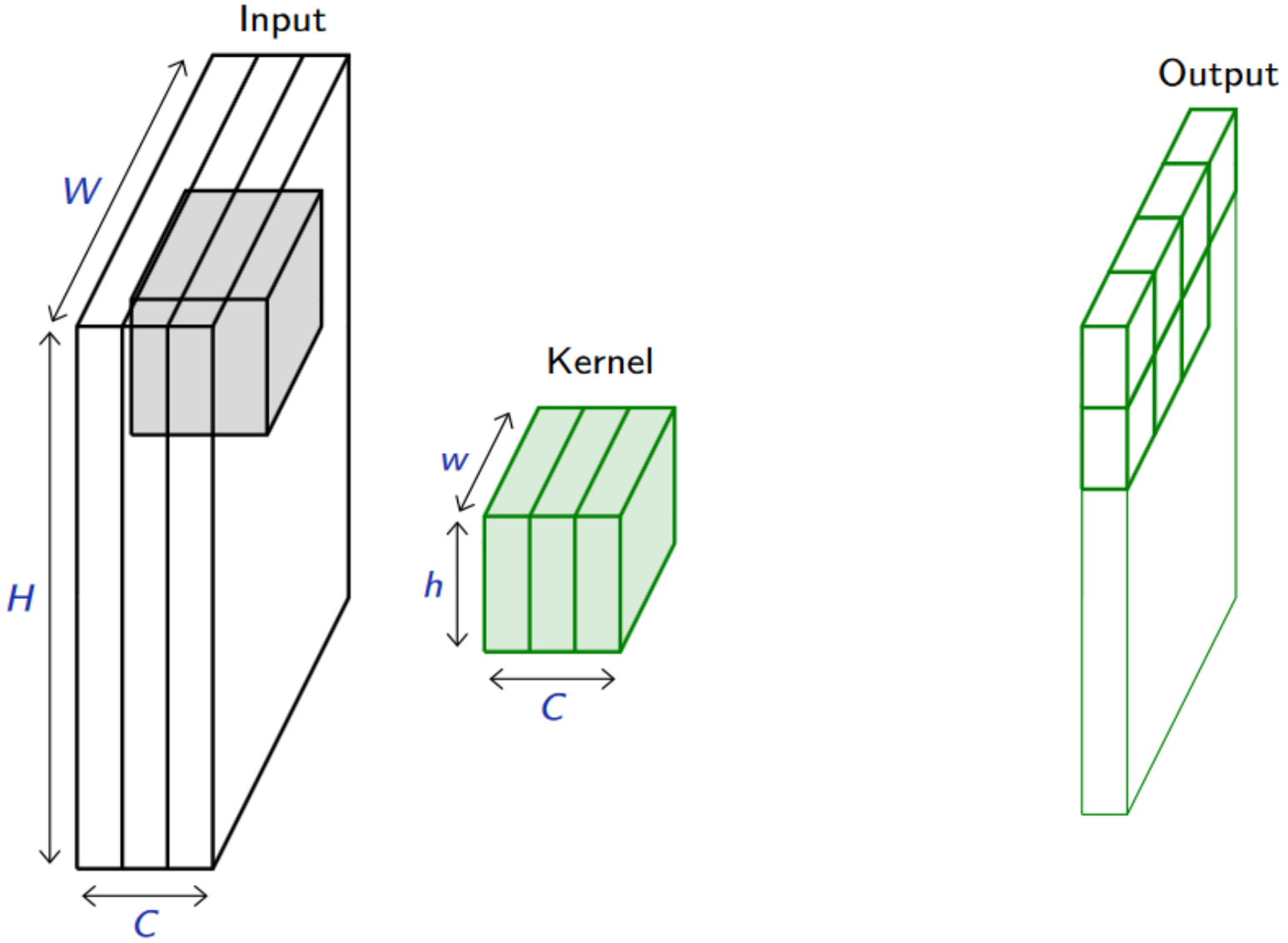
Высота (height) и ширина (width) каждого ядра

Шаг (stride) — на сколько смещается ядро при вычислении свёрток (чем больше, тем меньше размер итогового изображения)

Отступ (padding) – для дополнения изображения нулями по краям

Ядро (kernel) или фильтр (filter) – размерность как у предыдущего тензора; в 3D длина и ширина меньше (глубина совпадает)

Свёртка (Convolution)



Что делает свёртка?

https://algotravelling.com/ru/%d0%bc%d0%b0%d1%88%d0%b8%d0%bd%d0%bd%d0%be%d0%b5-%d0%be%d0%b1%d1%83%d1%87%d0%b5%d0%bd%d0%b8%d0%b5/

Что делает свёртка?

Что делает свёртка?

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, color

image = color.rgb2gray(data.chelsea())

gx = np.empty(image.shape, dtype=np.double)
gx[:, 0] = 0
gx[:, -1] = 0
gx[:, 1:-1] = image[:, :-2] - image[:, 2:]


gy = np.empty(image.shape, dtype=np.double)
gy[0, :] = 0
gy[-1, :] = 0
gy[1:-1, :] = image[:-2, :] - image[2:, :]
```

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(7, 4),
sharex=True, sharey=True)
ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
# ax1.set_title('Original image')

ax2.axis('off')
ax2.imshow(gx, cmap=plt.cm.gray)
# ax2.set_title('Horizontal gradients')

ax3.axis('off')
ax3.imshow(gy, cmap=plt.cm.gray)
plt.tight_layout(pad=0.0, h_pad=0, w_pad=0)
# ax3.set_title('Vertical gradients')
```

Свёртка (Convolution): мотивация

Раньше: обработка изображений – специально построенные свёртки

edges, corners, colors, shapes…

Сейчас: не будем специально строить свёртки – их параметры настроятся сами!

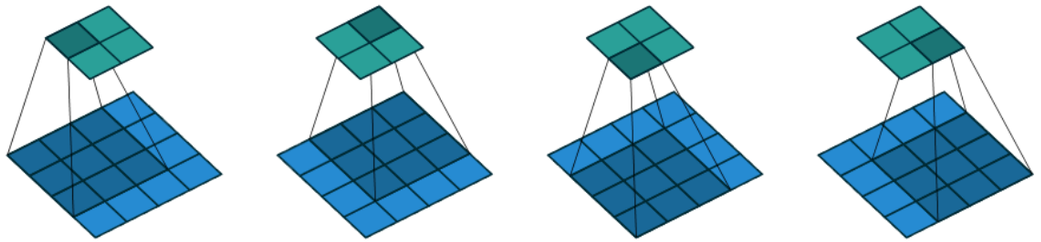
Важно: свёртку можно применять к изображениям любых размеров!

Нет ограничений на размеры входа…

Отступ (padding) – чтобы сохранялись размеры изображения

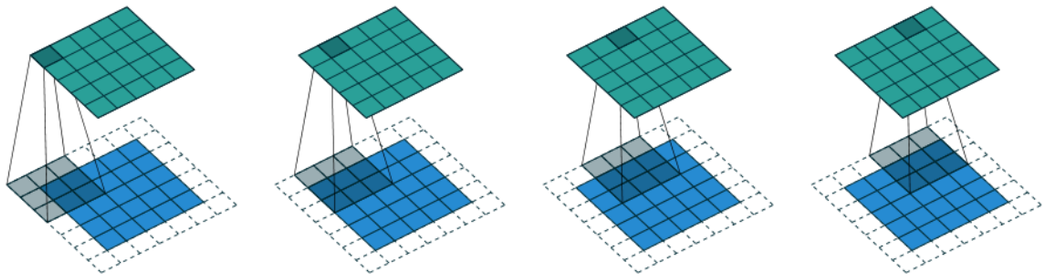
padding = VALID (в TF)

$$\begin{bmatrix}1 & 2 & 3\\ 3 & 4 & 5\\ 1 & 0 & 2\end{bmatrix} * \begin{bmatrix}0 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}1 & 2\\ 5 & 3\end{bmatrix}$$



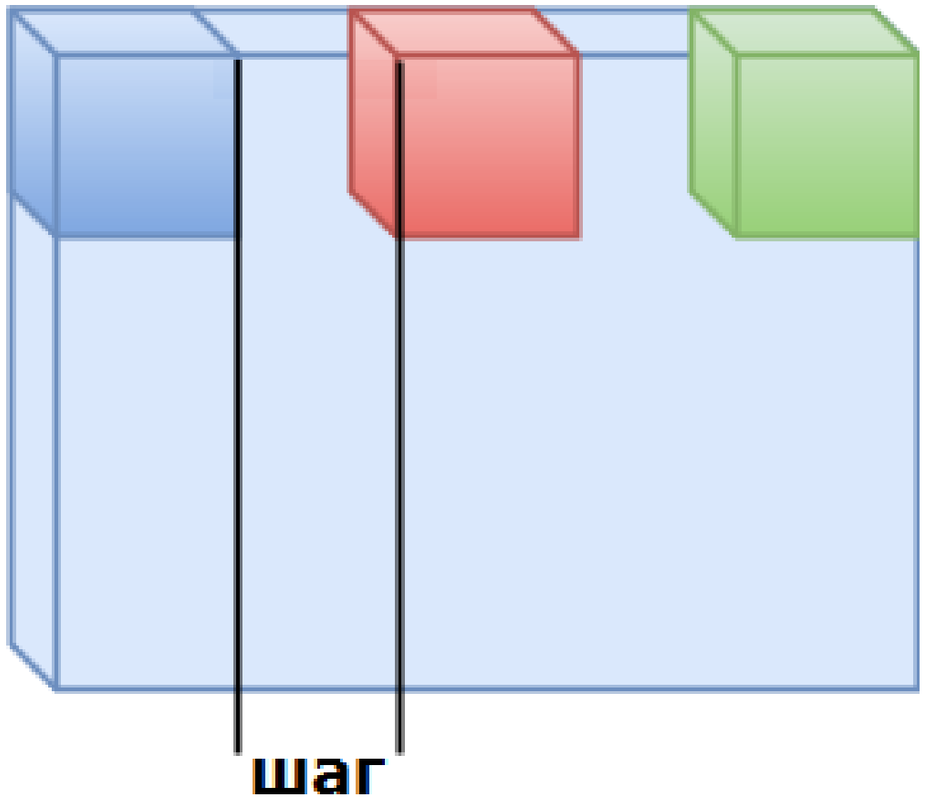
padding = SAME (в TF)

$$\begin{bmatrix}1 & 2 & 3\\ 3 & 4 & 5\\ 1 & 0 & 2\end{bmatrix} * \begin{bmatrix}0 & 1\\ 1 & -1\end{bmatrix} \equiv \begin{bmatrix}1 & 2 & 3 & 0\\ 3 & 4 & 5 & 0\\ 1 & 0 & 2 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} * \begin{bmatrix}0 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}1 & 2 & 5\\ 5 & 3 & 4\\ 0 & 2 & 0\end{bmatrix}$$



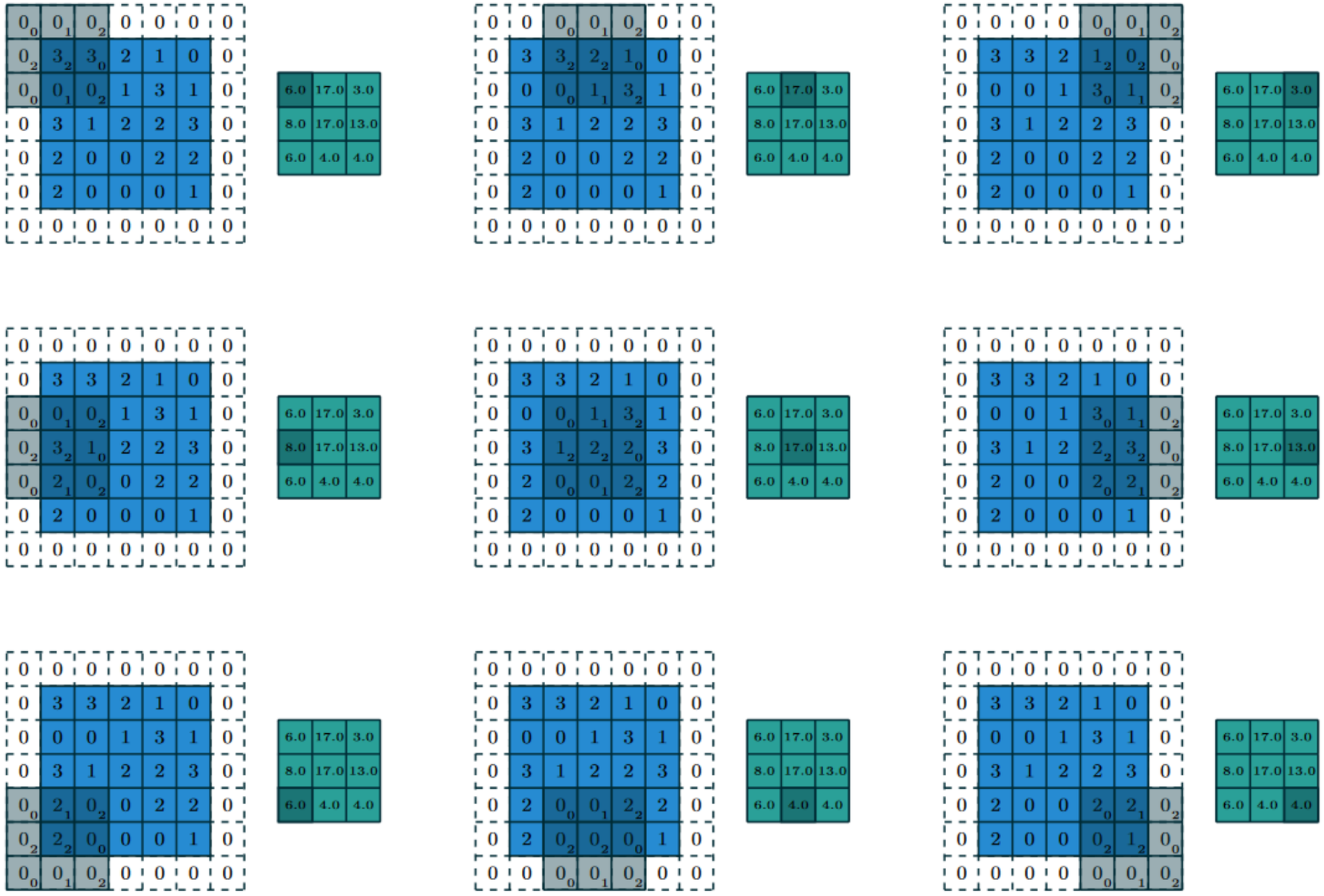
Шаг (stride)

смещаемся при вычислении свёртки (можно в каждой её размерности)



с шагом 2

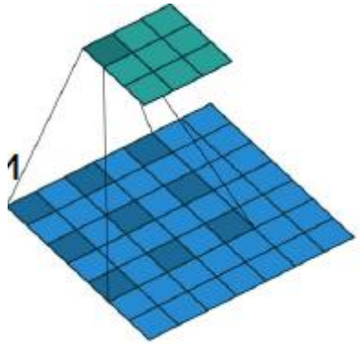
$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 \\ 3 & 4 & 5 & 2 & 1 \\ 1 & 0 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 & 2 \\ 1 & 2 & 4 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -5 & 2 \end{bmatrix}$$


6.0 17.0 3.0
8.0 17.0 13.0
6.0 4.0 4.0

Свёртка (Convolution)

Параметры (torch.nn.Conv2d):

- **Количество каналов на входе и выходе**
- **Размеры ядра**
- **Смещение (stride) – можно понижать разрешение**
- **Padding**
- **Dilation – увеличить область зависимости**
- **Размер выхода сети**



Реализация свёртки

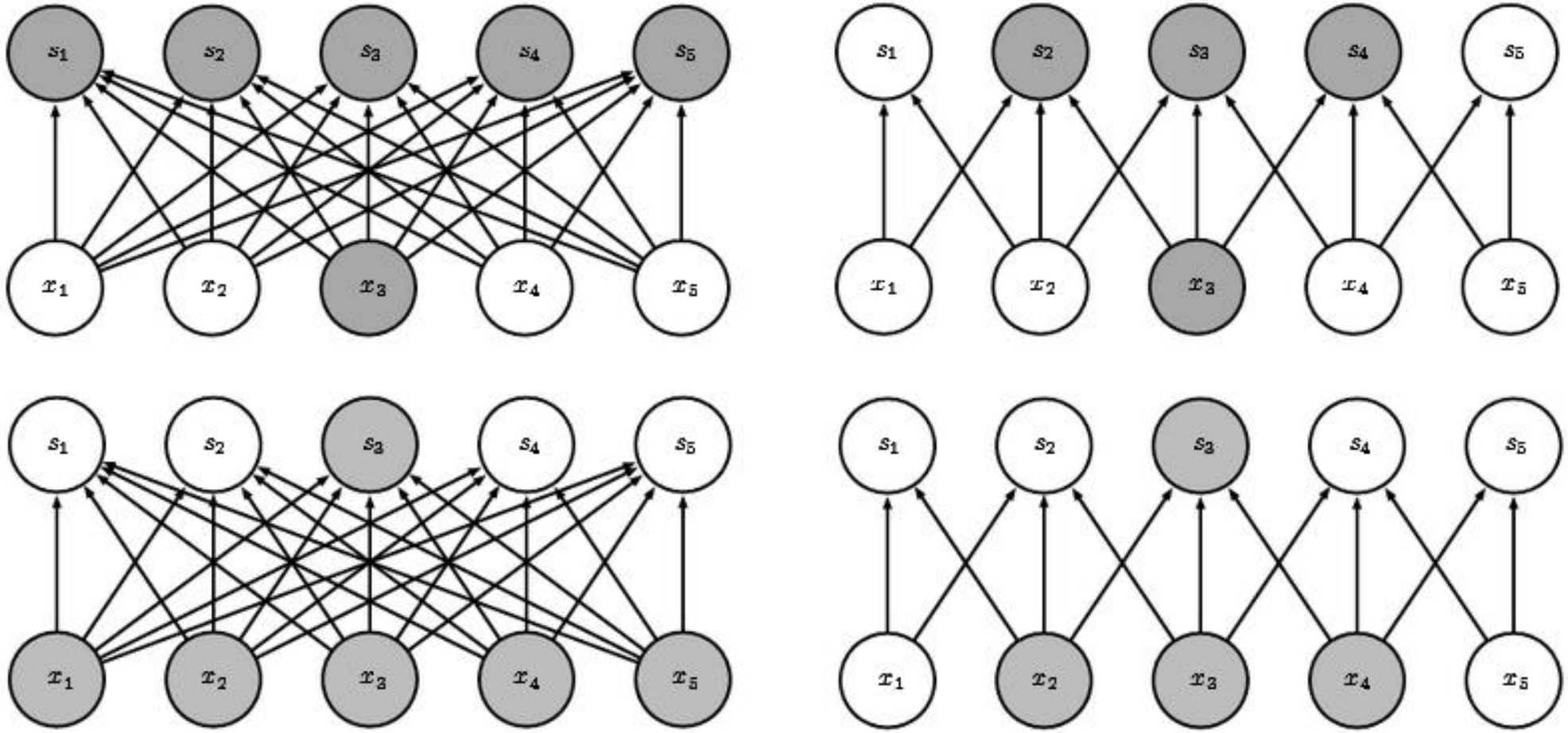
Это линейная операция!

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} * \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 \\ 0 & 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} \end{pmatrix} \cdot \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} k_{11}x_{11} + k_{12}x_{21} + k_{21}x_{21} + k_{22}x_{22} \\ k_{11}x_{21} + k_{12}x_{31} + k_{21}x_{22} + k_{22}x_{23} \\ k_{11}x_{31} + k_{12}x_{21} + k_{21}x_{31} + k_{22}x_{32} \\ k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33} \end{pmatrix}$$

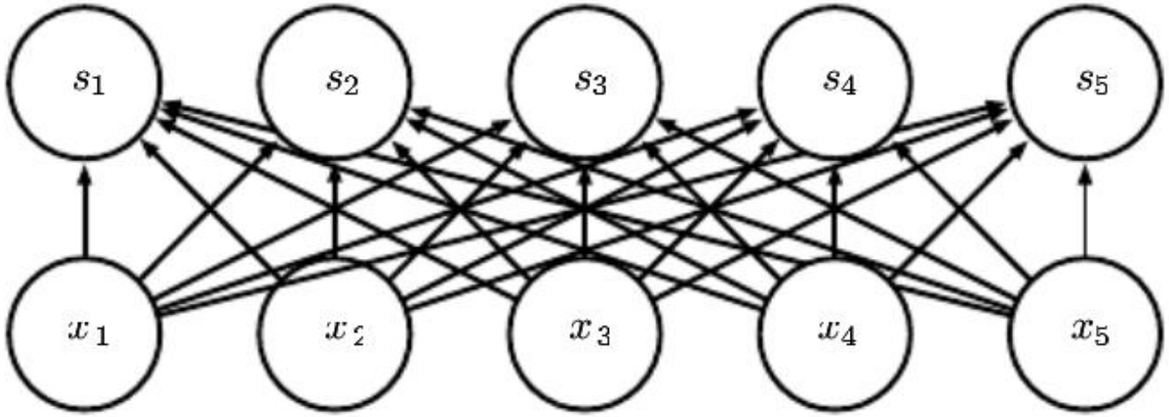
Поэтому надо быстро делать матричные перемножения…

Разреженные взаимодействия (sparse interactions)



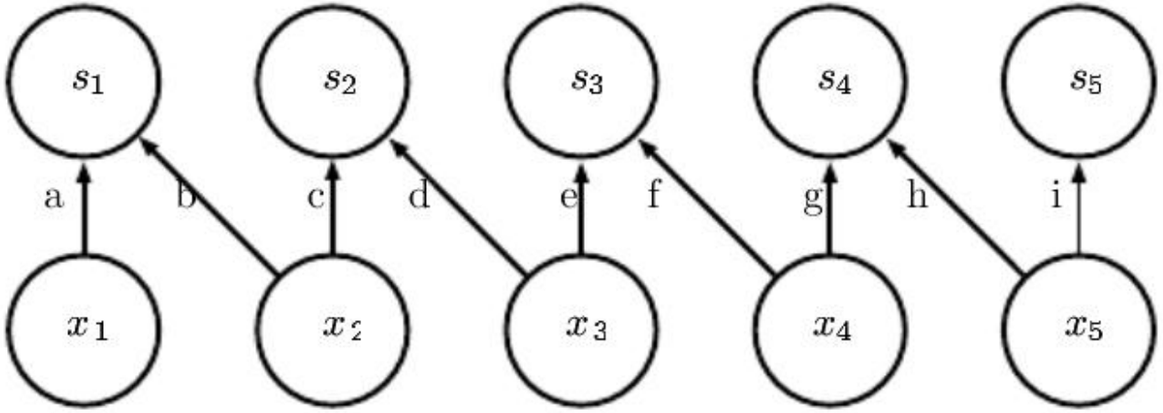
http://www.deeplearningbook.org/contents/convnets.html

Полная связность (full connections)

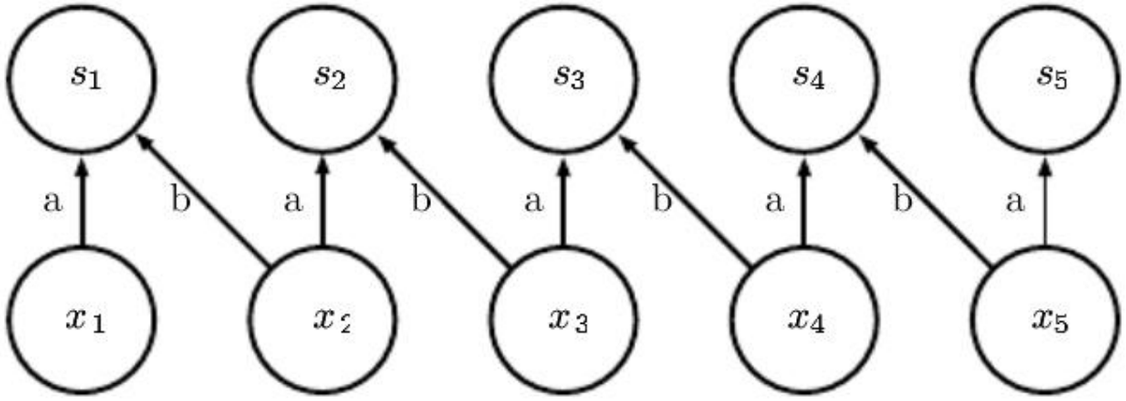


Локальная связность (local connections)

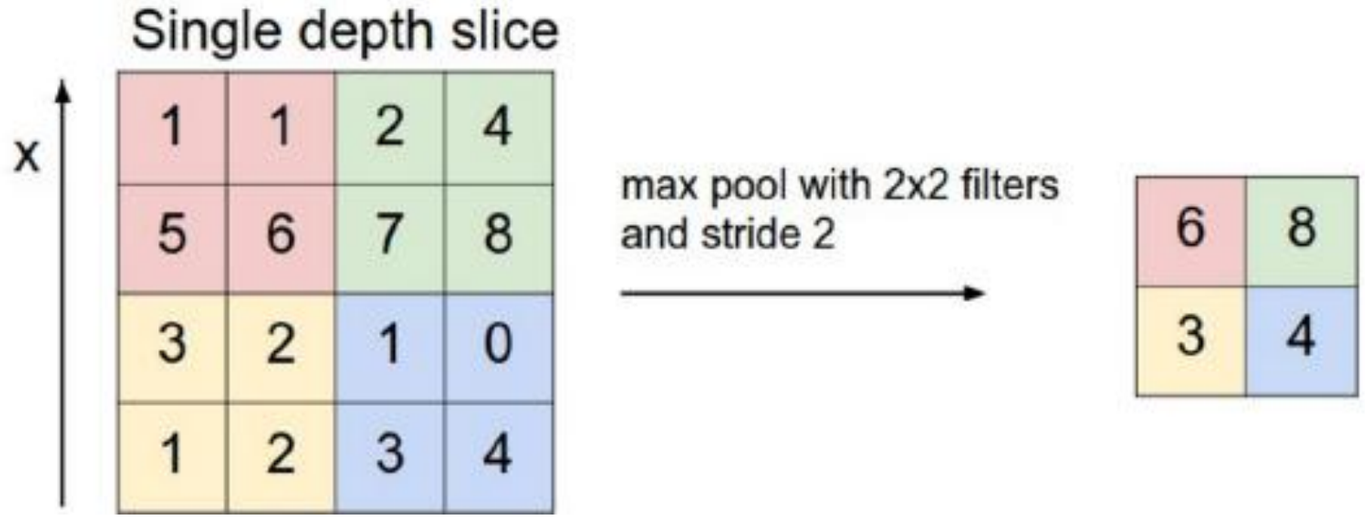
неразделяемая свёртка (unshared convolution)

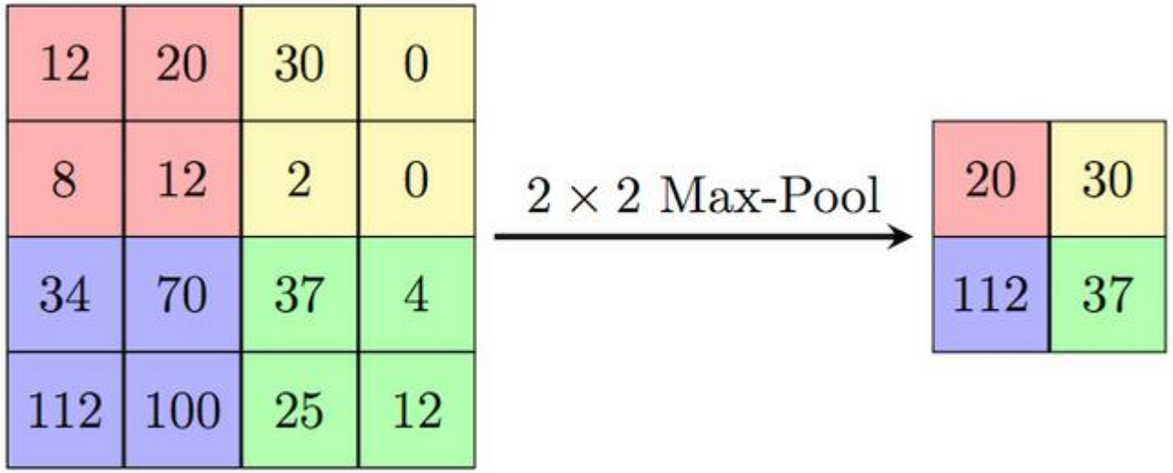


Свёртка (convolution)

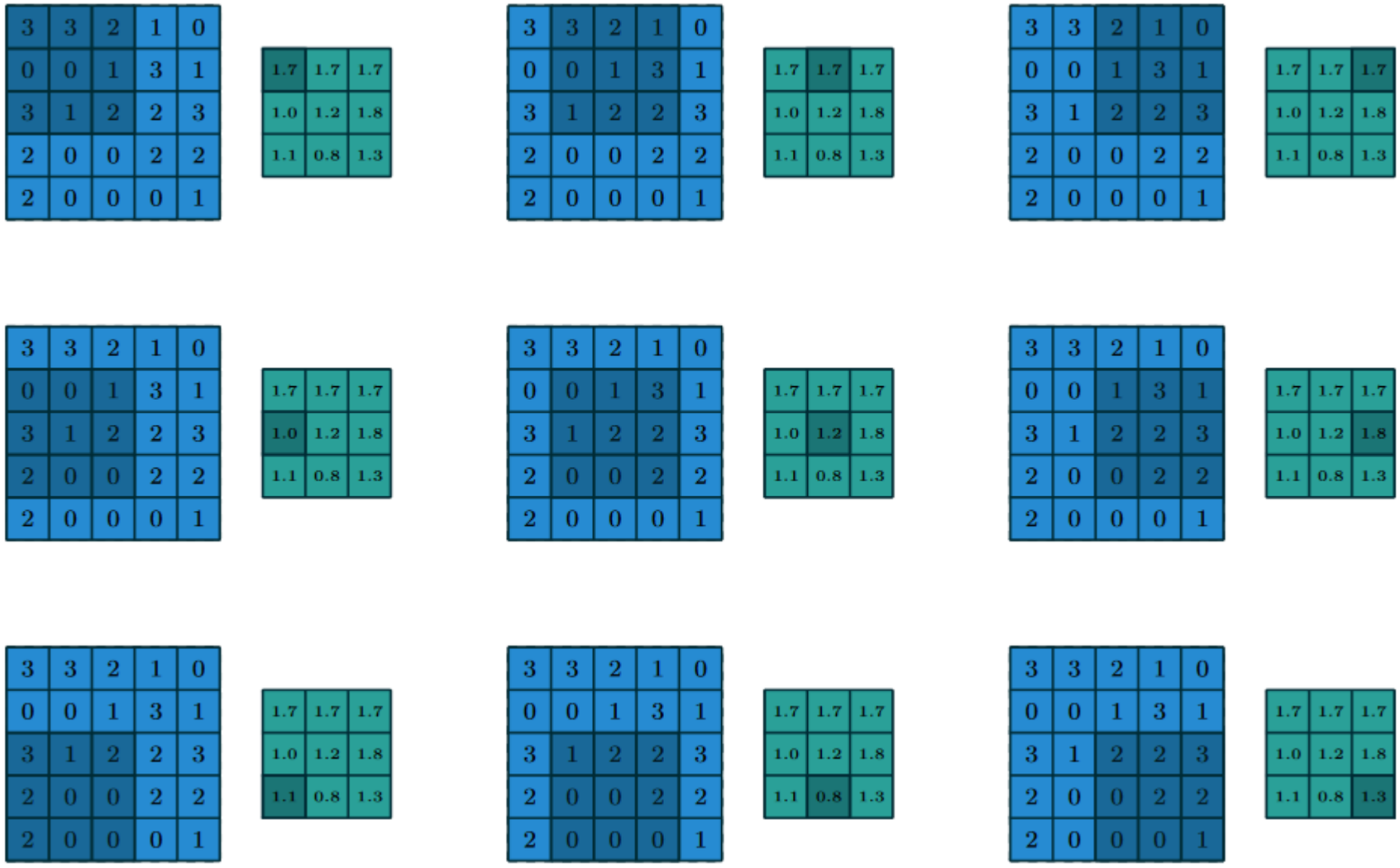


Pooling (агрегация, субдискретизация / subsampling)



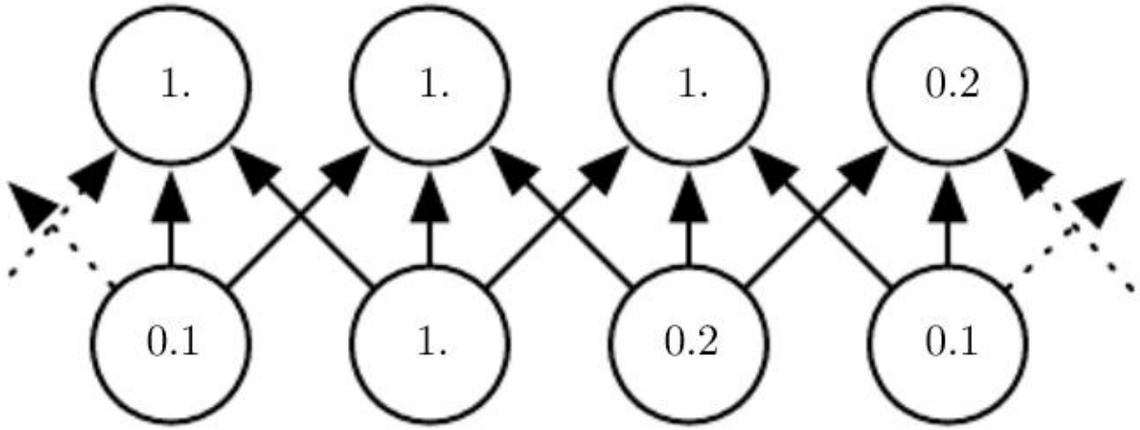


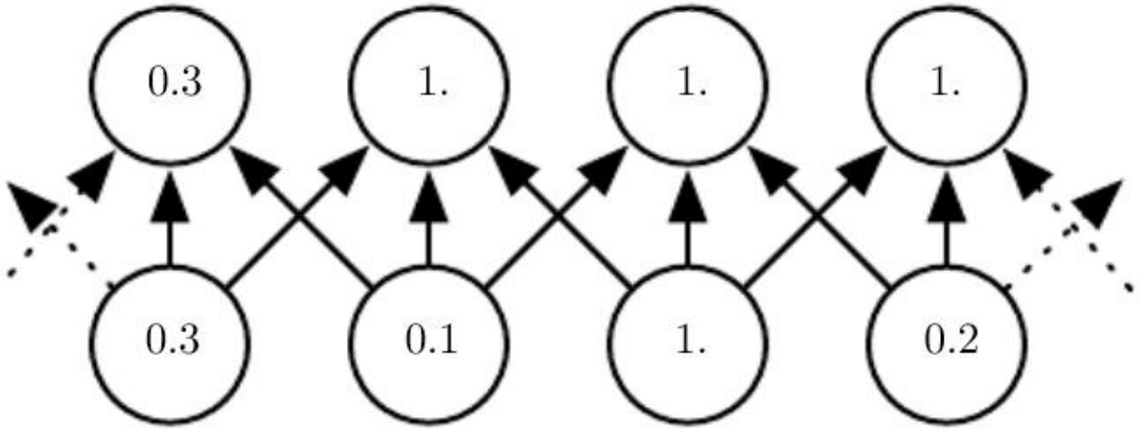
Агрегация (Pooling) усреднением



Агрегация (Pooling)

Max-pooling – инвариантность к небольшим сдвигам





Аналог голосования…

Если надо найти кошку, то в определённой окрестности

⇒ опросить соседей, есть ли кошка

Агрегация (Pooling): виды

Есть разные виды пулинга:

- усреднение
- усреднение с весами
- L2-норма
- **Stochastic Pooling**

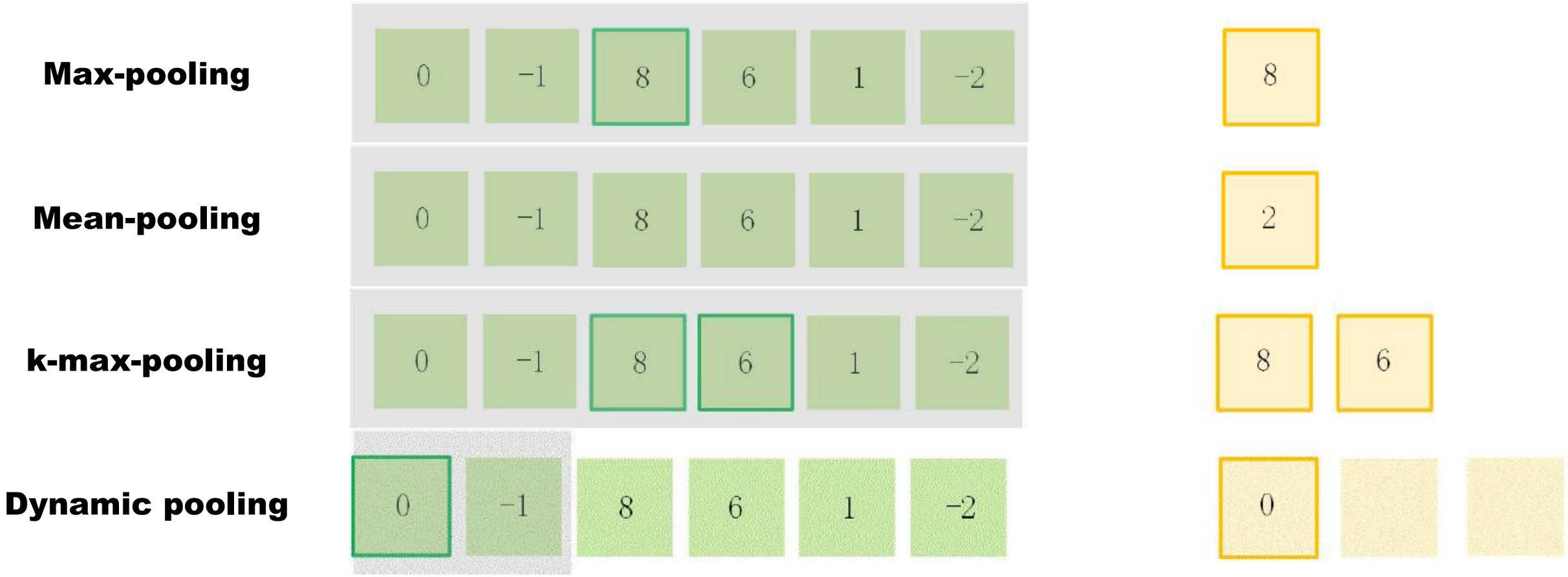
выдаём значение с вероятностью ~ значение

При дифференцировании возвращают градиент в позициях максимумов

С помощью пулинга можно приводить изображение к нужному размеру!

(его можно делать с шагом)

Агрегация (Pooling): виды



http://www.phontron.com/class/nn4nlp2020/schedule.html

Устройство слоя свёрточной НС:

свёртка → нелинейность → пулинг

Мотивация:

- **разреженные взаимодействия (sparse interactions)**

нет связи нейронов «каждый с каждым»

У свёрточных НС мало весов!!!

- **разделение параметров (parameter sharing)**

одна свёртка используется «по всему изображению»

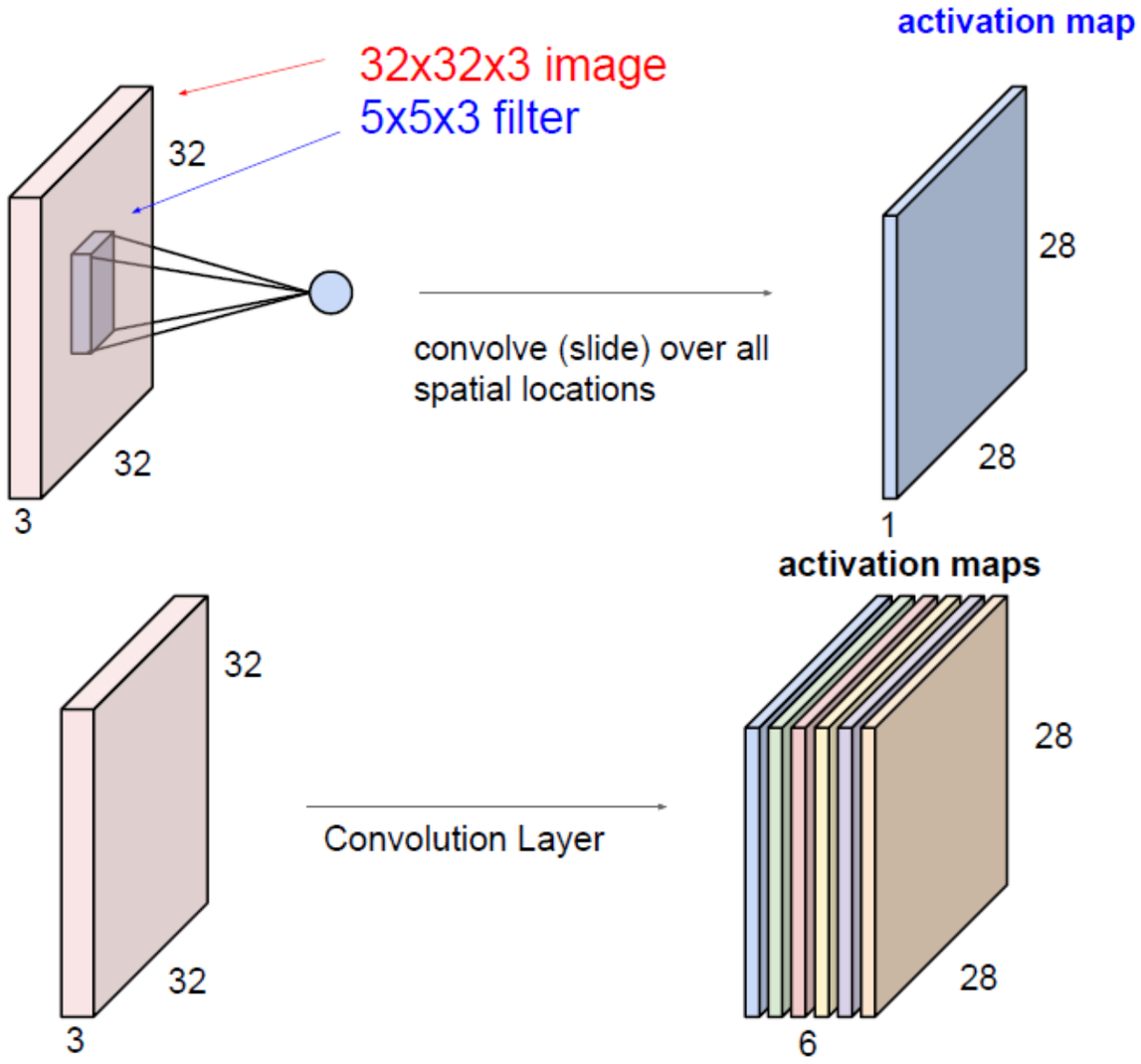
⇒ мало параметров

- **инвариантные преобразования (equivariant representations)**

инвариантность относительно сдвига

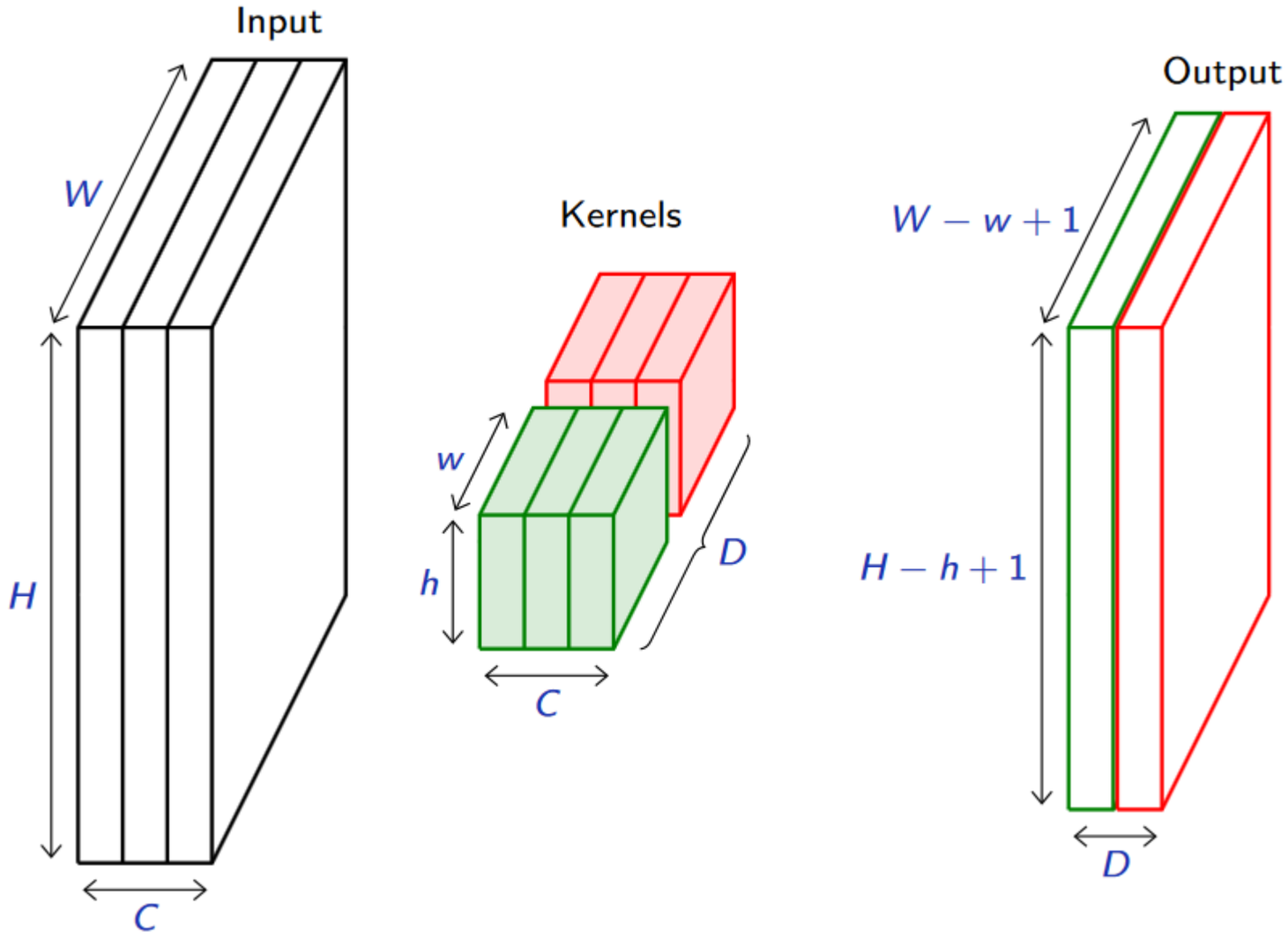
http://www.deeplearningbook.org/contents/convnets.html

Свёрточная НС: тензор → тензор

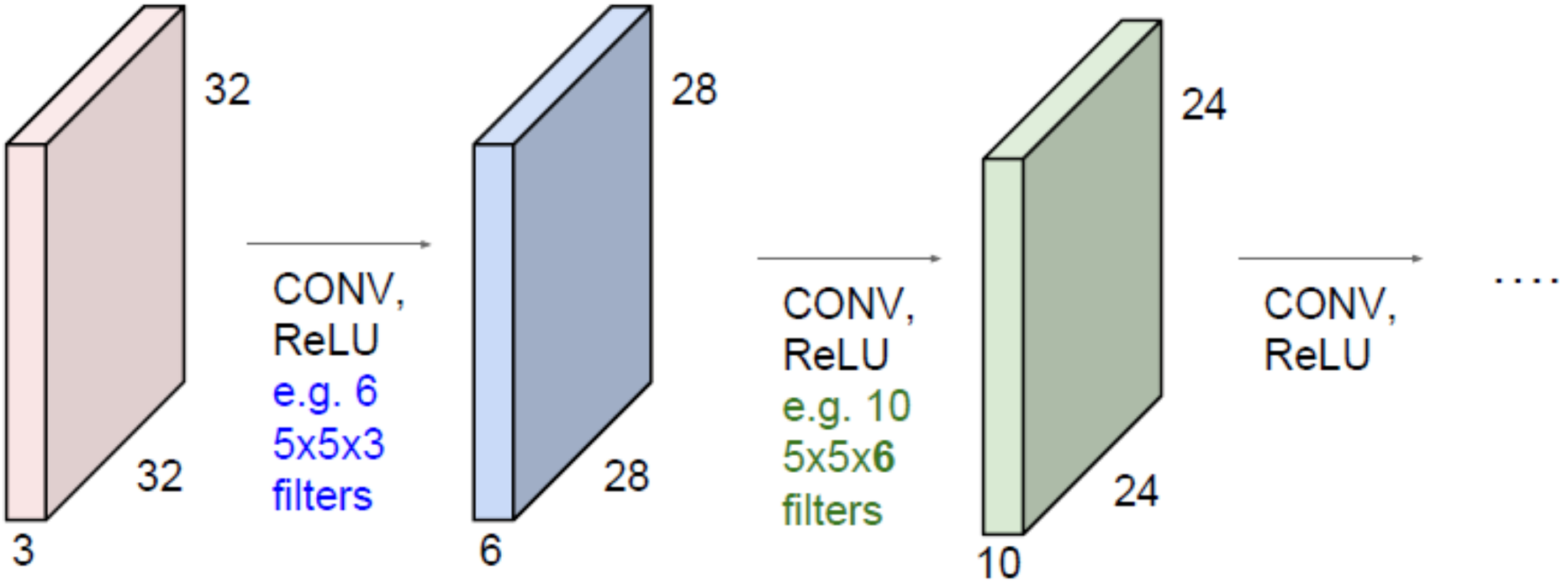


32×32×3 → 28×28×6 (карта признаков)

Свёрточная НС: каждая свёртка – 1 «лист»



Свёрточная НС: тензор → тензор



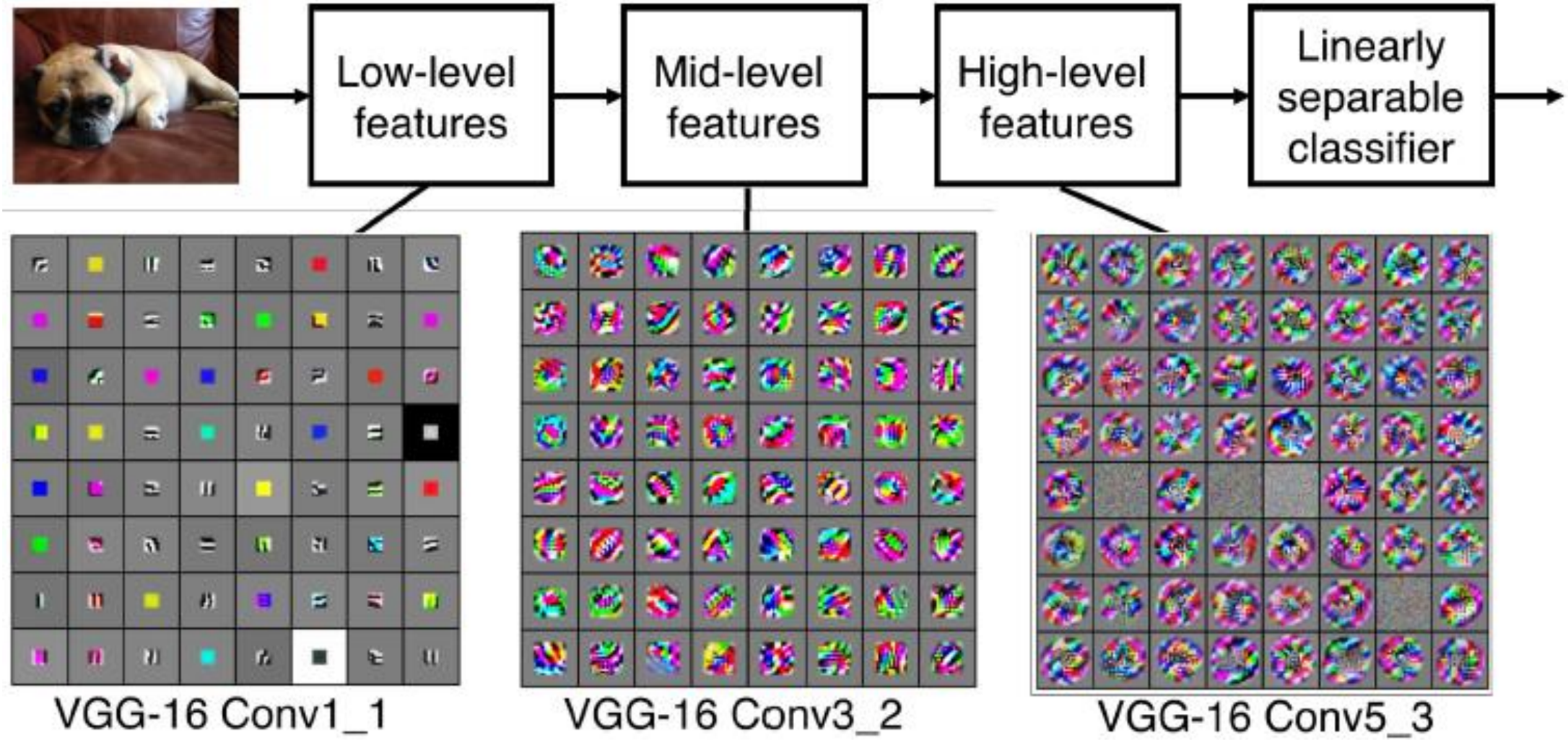
Каждый тензор:

ширина × высота × # признаков / каналов (глубина)

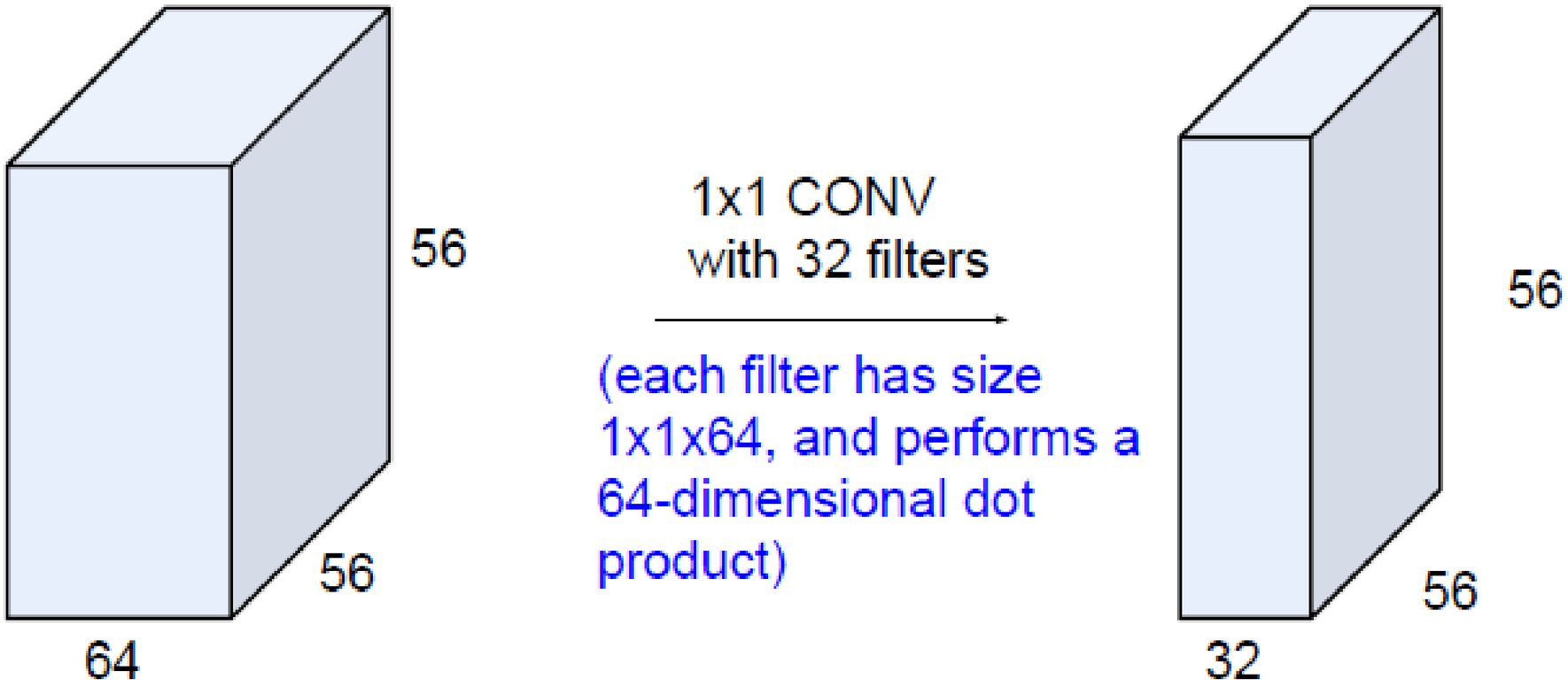
Важно:

Свёрточный слой: тензор → тензор (м.б. другой размерности)

Визуализация признаков

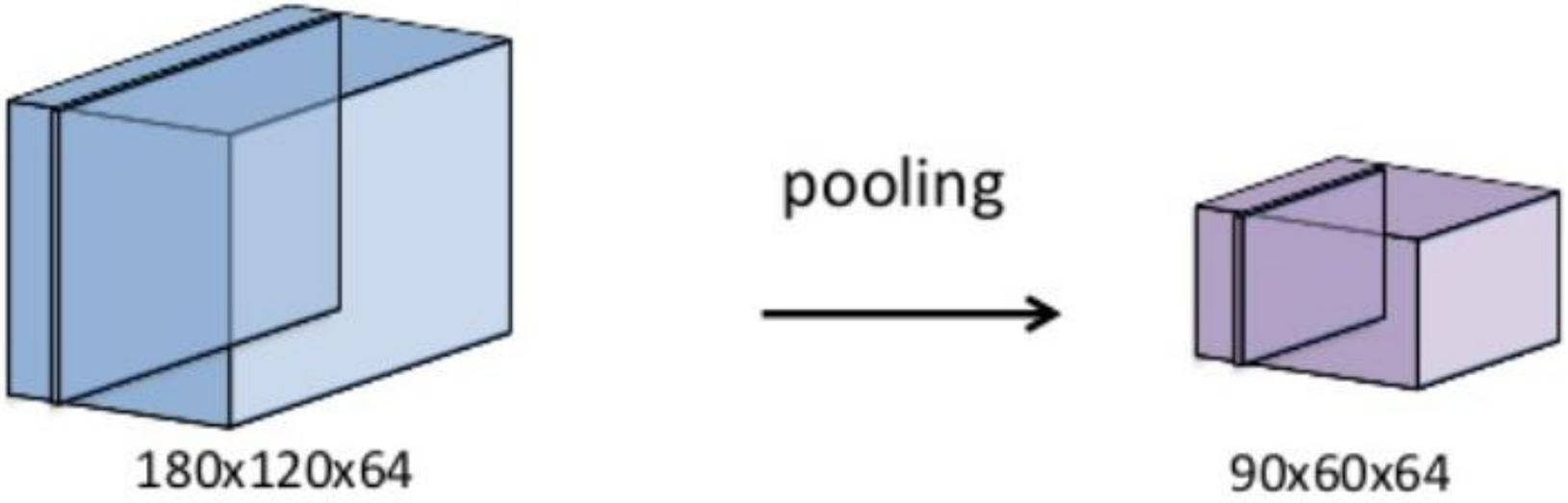


Смысл свёрток 1×1



Преобразование признаков!

Смысл пулинга

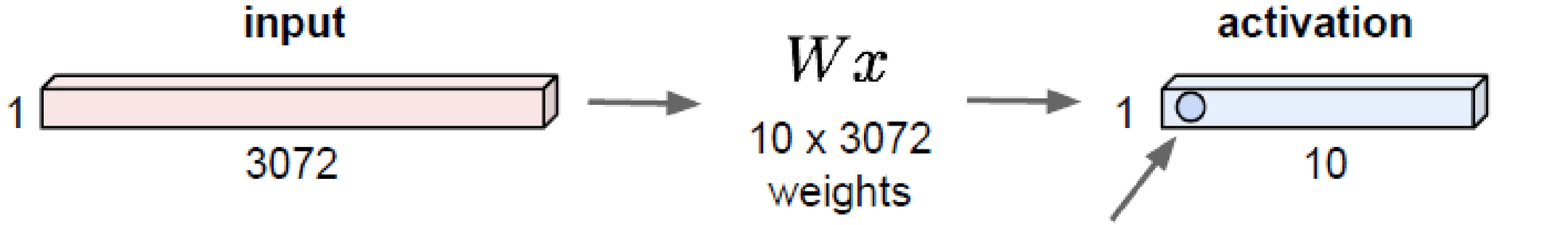


- **уменьшение размеров**
- **на каждой карте действует независимо**

Полносвязный слой

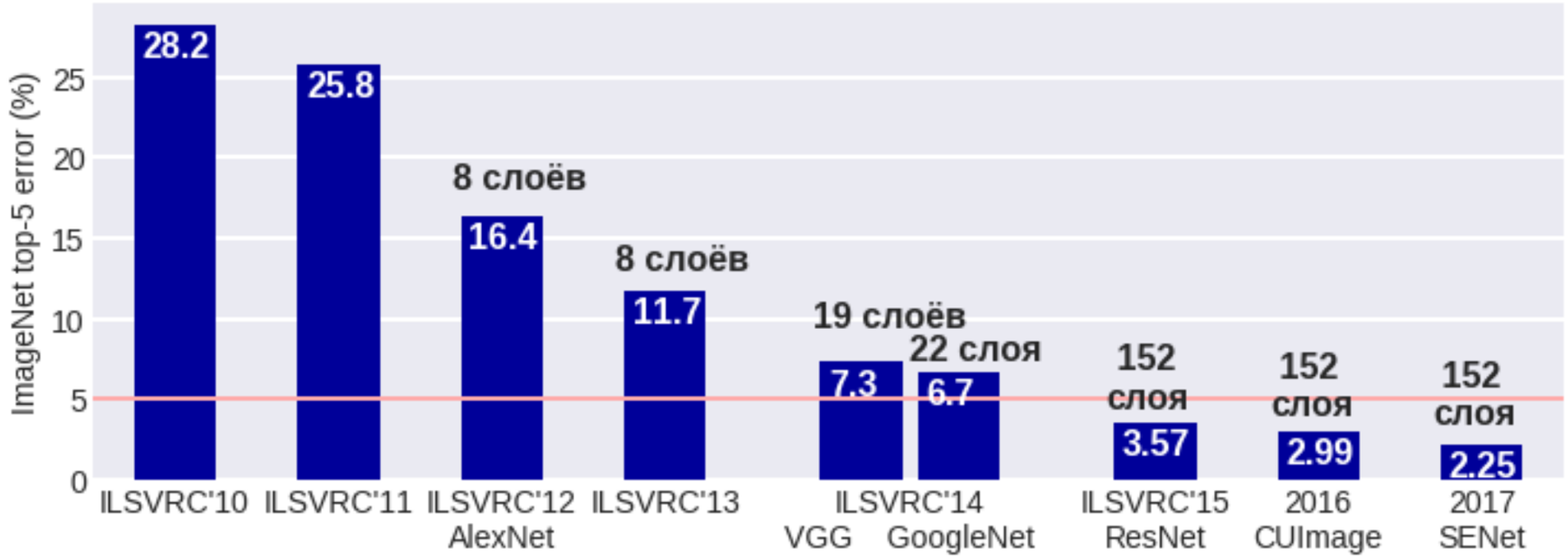
32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

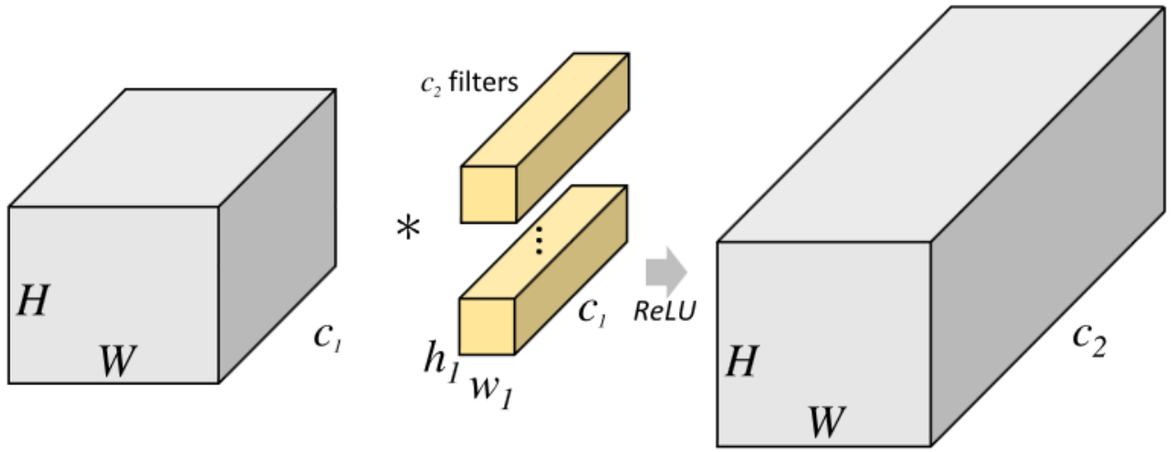
Революция в машинном обучении



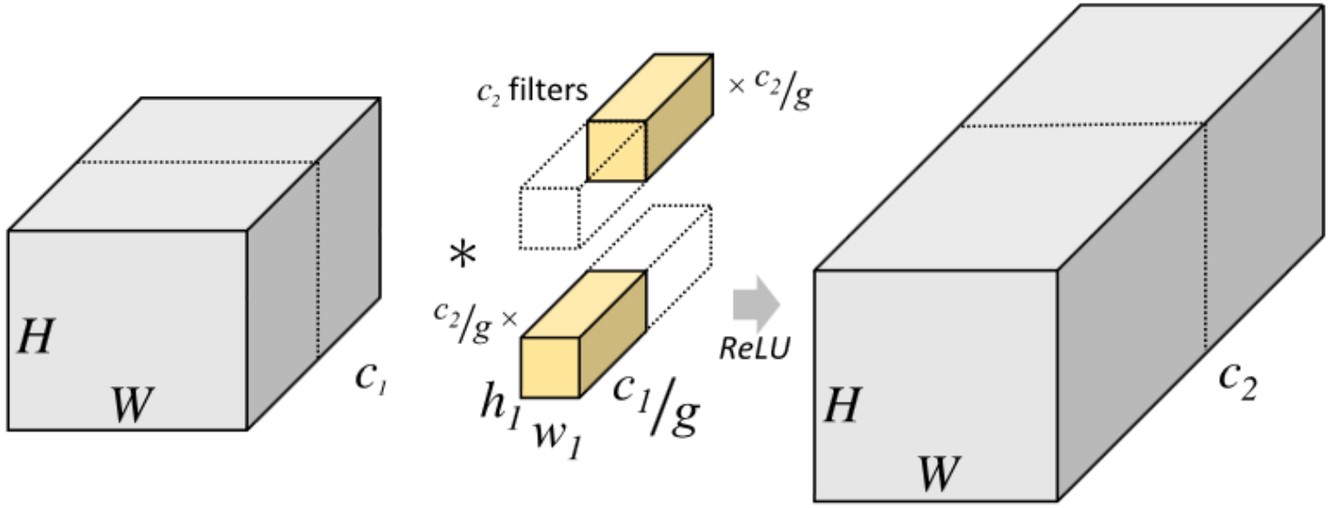
ошибка человека – 5.1

Какие бывают свёртки: Group Convolutions

Более частая ситуация



с разбиением на две группы

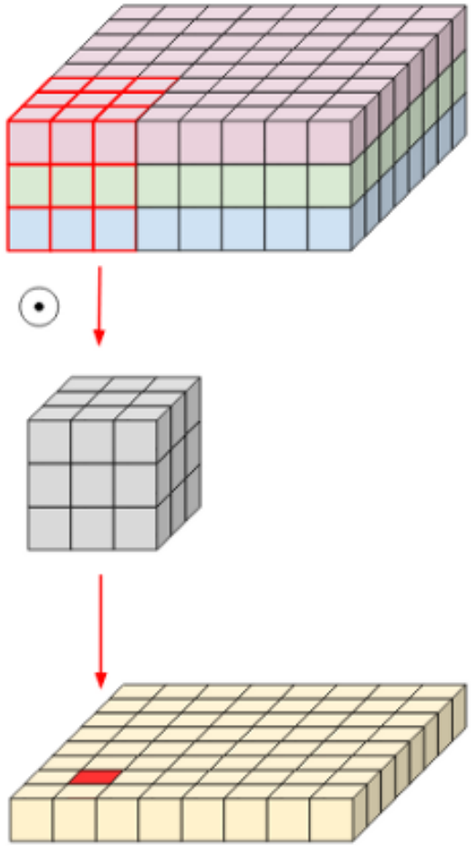


идея из AlexNet, где были ограничения по памяти
могут быть лучшие (разреженные) признаковые представления
но выходные каналы зависят от узкой группы входных

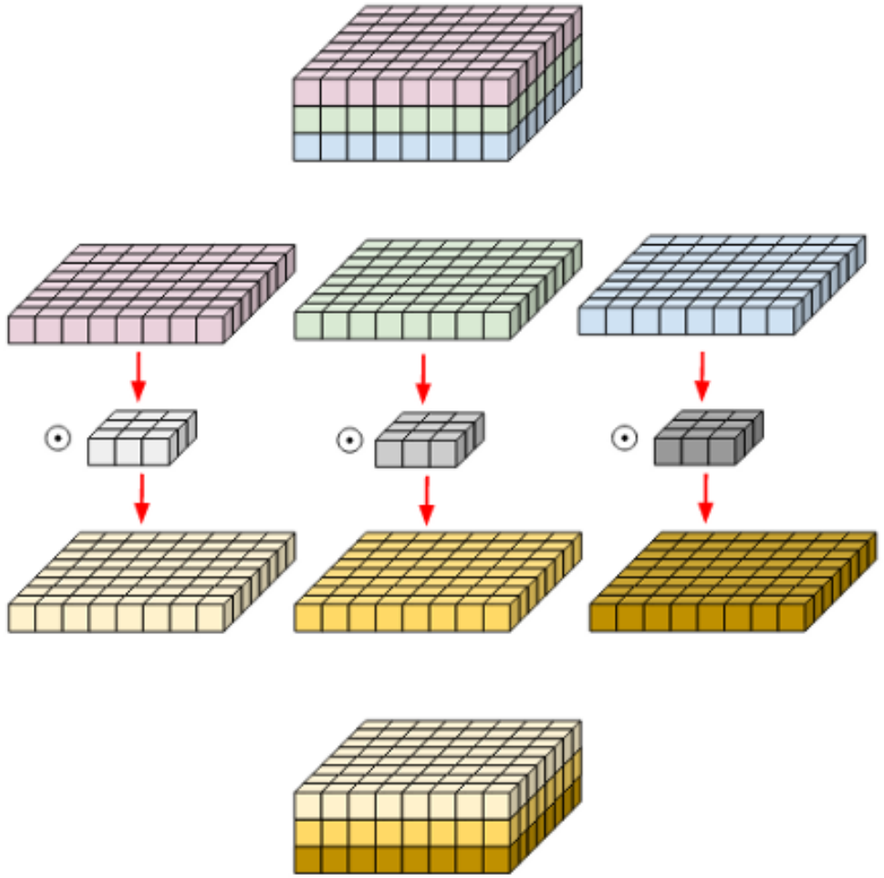
https://blog.yani.io/filter-group-tutorial/

Какие бывают свёртки

convolution



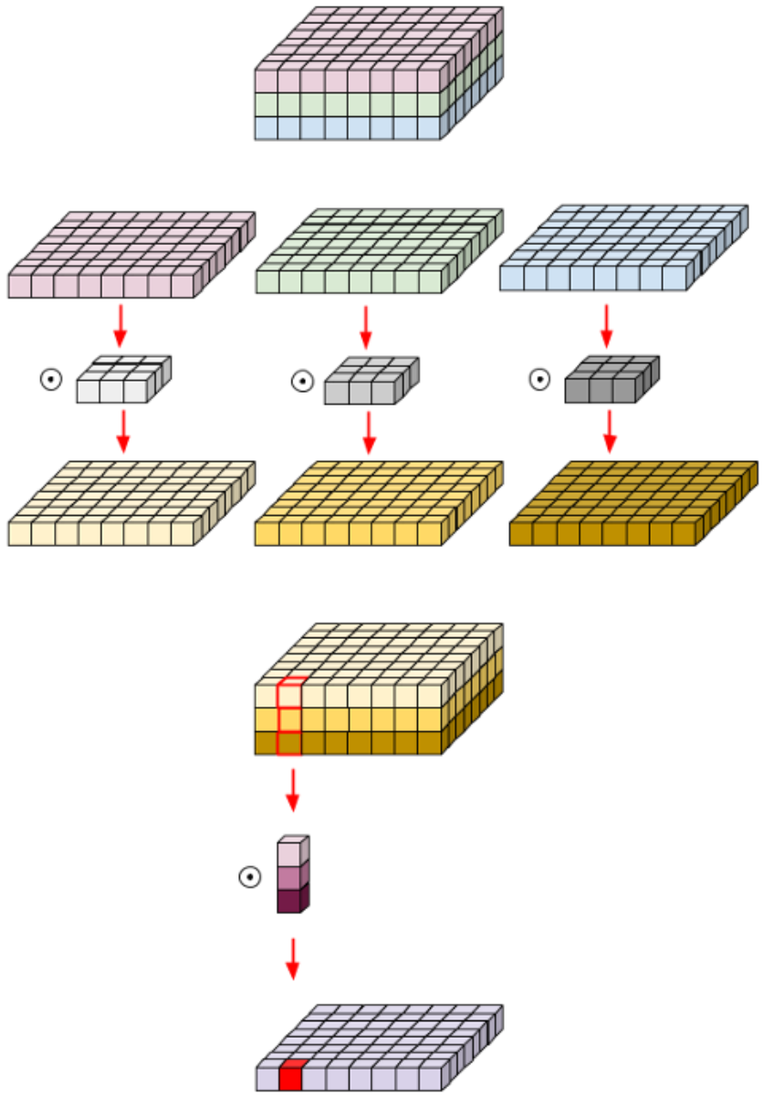
depth-wise convolution



каждый канал «сворачивается» отдельно

https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/

Какие бывают свёртки: Depth-wise separable convolution



теперь результат зависит от всех каналов

```
S=128, F=3, inC=3, outC=16

Regular convolution:
Parameters:
   3*3*3*16 = 432
Computation cost:
   3*3*3*128*128*16 = ~7e6

Depthwise separable convolution:
Parameters:
   3*3*3+3*16 = 75
Computation cost:
   3*3*3*128*128+128*128*3*16
   = ~1.2e6
```

Итог

В изображениях свёртки – естественная операция

Естественное устройство CNN: свёртка, пулинг, нелинейность, полносвязность

В отличие от классического CV не придумываем фильтры

Они обучаются сами!

Свёртка – первый пример разделения весов.

Есть способы экономии параметров – и ими пользуются!

Свёртки продолжают совершенствоваться

(более разумные представления, экономия параметров)