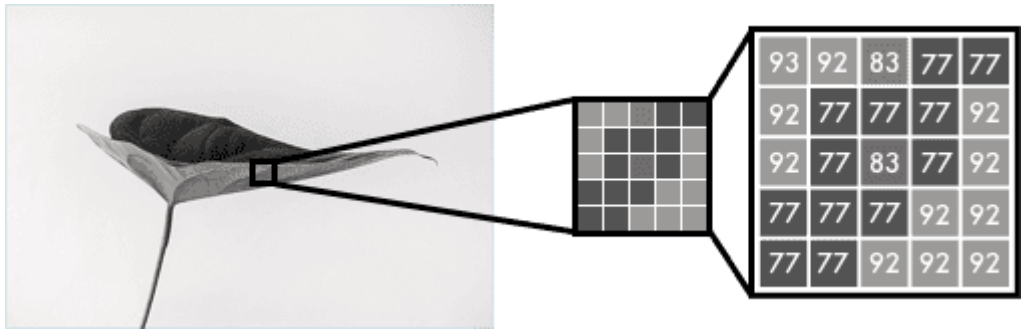
Глубокое обучение:
Свёрточные нейронные сети

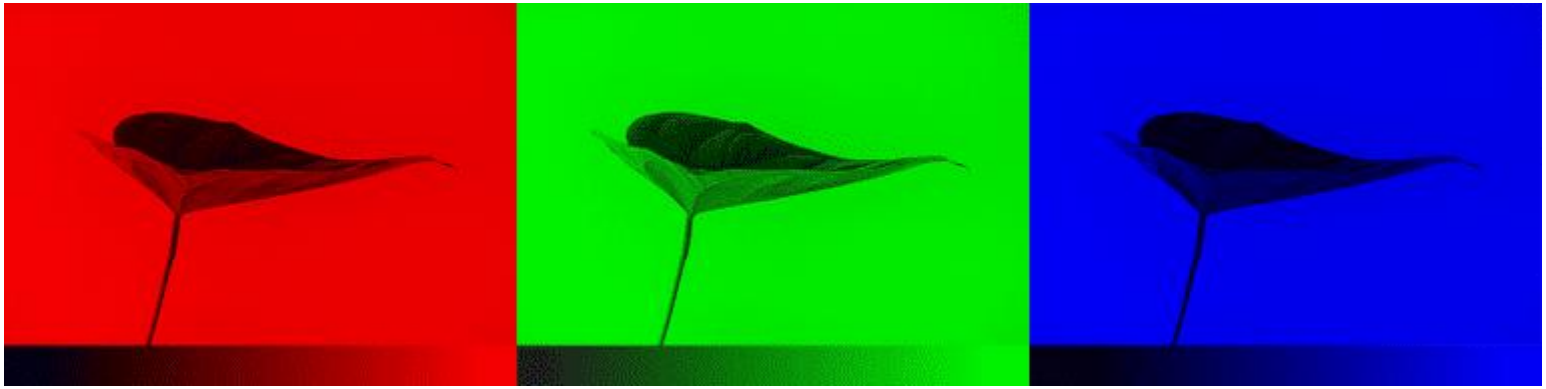
Дьяконов А.Г.

Московский государственный университет имени М.В. Ломоносова (Москва, Россия)

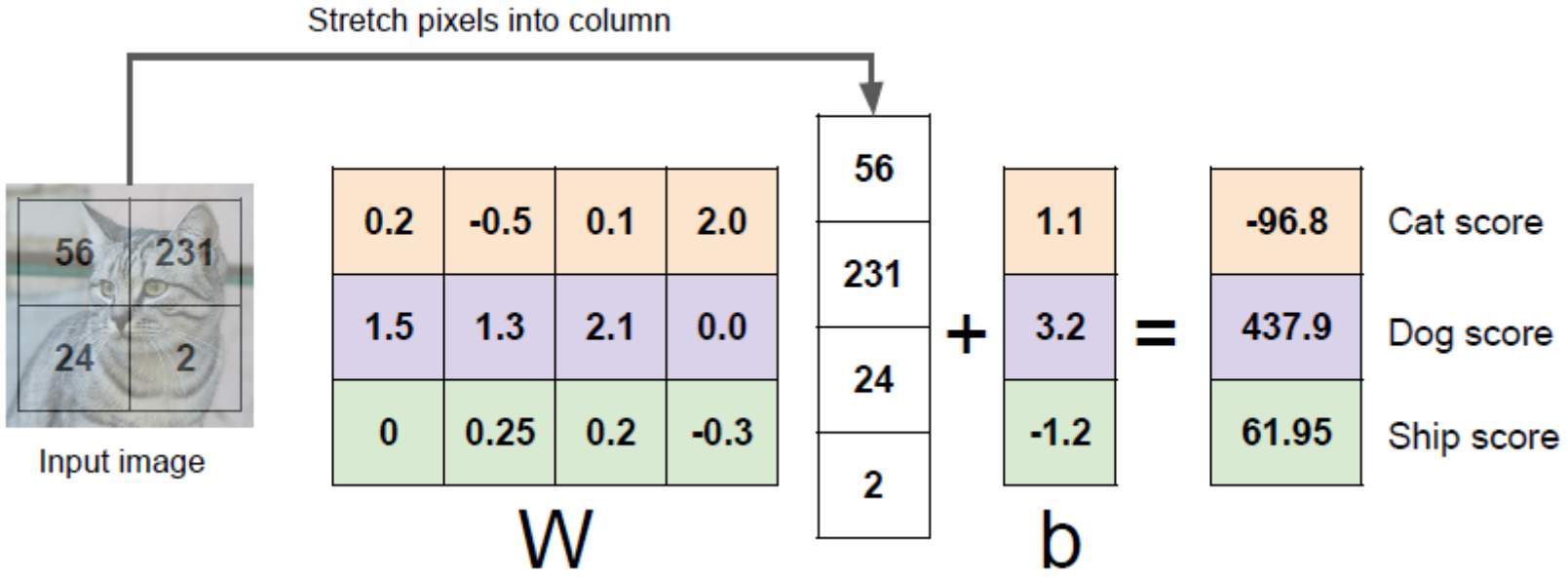
Что такое изображение

Что такое изображение





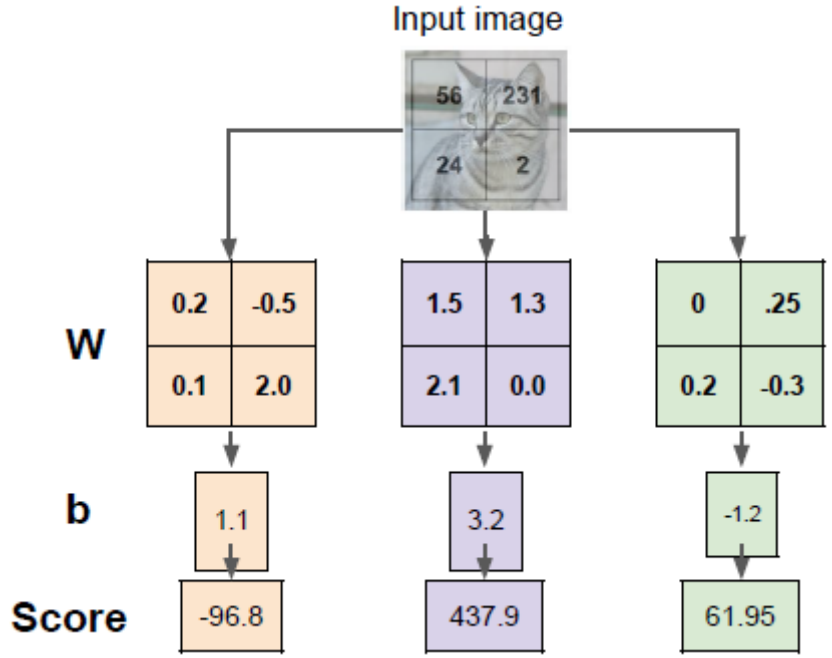
Линейный подход к классификации на несколько классов



3 класса = 3 вектора признаков

Изображение = вытянуть в вектор

Линейный подход к классификации на несколько классов



Вектора весов можно потом опять «прорешейпить» в изображения:

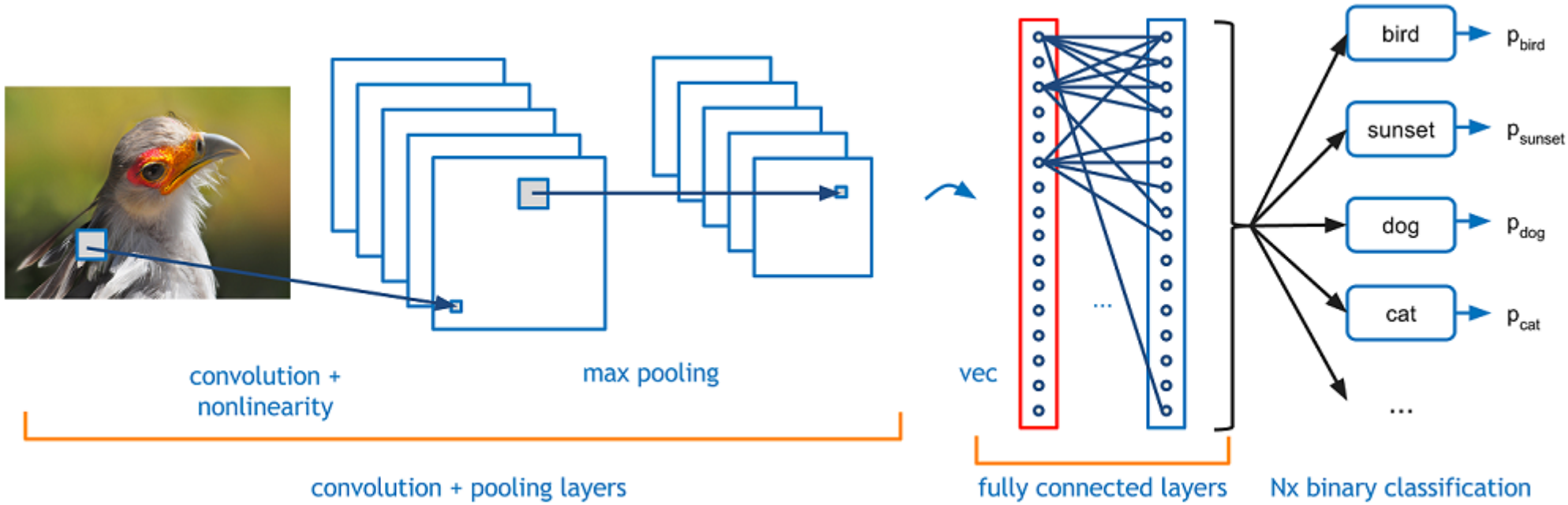
Проблема

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!

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/

Что такое свёртка (Convolution)

сначала 1D-свёртки

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:

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

Свёртка (Convolution)

-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		

Свёртка (Convolution)

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

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

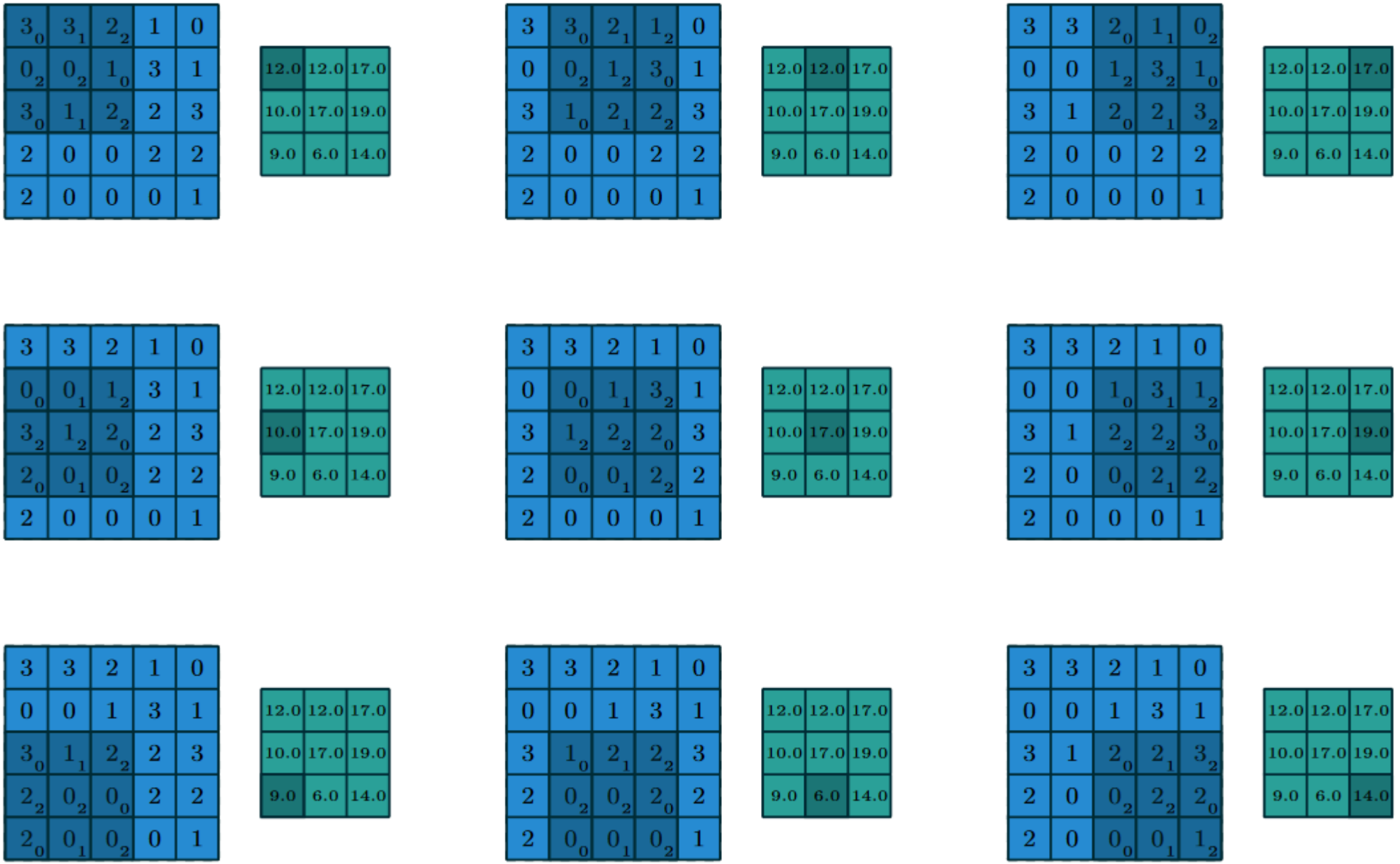
$$= \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$$

может быть немного другая индексация

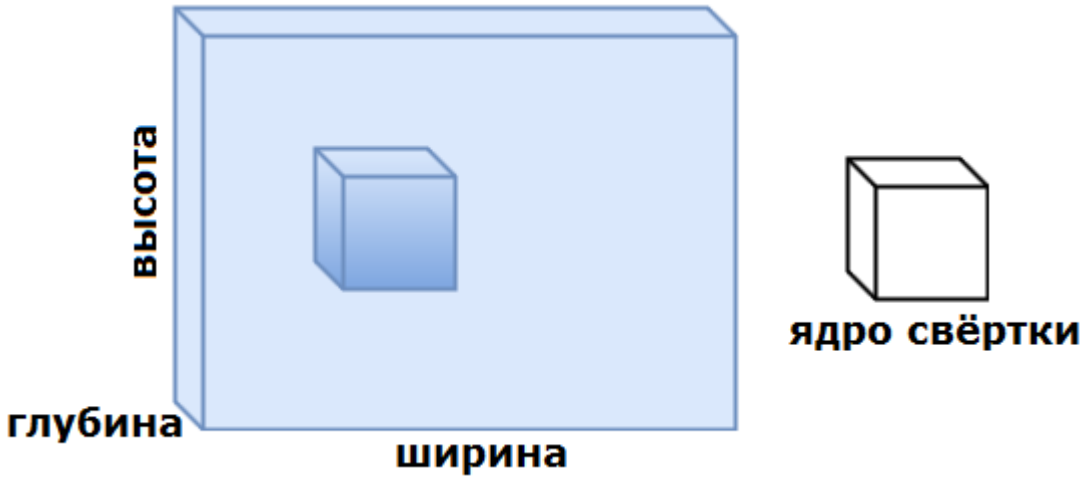
хорошее объяснение:

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

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


3 3 2 1 0
0 0 1 3 1
3 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

Свёртка (Convolution)



Глубина (depth) / число каналов

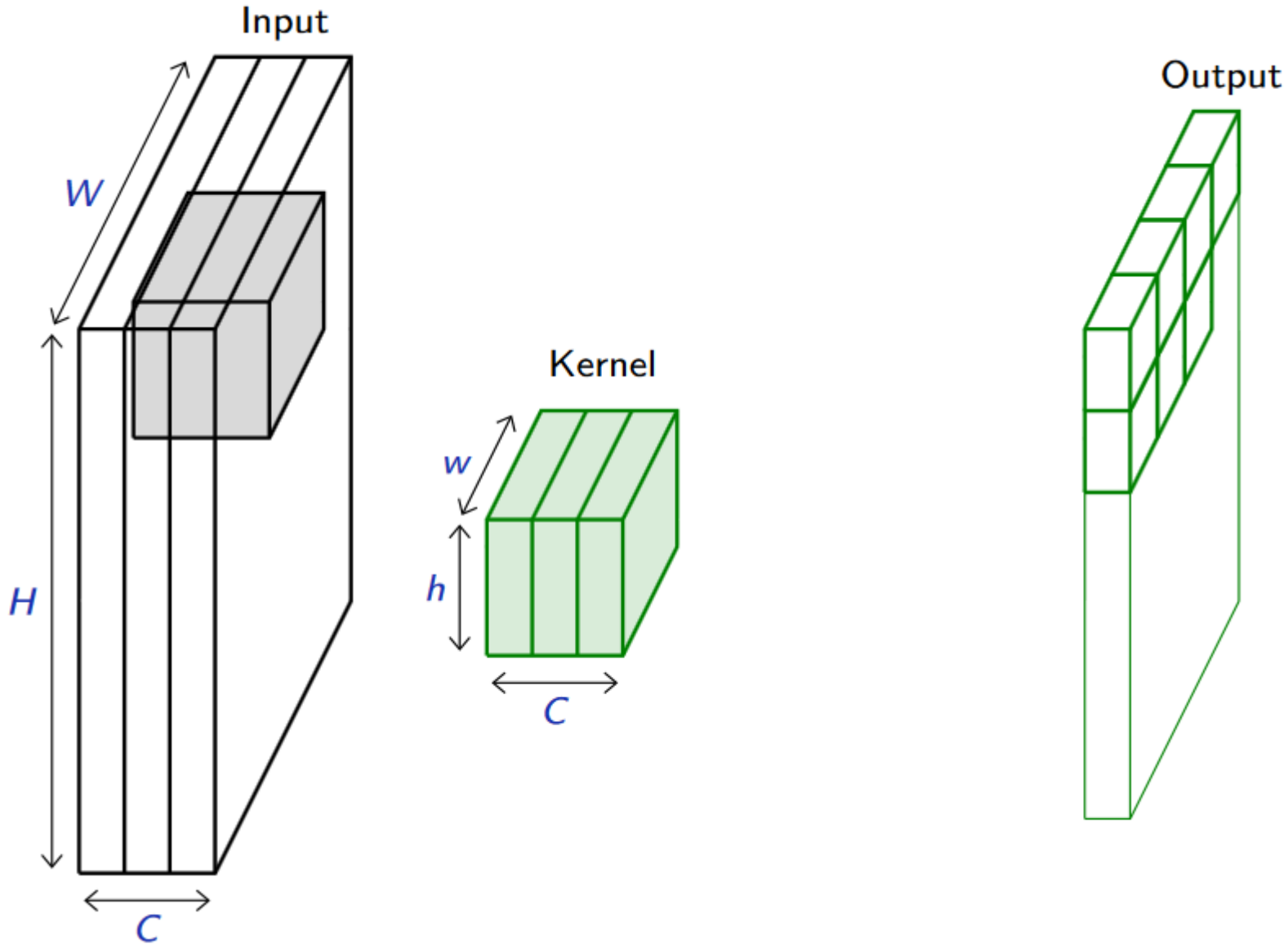
Высота (height) и ширина (width) каждого ядра

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?

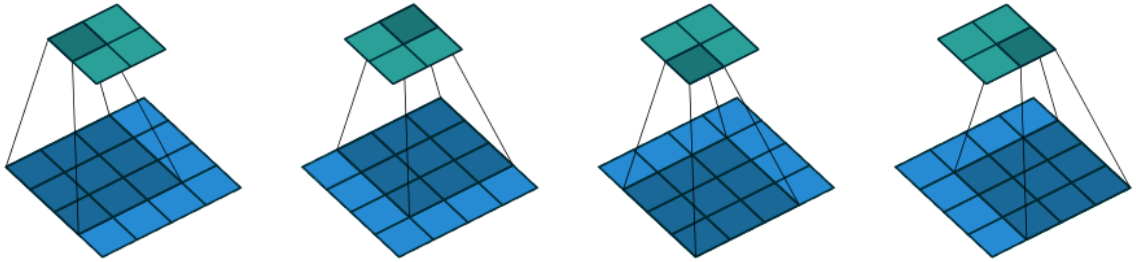
```
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, color

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')
```

Отступ (padding) – чтобы сохранялись размеры изображения

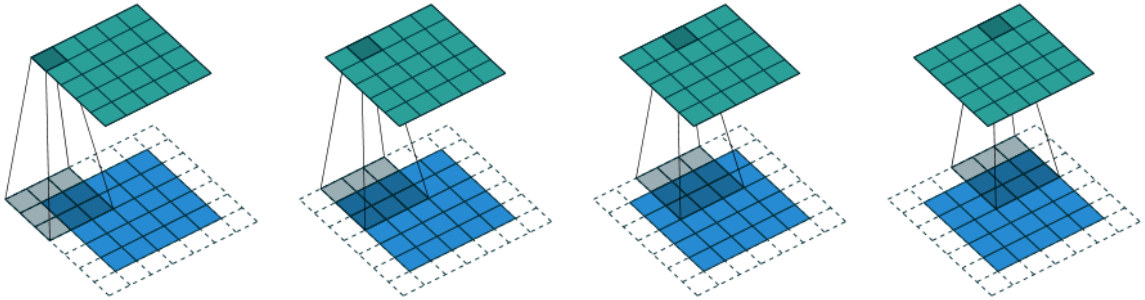
padding = VALID (в TF)

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



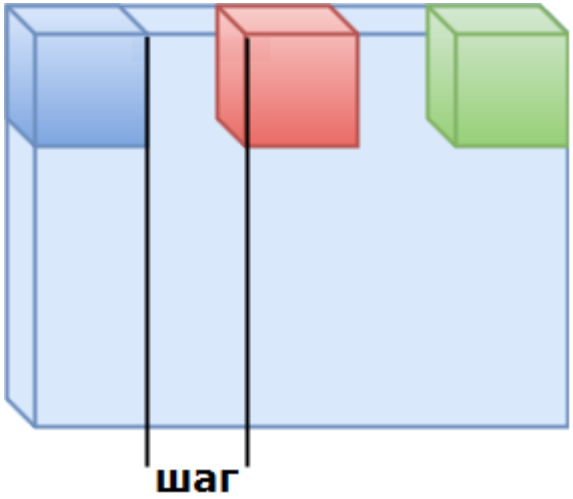
padding = SAME (в TF)

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



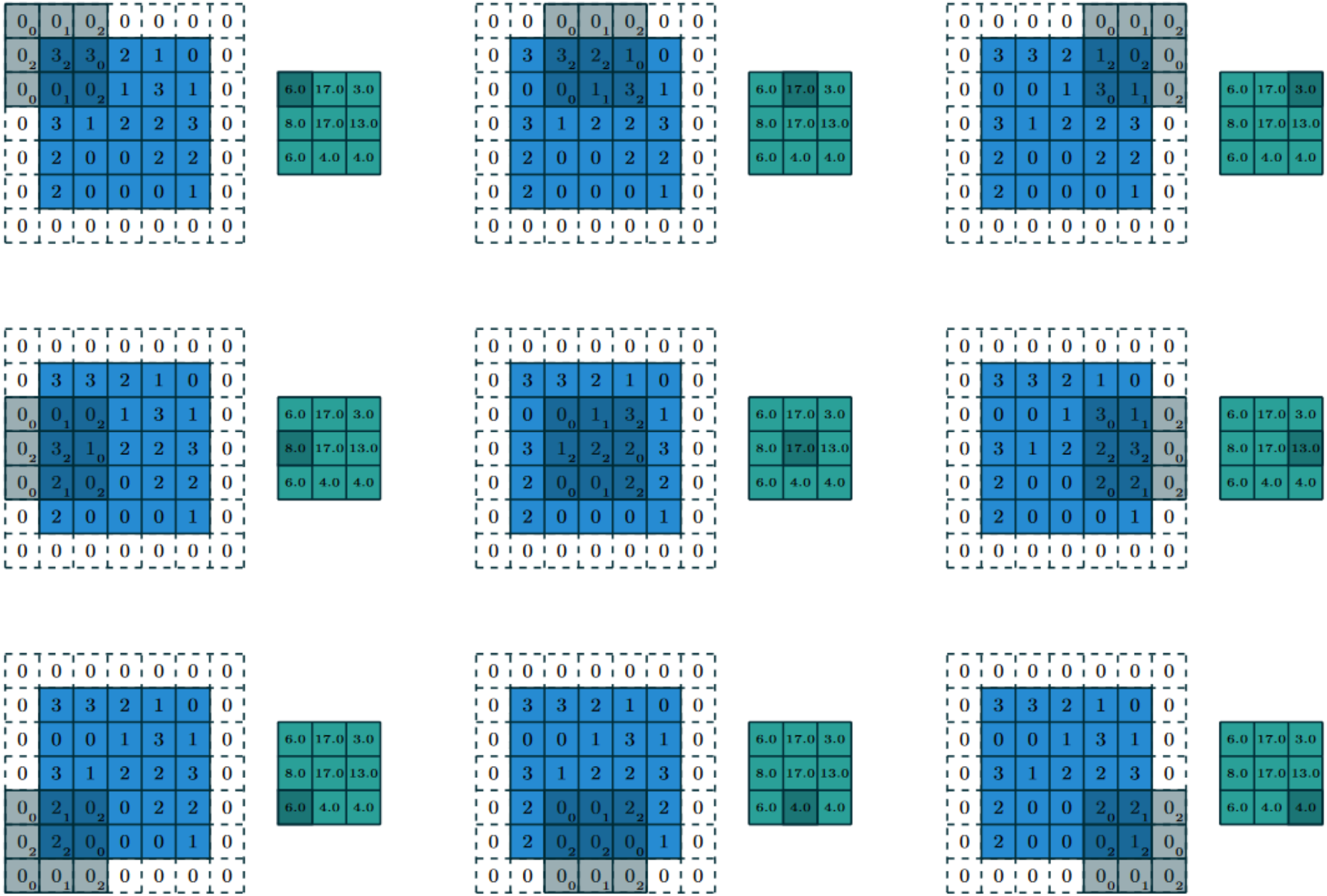
Шаг (stride)

смещаемся при вычислении свёртки
(можно в каждой её размерности)



с шагом 2

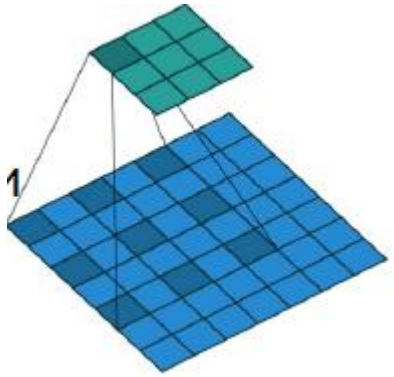
$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 \\ 3 & 4 & 5 & 2 & 1 \\ 1 & 0 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 & 2 \\ 1 & 2 & 4 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -5 & 2 \end{bmatrix}$$


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!

Поэтому надо быстро делать матричные перемножения…

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

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

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

Свёртка (Convolution): мотивация

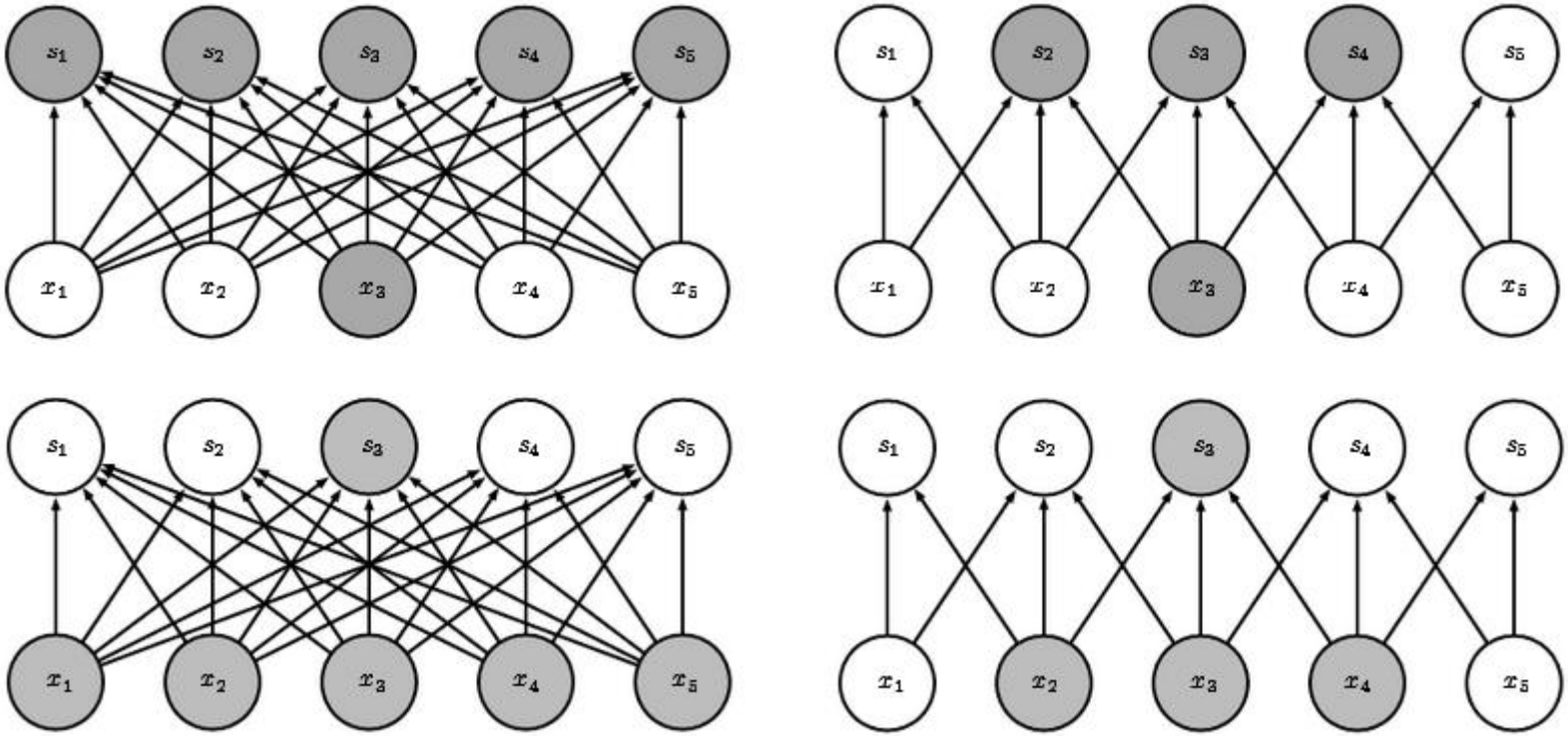
Раньше: обработка изображений – специально построенные свёртки

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

Важно: свёртку можно применять к изображениям любых размеров!

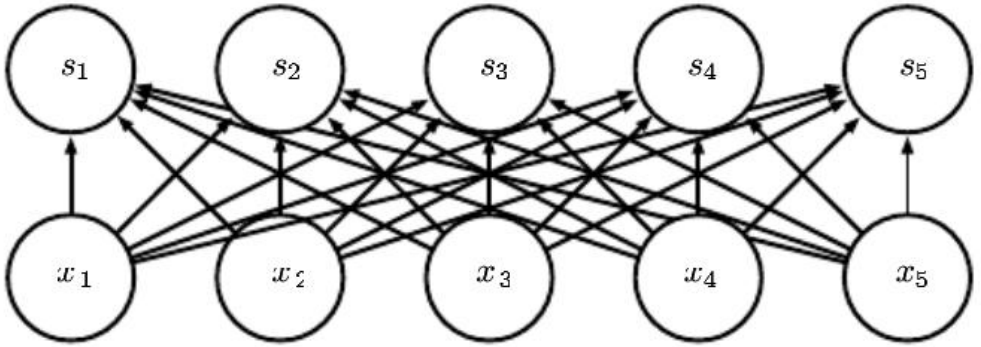
Нет ограничений на размеры входа…

Разреженные взаимодействия (sparse interactions)



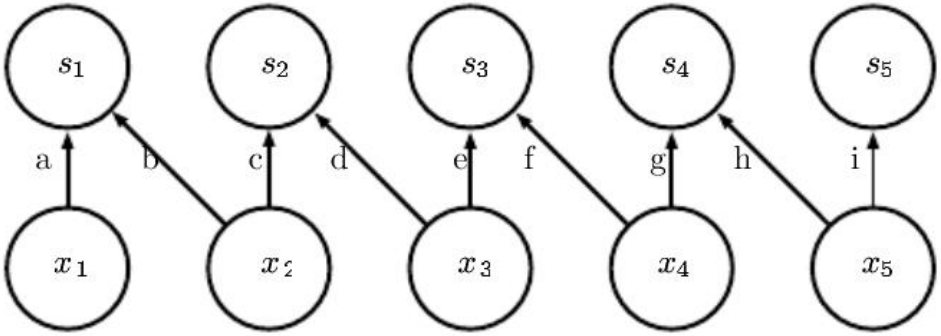
http://www.deeplearningbook.org/contents/convnets.html

Полная связность (full connections)

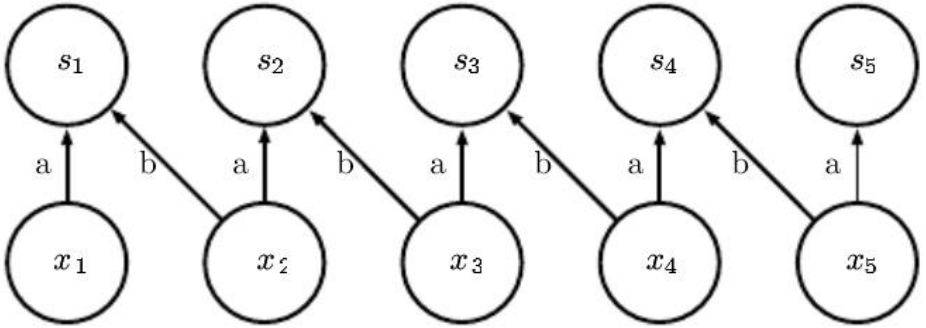


Локальная связность (local connections)

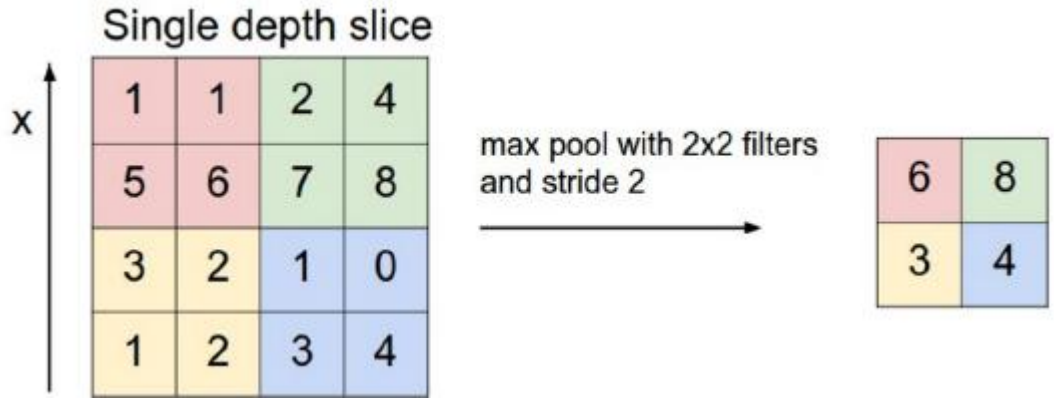
неразделяемая свёртка (unshared convolution)

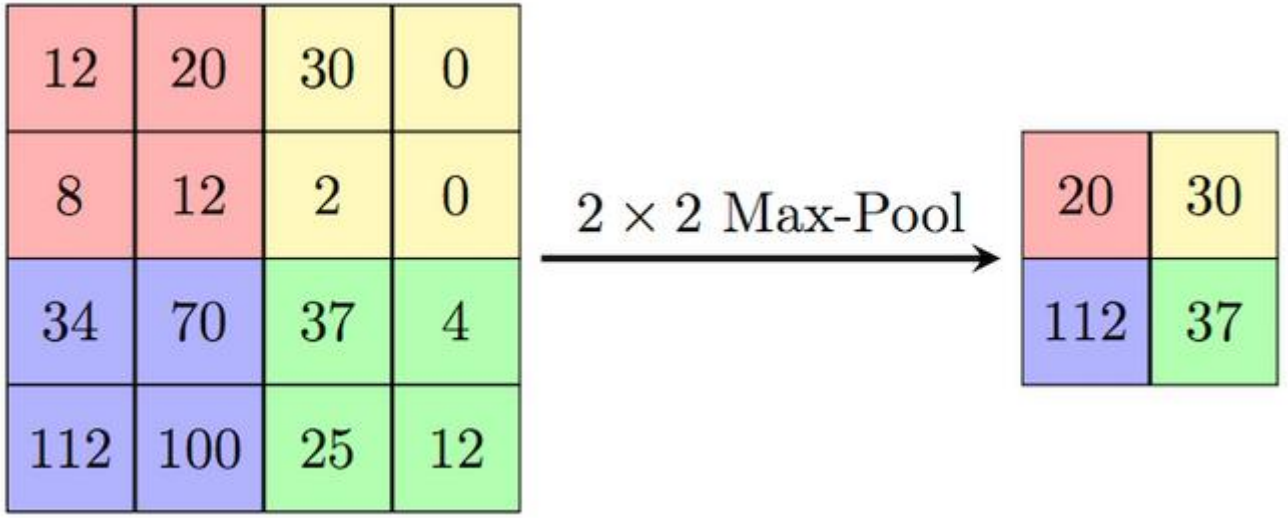


Свёртка (convolution)

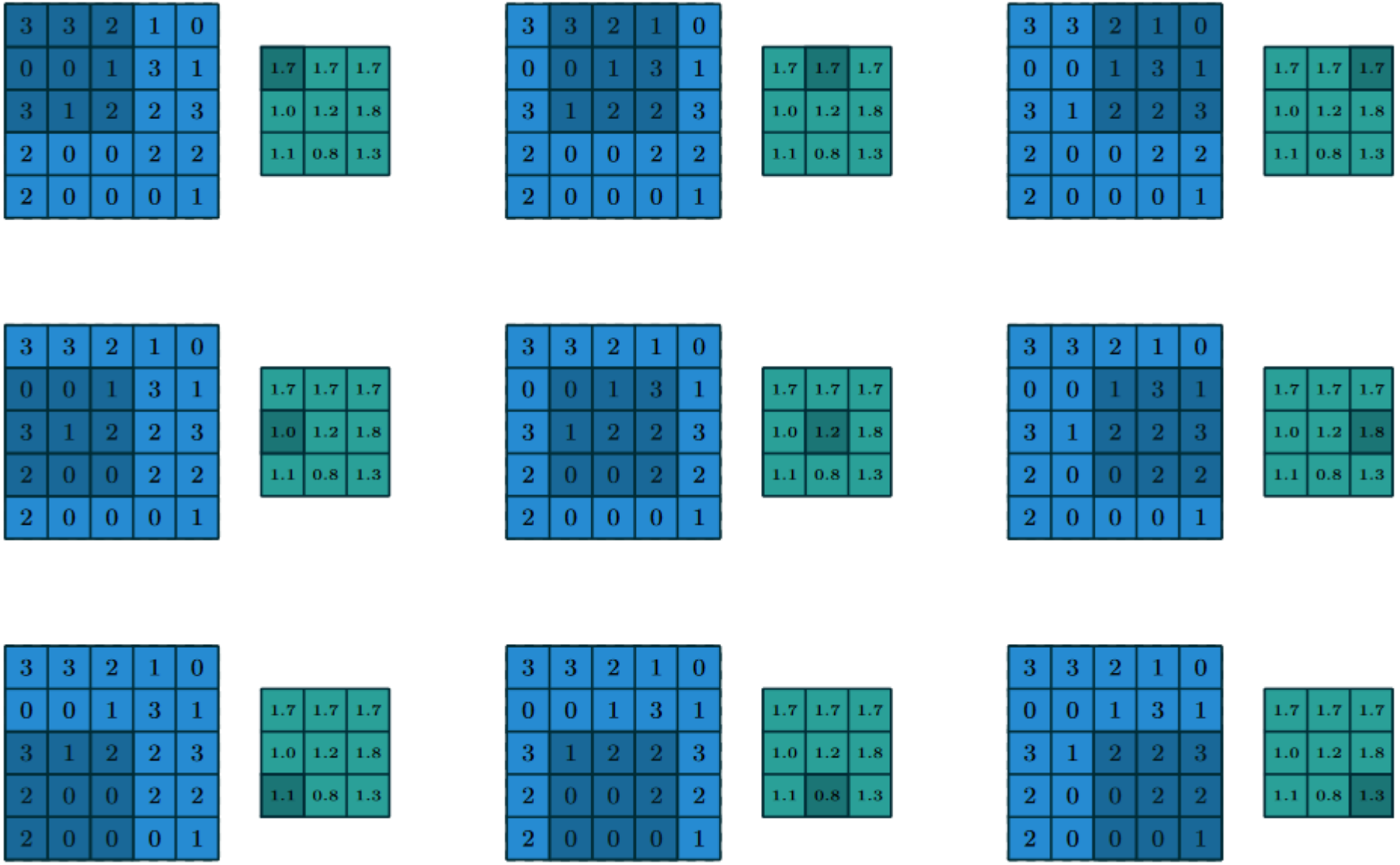


Pooling (агрегация, субдискретизация / subsampling)



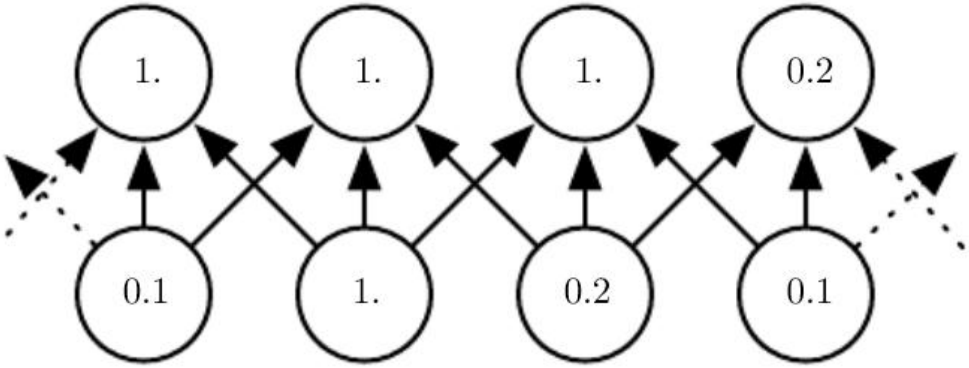


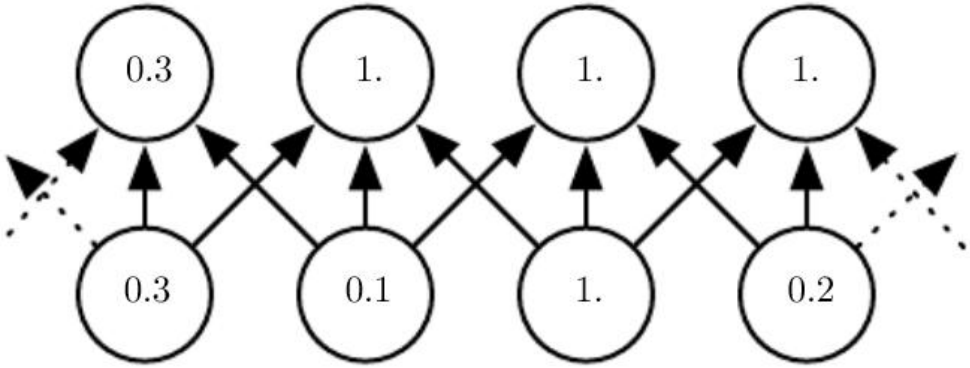
Агрегация (Pooling) усреднением



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)

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)

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»

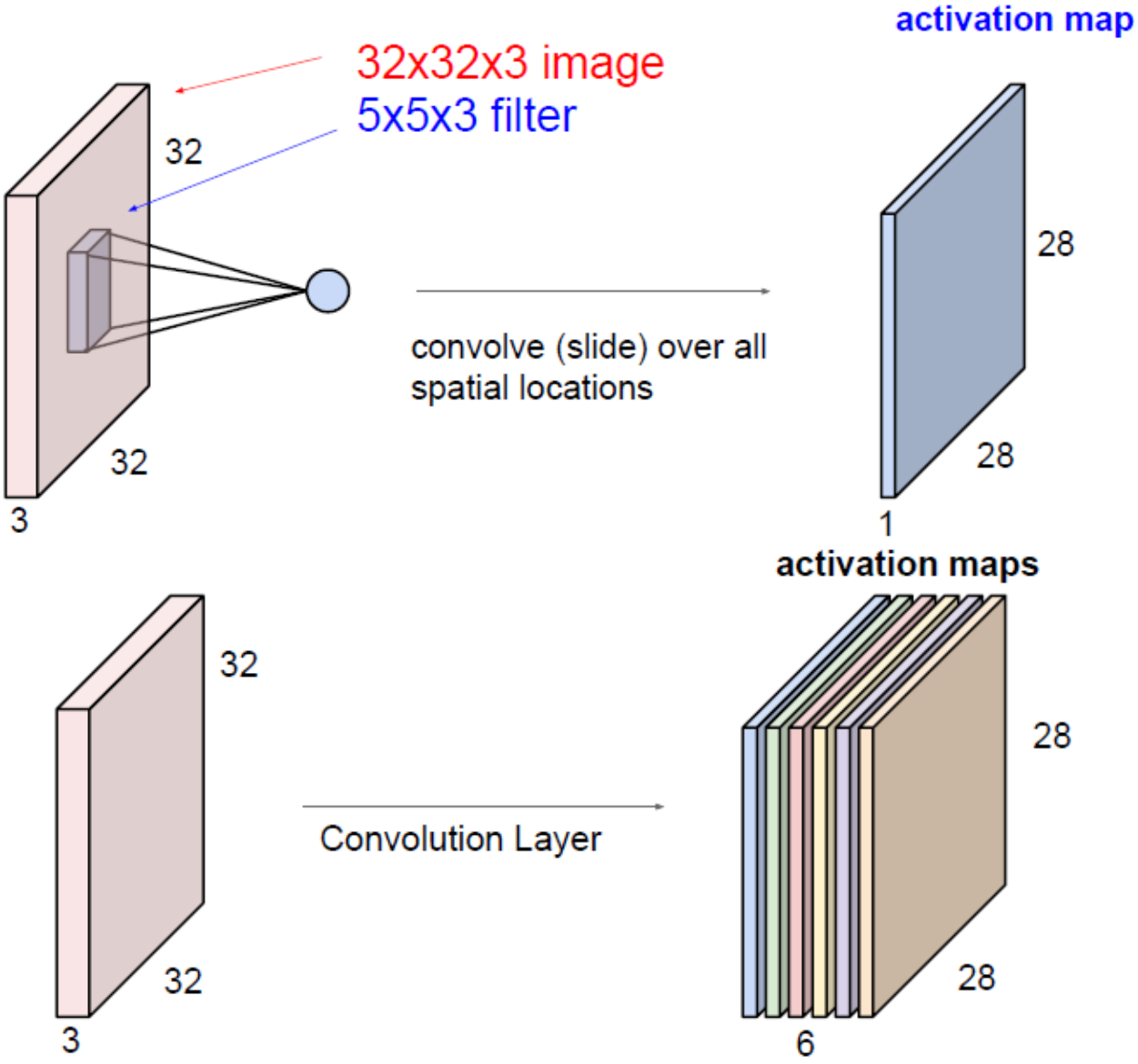
⇒ мало параметров

- **инвариантные преобразования (equivariant representations)**

инвариантность относительно сдвига

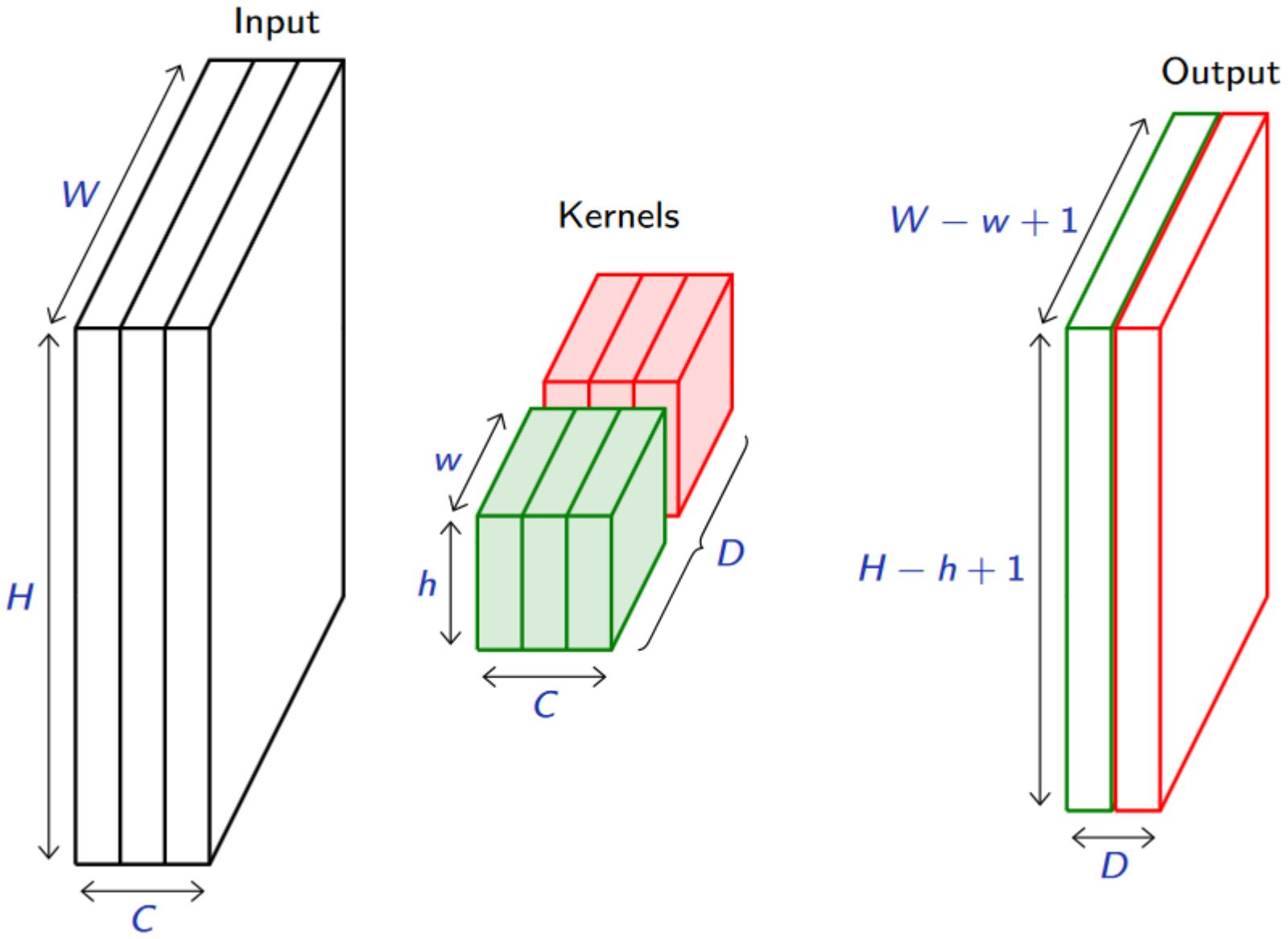
http://www.deeplearningbook.org/contents/convnets.html

Свёрточная НС: тензор → тензор

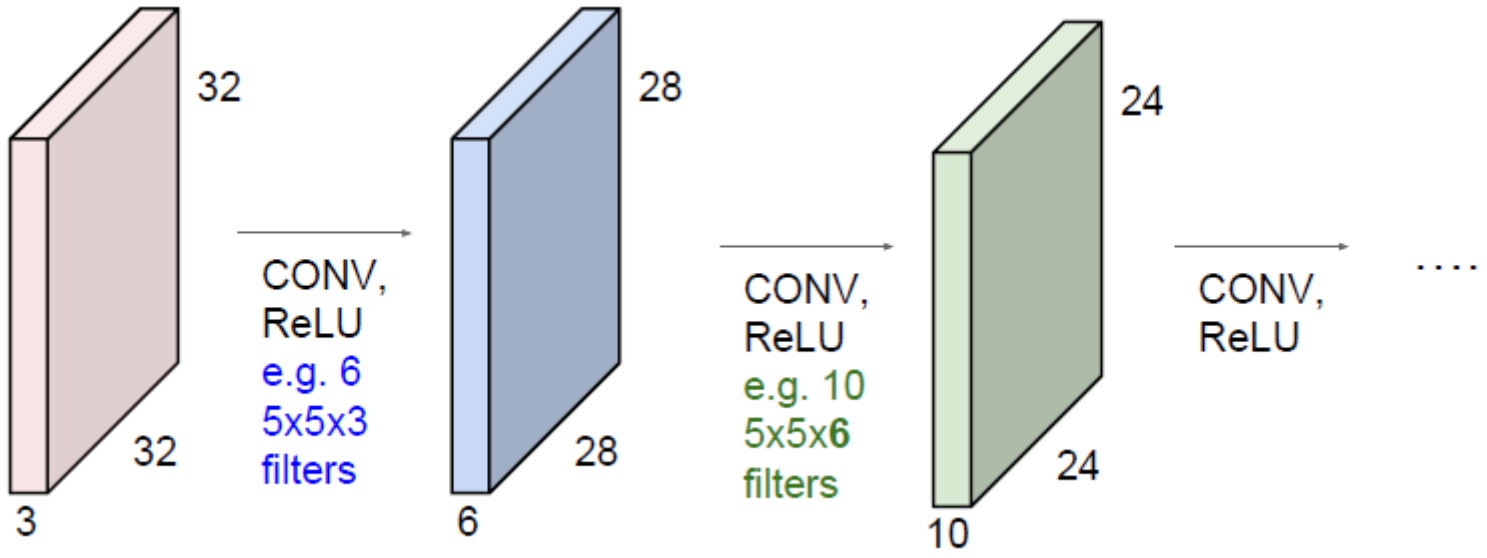


32×32×3 → 28×28×6 (карта признаков)

Свёрточная НС: каждая свёртка – 1 «лист»



Свёрточная НС: тензор → тензор



Каждый тензор:

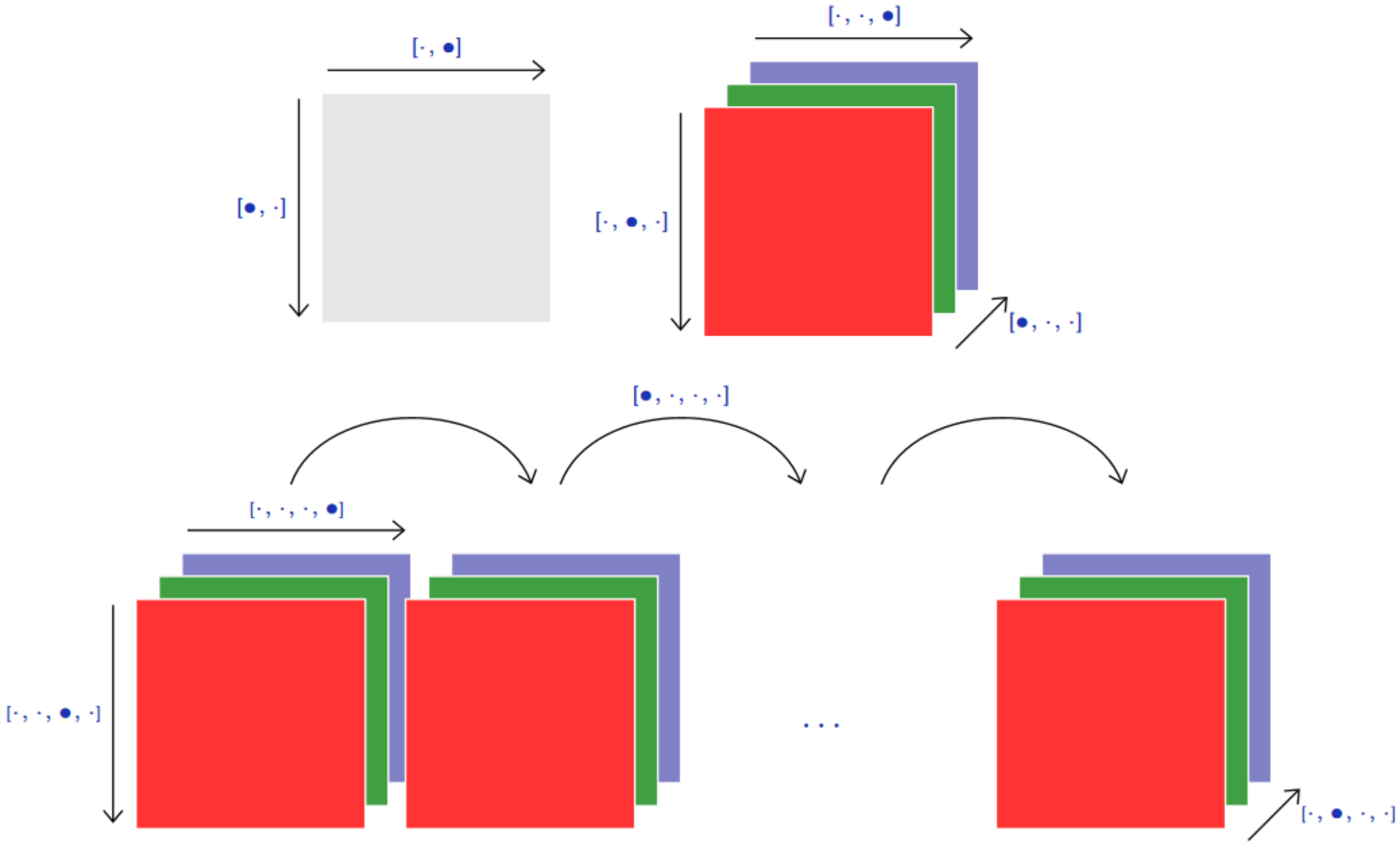
ширина × высота × # признаков / каналов (глубина)

Важно:

Свёрточный слой:

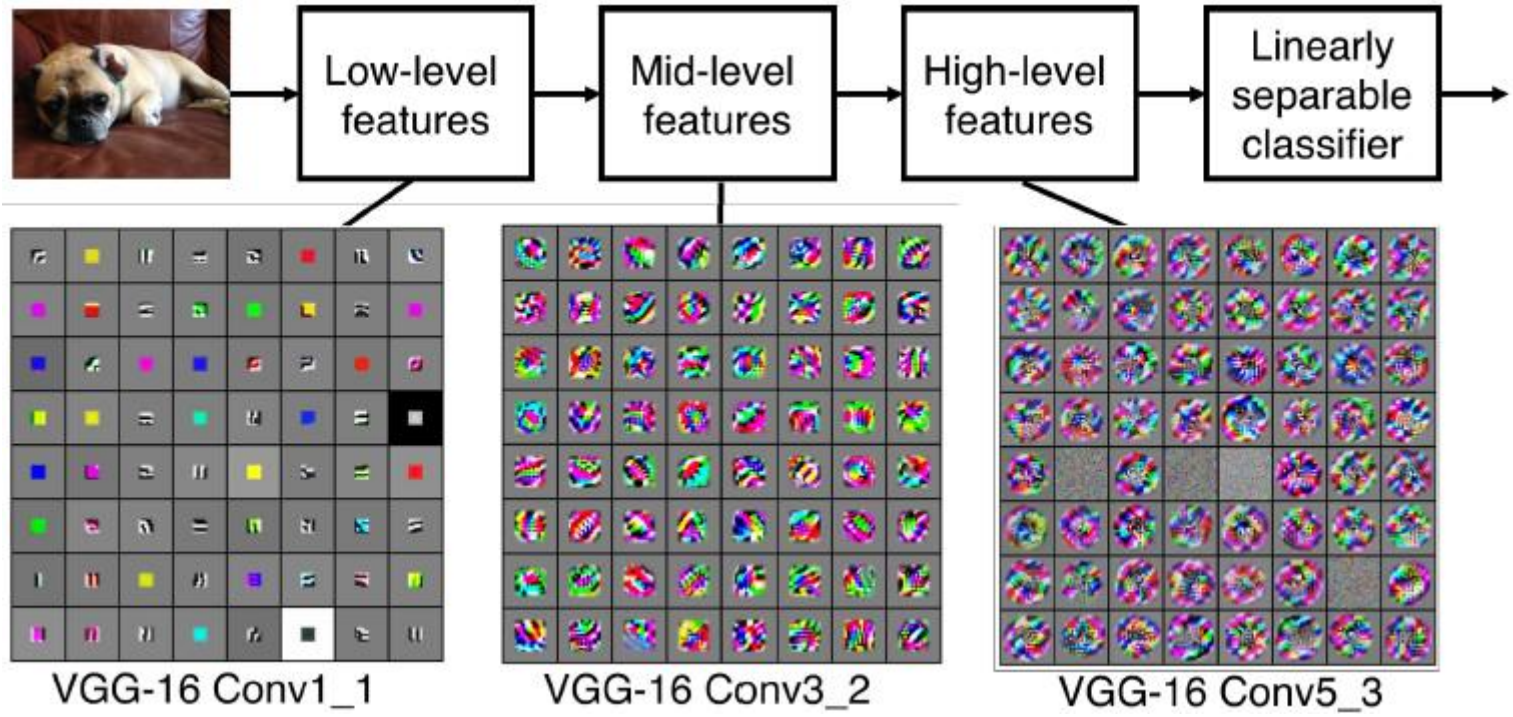
тензор → тензор (м.б. другой размерности)

Тензор – у нас «многомерная матрица»

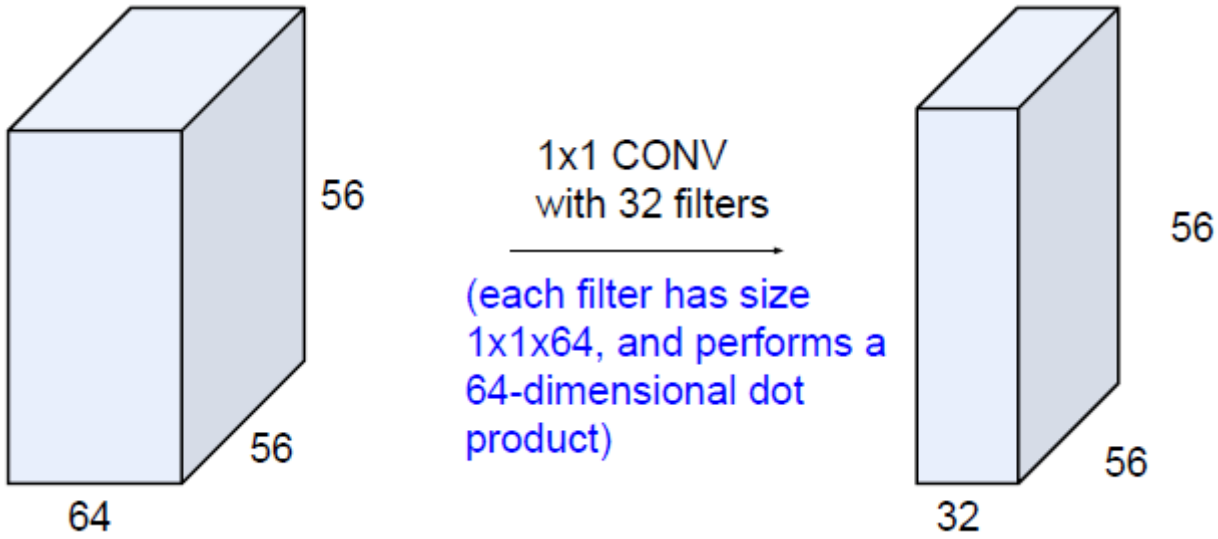


https://fleuret.org/ee559/ee559-slides-1-5-high-dimension-tensors.pdf

Визуализация признаков

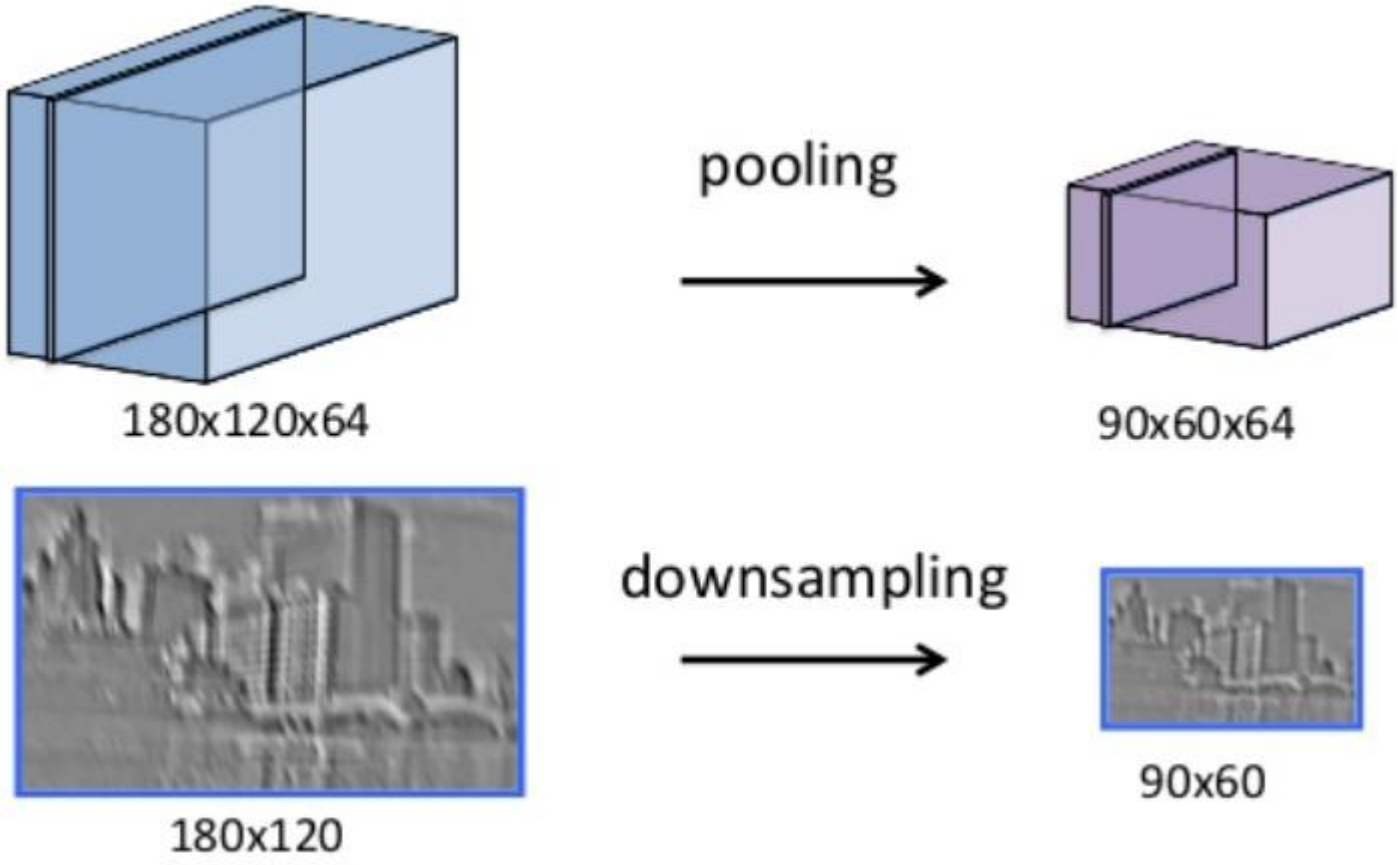


Смысл свёрток 1×1



Преобразование признаков!

Смысл пулинга

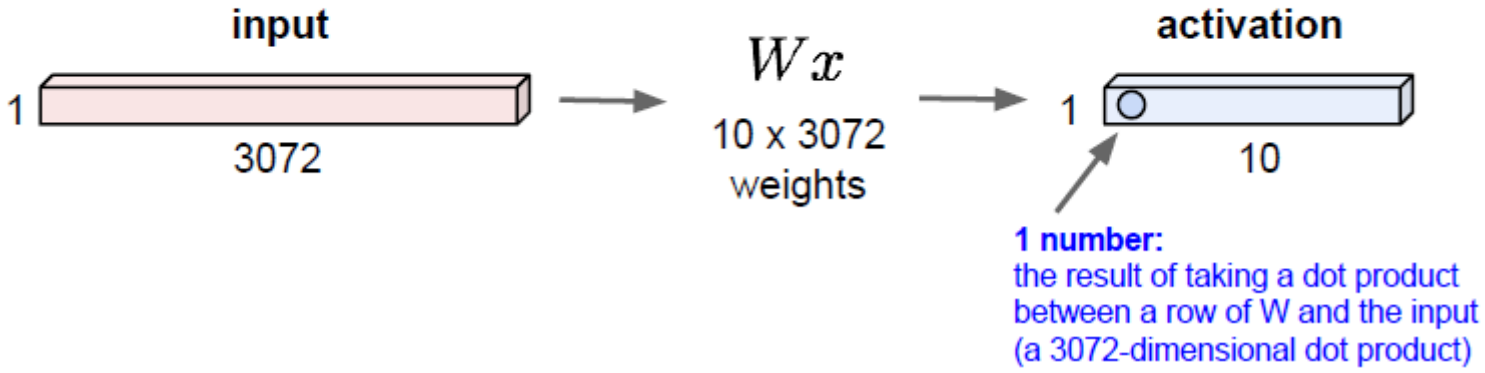


- **уменьшение размеров**
- **на каждой карте действует независимо**

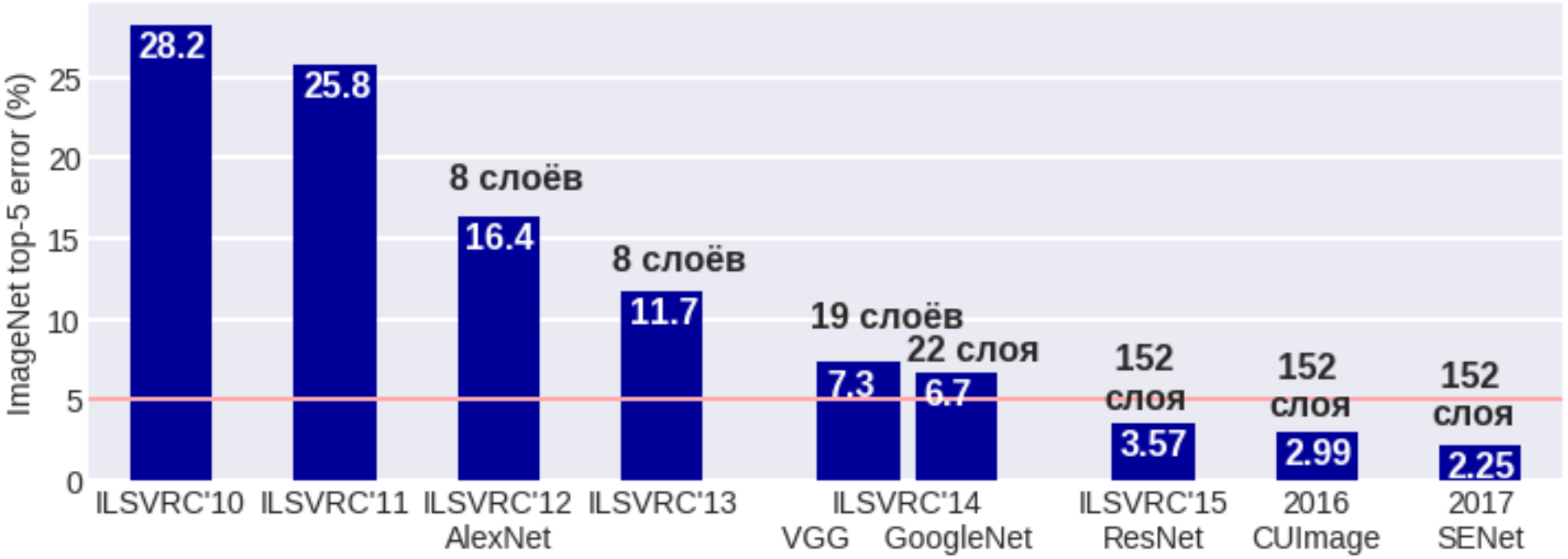
Полносвязный слой

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume

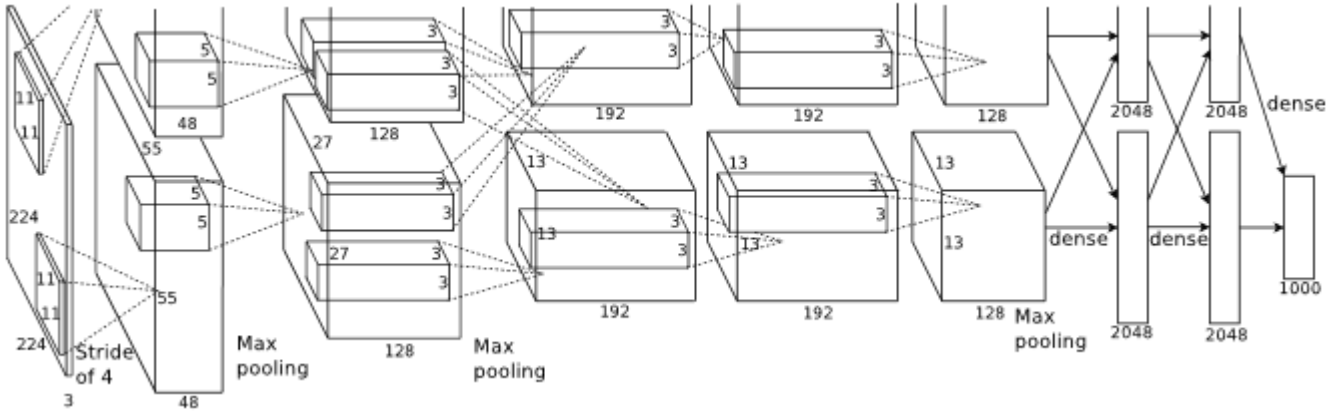


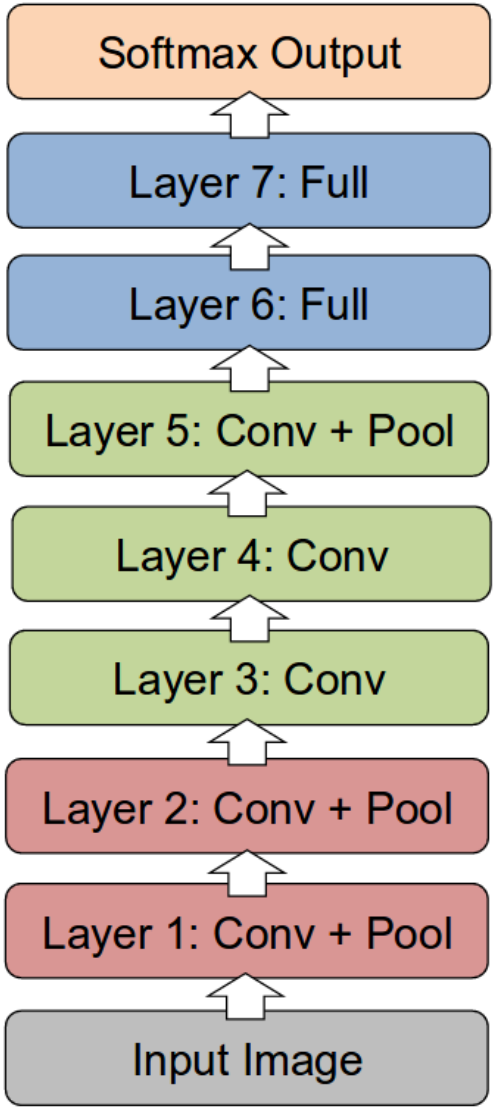
Революция в машинном обучении



ошибка человека – 5.1

AlexNet (2012)





- **ReLu**
- **Max-pooling**
- **Полно-связные слои**
- **Data augmentation**
- **Dropout 0.5**
- **Batch size = 128**
- **SGD Momentum 0.9**
- **60M параметров**
- **650K нейронов**
- **1 неделя на 2 GPU**
- **7 скрытых слоёв**

AlexNet (2012)

свёртки 3×3, 5×5, 11×11

7 CNN ансамбль: 18.2% → 15.4%

Интересно

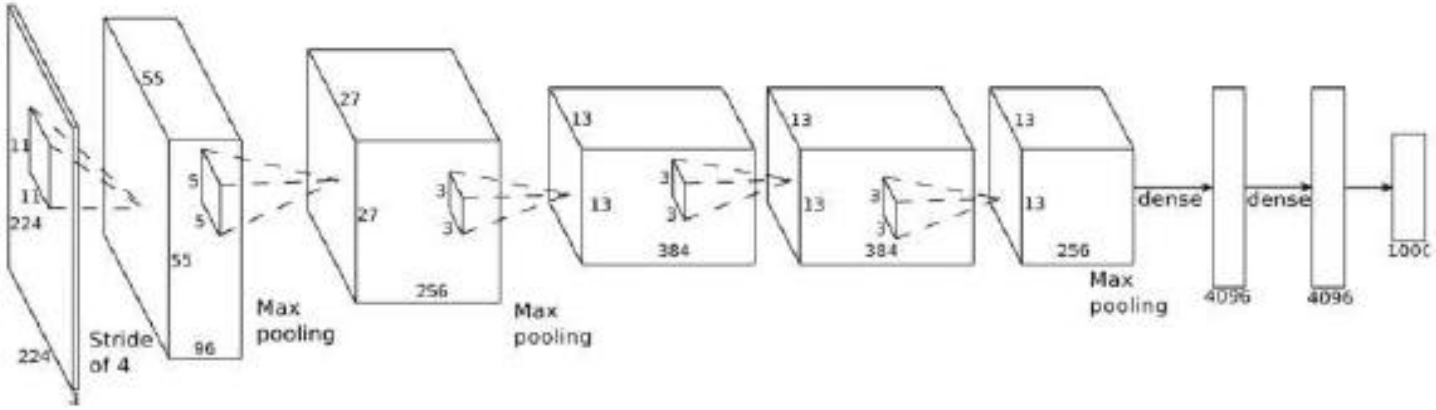
Убрать 16M параметров (последний полносвязный слой) – качество 1.1% ↓

Убрать 50M параметров (2 последних полносвязных слоя) – качество 5.7% ↓

Убрали 1M параметр (3 и 4 слои) – качество 3.0% ↓

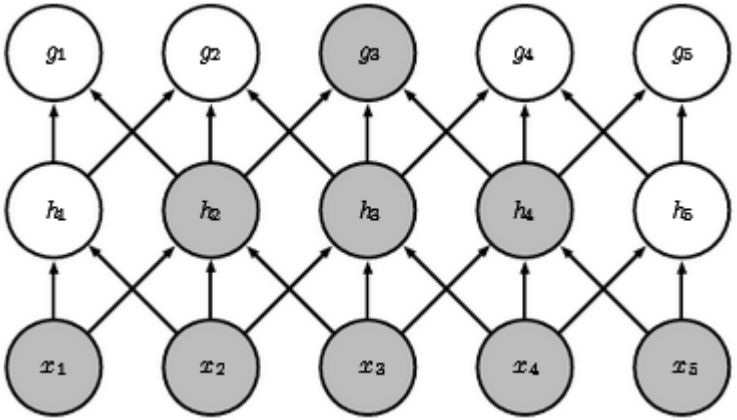
Убрать несколько слоёв (3, 4, 6, 7) – качество 33.5% ↓

VGG (2014)

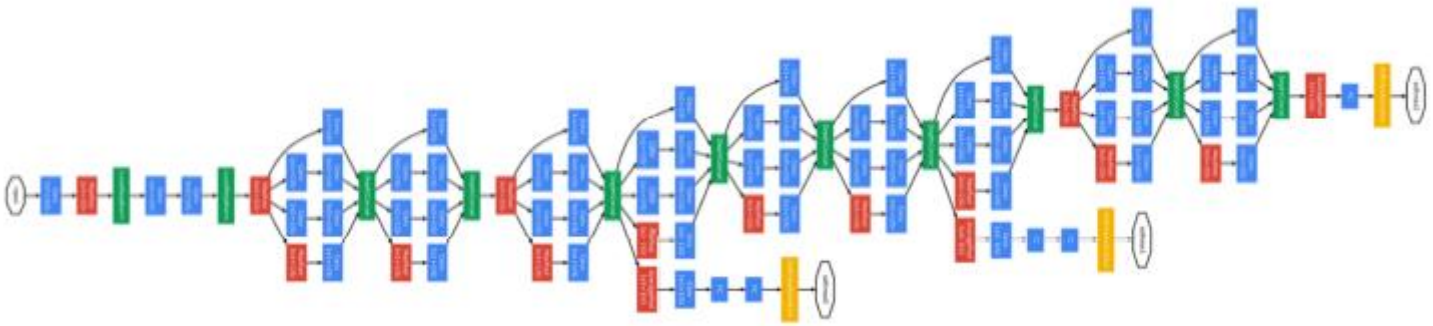


- **Каскад 3×3 свёрток (замена 7×7)**
- **Несколько стадий обучения**
- **138M параметров**
- **обучение на разных глубинах**
- **3 недели 4 GPU**
- **тоже использовали ансамбль**

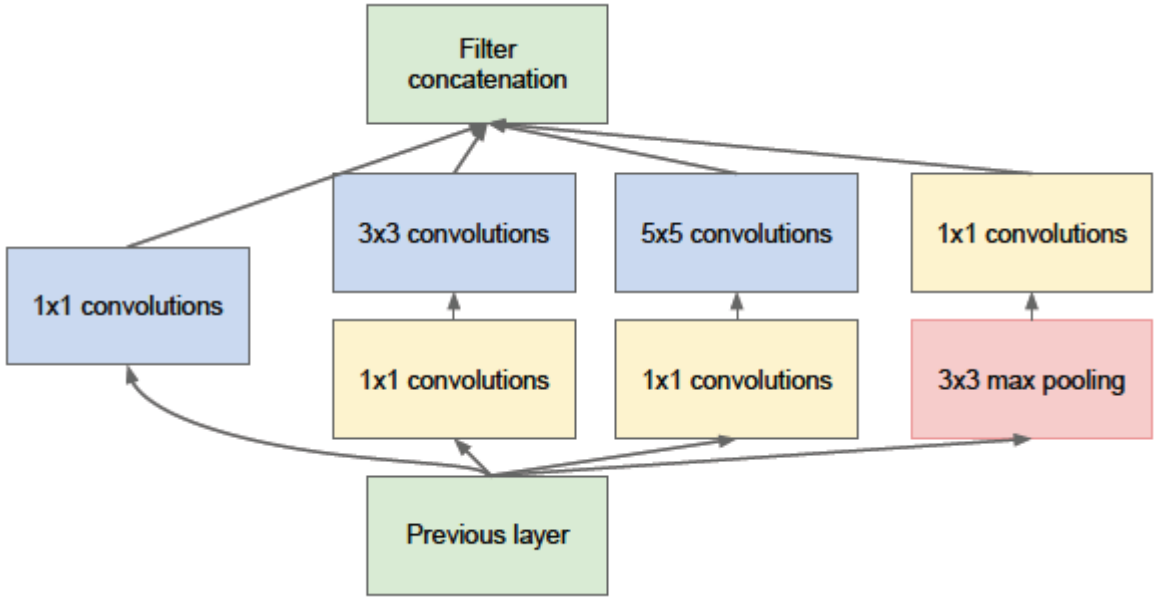
Идея каскада свёрток



GoogLeNet (2014)



- «конструктор» НС
- 22 слоя – нет полносвязных
- Модуль «Inception»
- 5M параметров (меньше!)
- дополнительные выходы классификации (для протекания градиента)



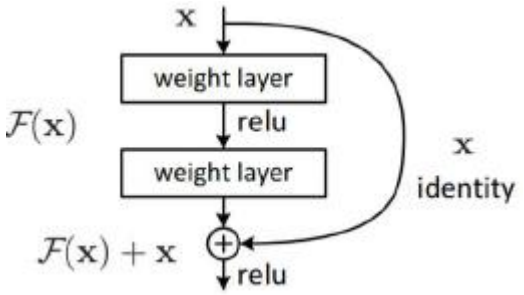
1×1-свёртки существенно уменьшают число параметров!

… а идея была (синие блоки) – разные свёртки + пулинг

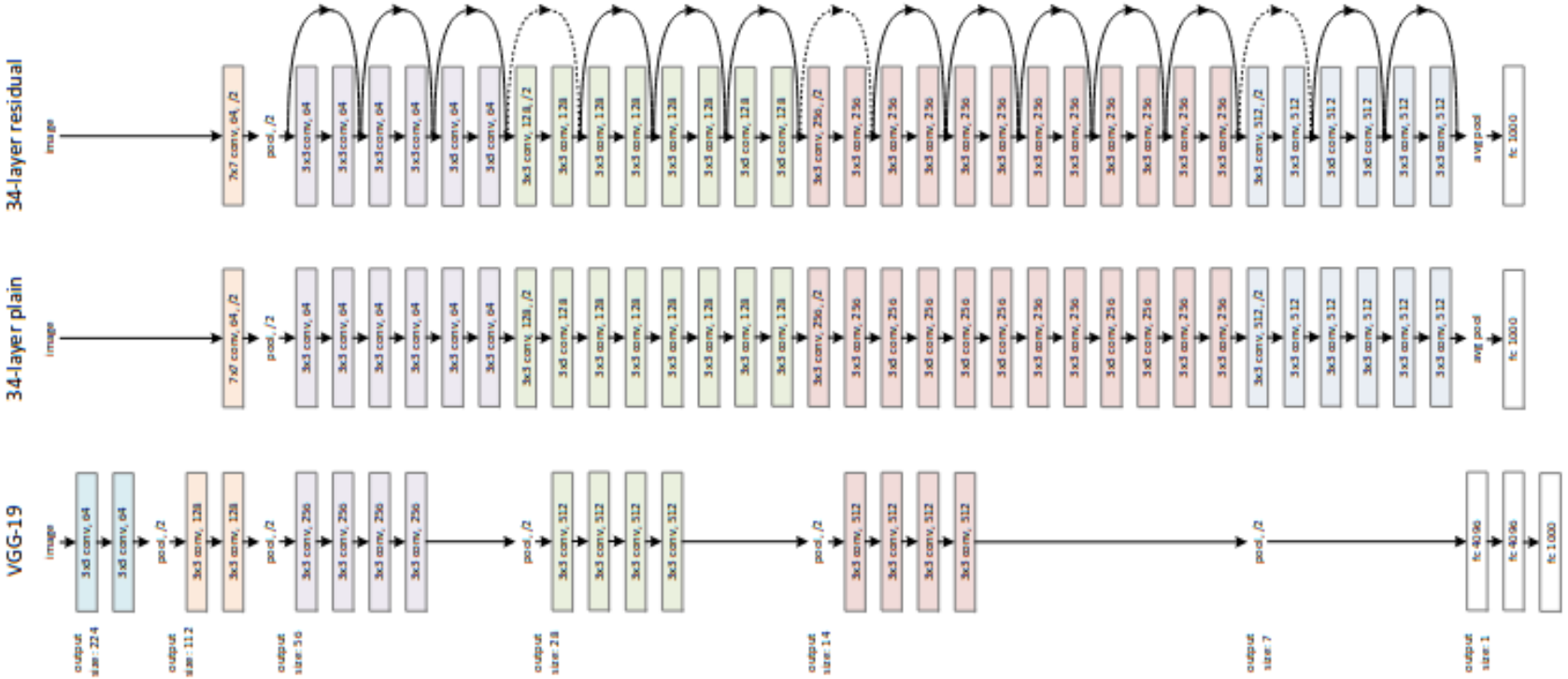
ResNet = Residual Network (2014)

$$y = f(x) + x$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} f'(x) + \frac{\partial L}{\partial y}$$



He, Zhang, Ren, Sun, CVPR 2016



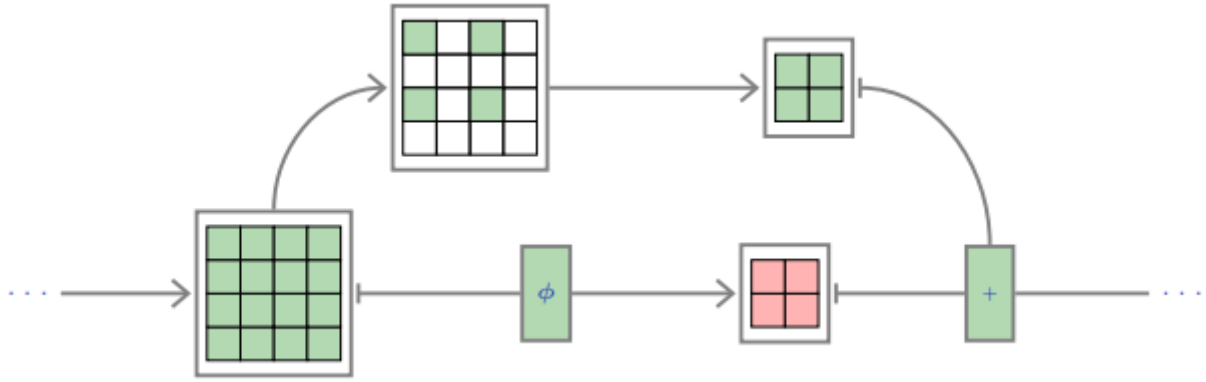
ResNet = Residual Network (2014)

- 152 слоя
- связи проходят через слои
- Batch Normalization после каждого CONV-layer
- Умные инициализации весов
- SGD + Momentum (0.9)
- Mini-batch size = 256
- Нет Dropout!

Просто добавление слоёв не помогает!

Добавлять надо по-умному…

Проблемы с прокидыванием в свёрточных слоях



1) размеры уменьшаются, поэтому не совсем прямая связь

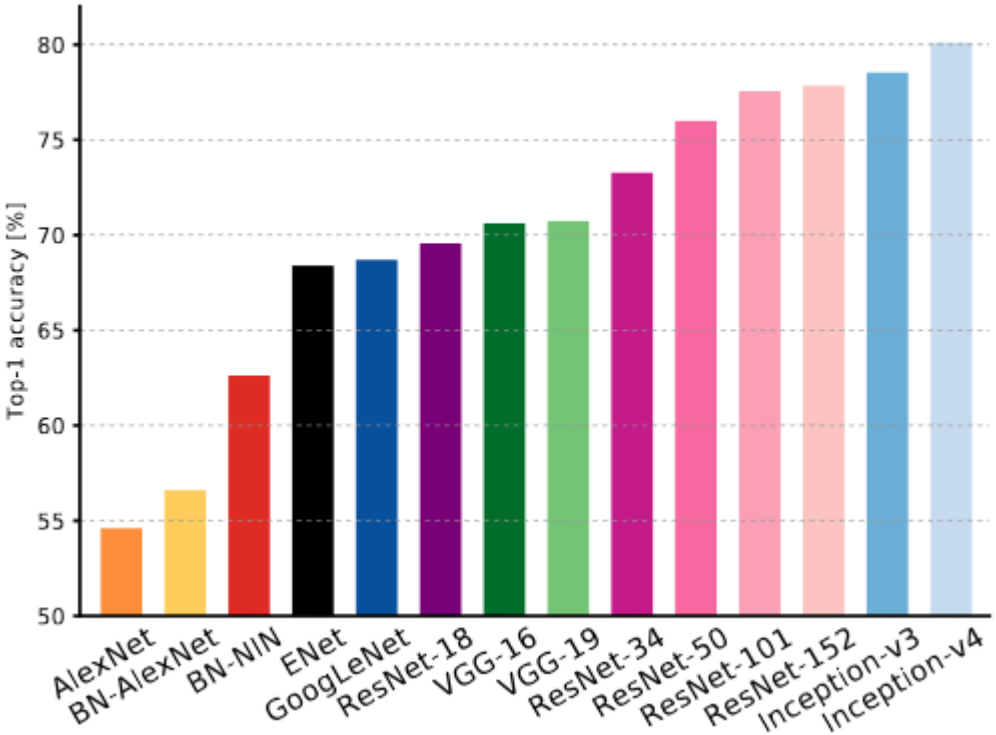
2) изменяется число каналов!

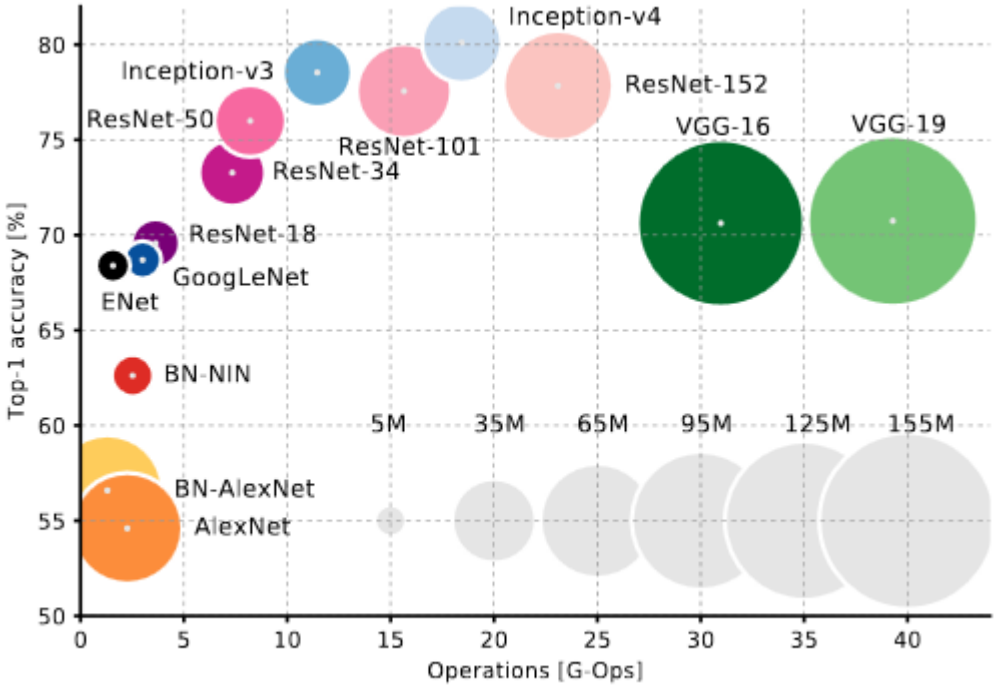
добавить нули

есть ещё способ… ;)

https://fleuret.org/ee559/

Inception-v4: Resnet + Inception!





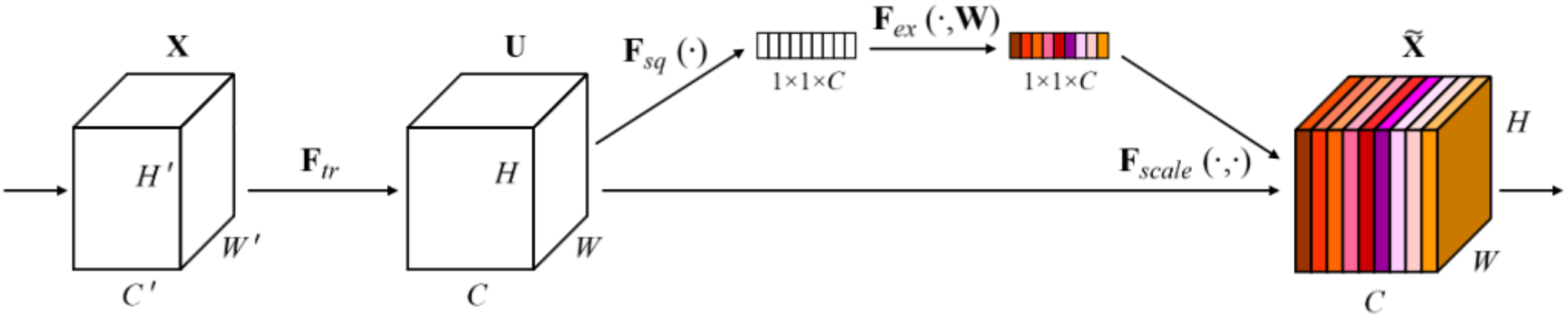
A. Canziani, A. Paszke, E. Culurciello, « An Analysis of Deep Neural Network Models for Practical Applications», 2017 https://arxiv.org/pdf/1605.07678.pdf

SENet (Squeeze-and-Excitation Network) 2017

Раньше: трансформация $F_{\text{tr}} : X_{H' \times W' \times C'} \to U_{H \times W \times C}$

(например, свёртка)

Теперь: «Squeeze-and-Excitation» (SE) block $F_{\text{tr}} \oplus \ldots$



сжатие (squeeze) – агрегация по каналам

$$F_{\text{sq}} : \| u_{h,w,c} \|_{H \times W \times C} \to \left\| \frac{1}{HW} \sum_{w=1}^{W} \sum_{h=1}^{H} u_{h,w,c} \right\|_{C}$$

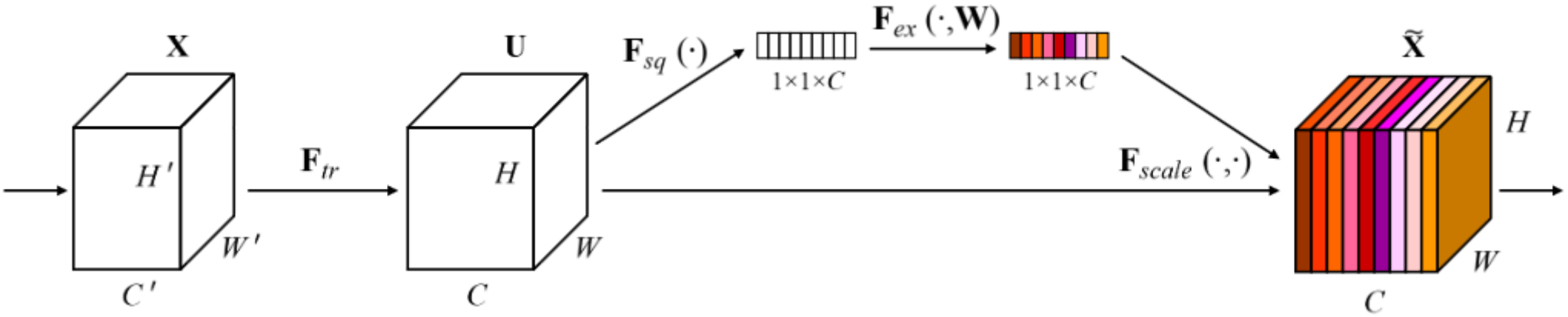
J. Hu и др. «Squeeze-and-Excitation Networks», 2018 https://arxiv.org/pdf/1709.01507.pdf

SENet (Squeeze-and-Excitation Network) 2017

Раньше: трансформация $F_{\text{tr}} : X_{H' \times W' \times C'} \to U_{H \times W \times C}$

(например, свёртка)

Теперь: «Squeeze-and-Excitation» (SE) block $F_{\text{tr}} \oplus \ldots$



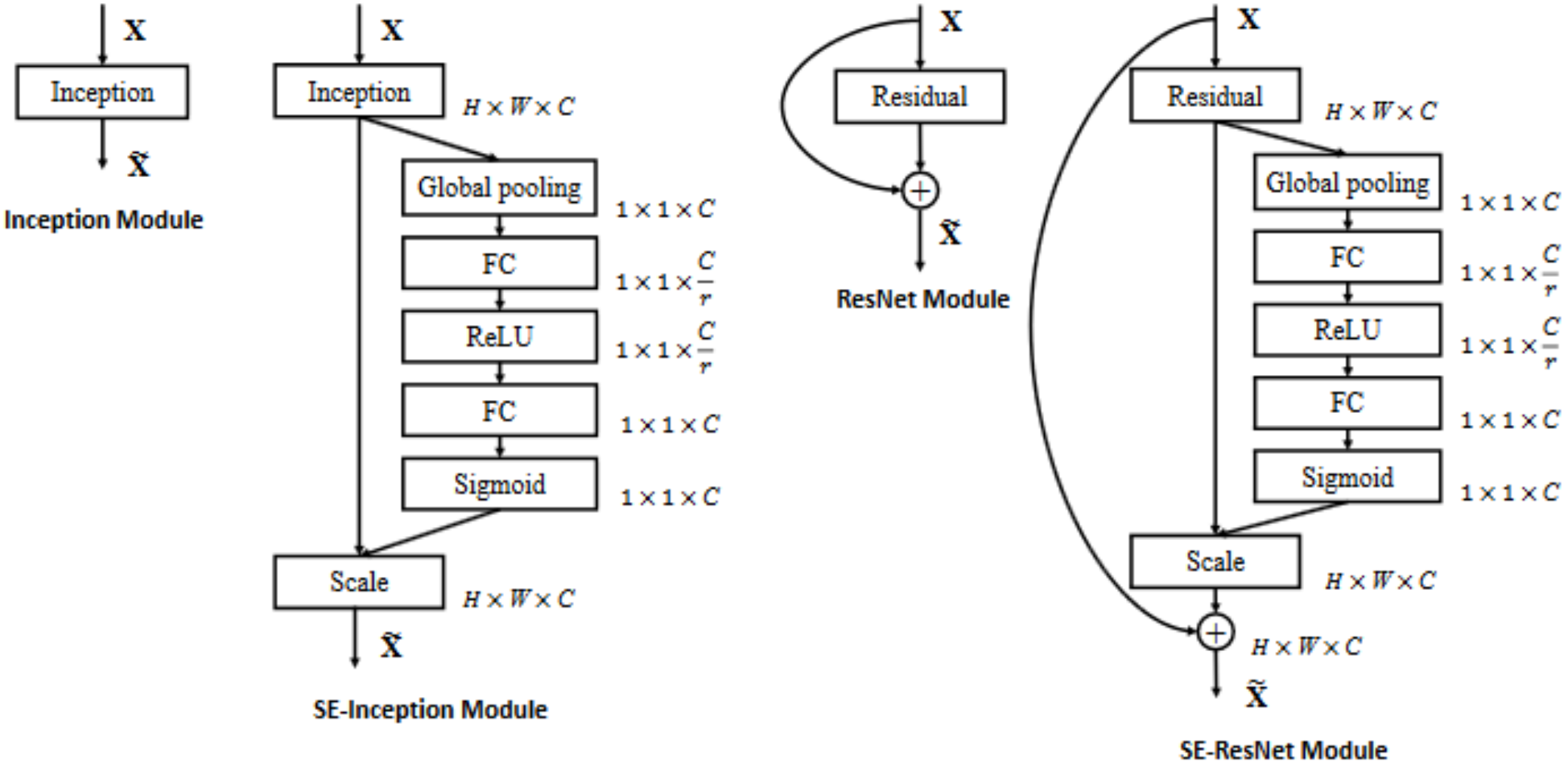
возбуждение (excitation) – адаптивная калибровка

$$F_{\text{ex}} = \sigma(W_{C \times k}\, \text{ReLu}(V_{k \times C} z_C))$$

$$F_{\text{scale}} : \| u_{h,w,c} \|_{H \times W \times C} \to \left\| u_{h,w,c} F_{\text{ex}}(z)_c \right\|_C$$

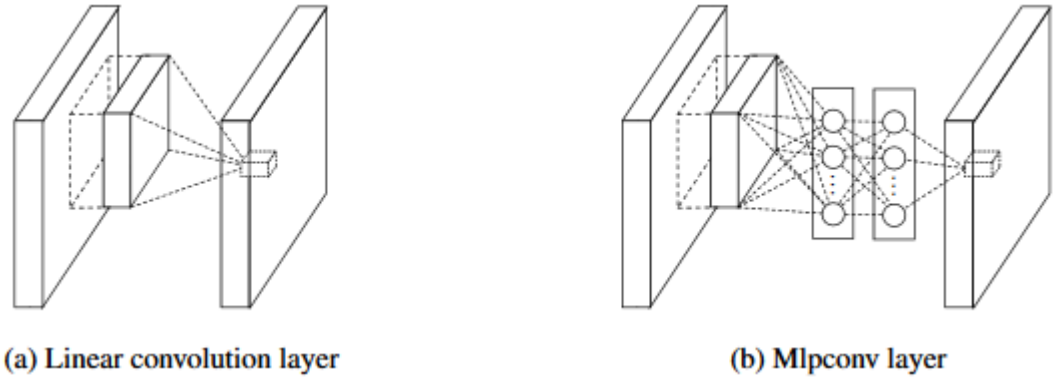
SENet (Squeeze-and-Excitation Network) 2017

Можно переделать «старые сети»

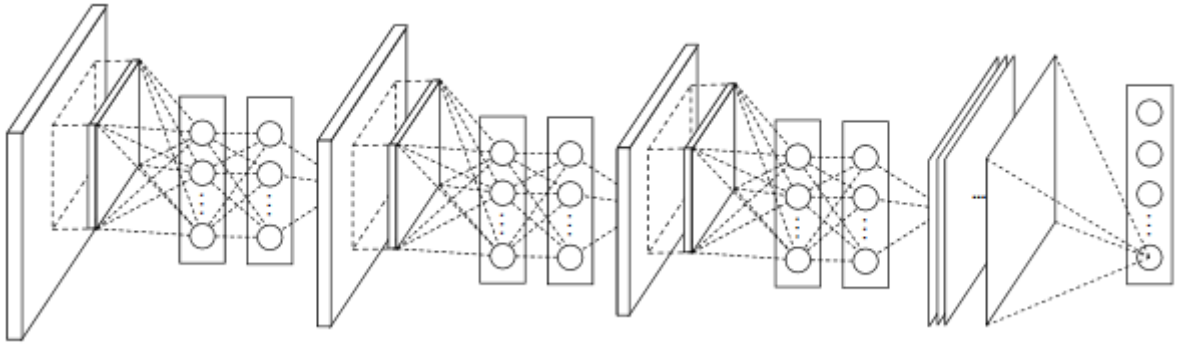


Динамическая перекалибровка признаков позволяет «увеличивать» важные признаки и «уменьшать» неважные

Какие архитектуры ещё надо знать…
Network in Network (NiN)



(a) Linear convolution layer (b) Mlpconv layer



полносвязность ~ свёртки 1×1 внутри свёртки

глобальный пулинг

Min Lin « Network in Network (NiN)» 2014, https://arxiv.org/pdf/1312.4400.pdf

Deep Networks with Stochastic Depth

- **Во время обучения: случайно удаляем подмножество слоёв**
  (используем менее глубокую сеть во время обучения)

- **«Прокидывание» тождественной функции**

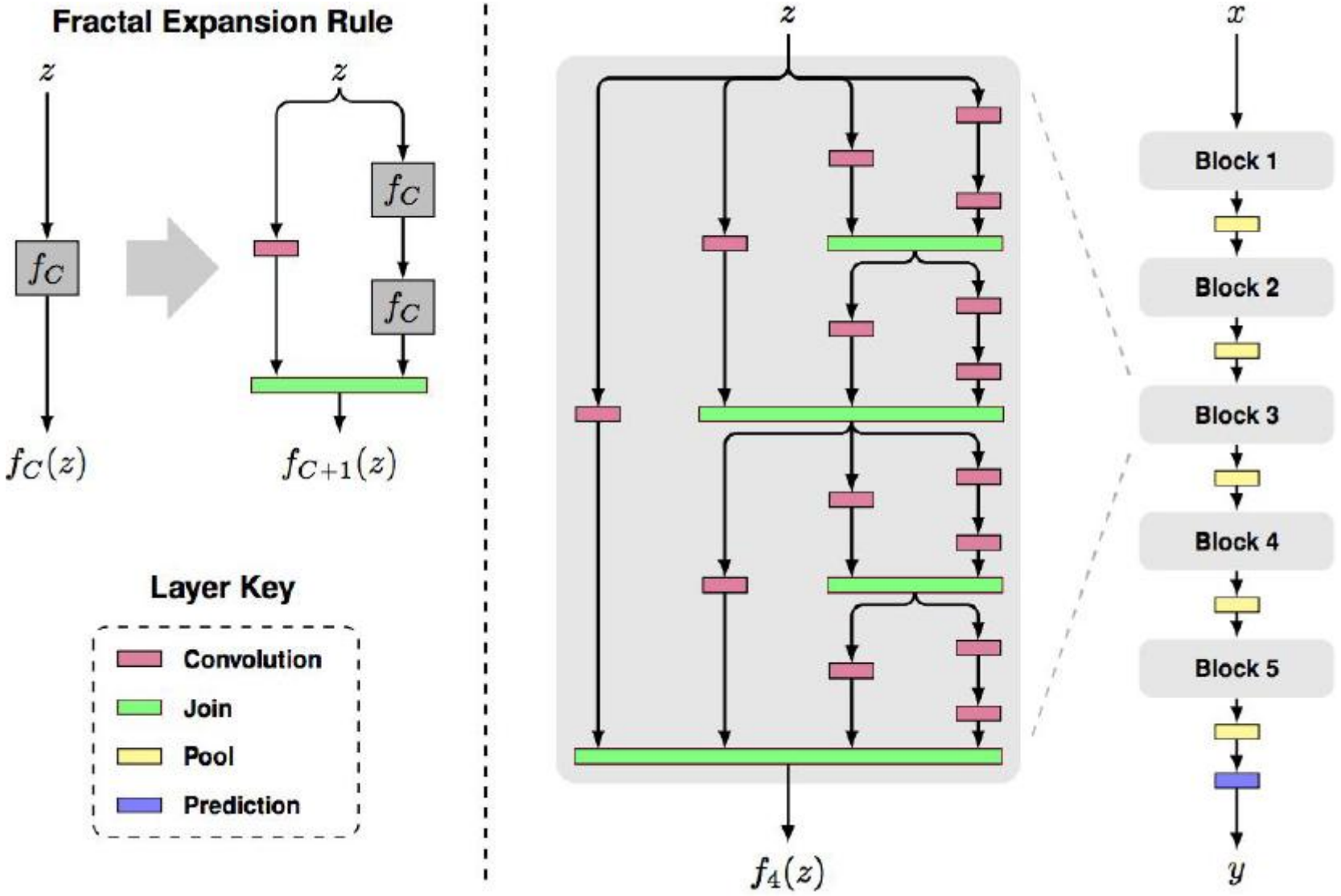
[Gao Huang, 2016 https://arxiv.org/abs/1603.09382]

FractalNet: Ultra-Deep Neural Networks without Residuals

Фрактальная архитектура с короткими и длинными связями
Обучение со случайным выбрасыванием связей

[Gustav Larsson 2017 https://arxiv.org/abs/1605.07648]

Фрактальные сети



Densely Connected Convolutional Networks (DenseNets)

Блоки в которых слой соединён с каждым последующим

[Gao Huang 2016 https://arxiv.org/abs/1608.06993]

Обычная сеть: $z_i = H_i(z_{i-1})$

где z_i **выход i-го слоя.**

ResNet: $z_i = H_i(z_{i-1}) + z_{i-1}$

DensNet: $z_i = H_i([z_{i-1}, z_{i-2}, \ldots, z_0])$

H = BN + ReLU + convolution + dropout

число признаков линейно вырастает…

ResNeXt

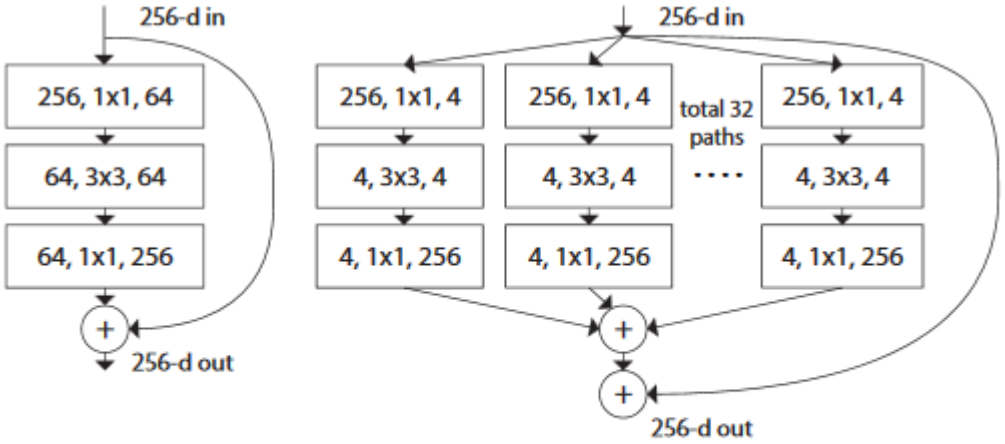


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

такое же число параметров как в ResNet,

но разнести их по 32 разным путям

тут блок – conv + ИТ + ReLU

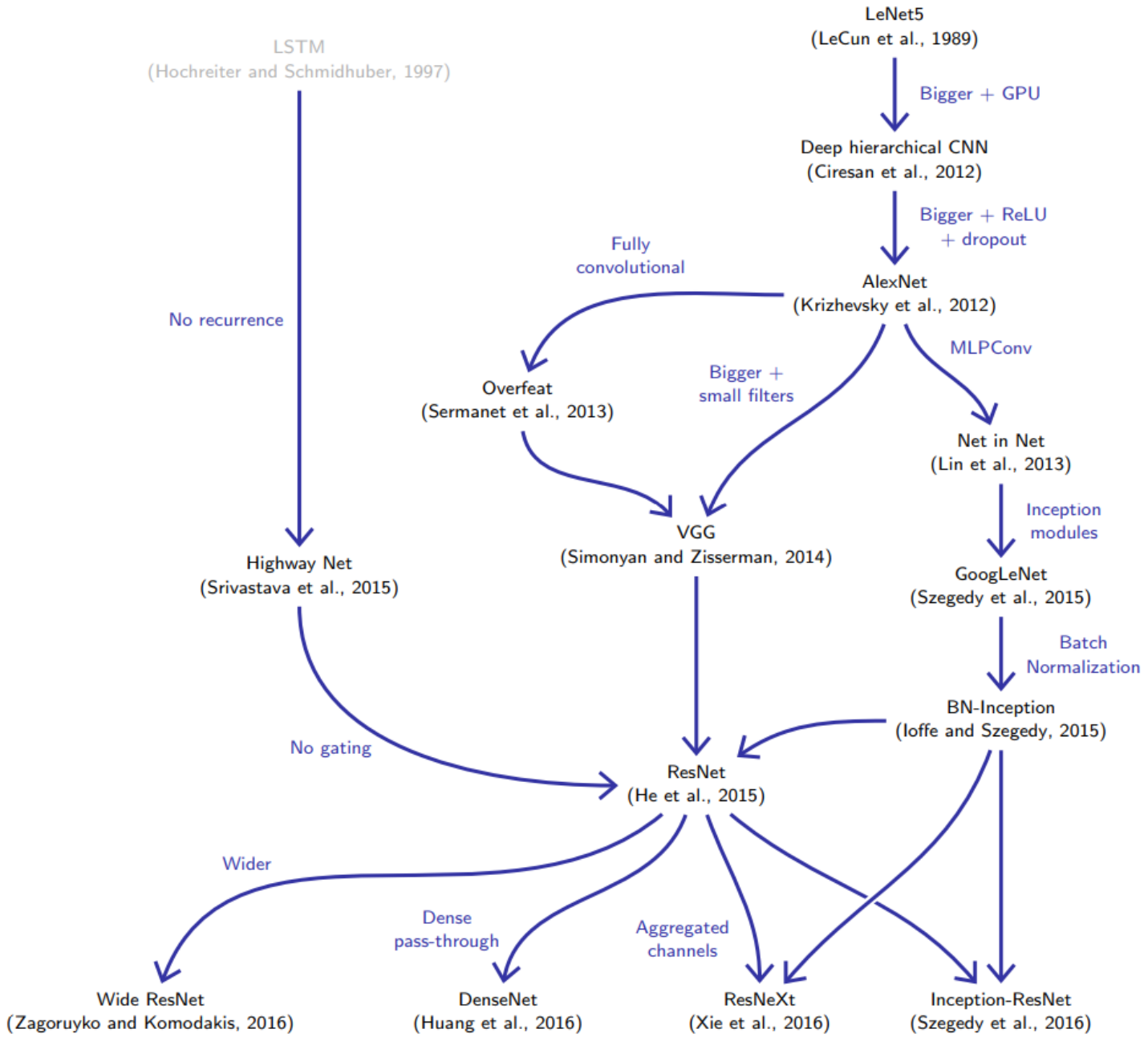
используется bottleneck!!!

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He
Aggregated Residual Transformations for Deep Neural Networks //
https://arxiv.org/abs/1611.05431

SqueezeNet – Сжатие AlexNet

сеть	подход	размер	отношение	Top1-точность	Top5-точность
AlexNet		240MB	1x	57.2%	80.3%
AlexNet	SVD	48MB	2x	56.0%	79.4%
AlexNet	Deep Compression	7MB	35x	57.2%	80.3%
SqueezeNet		5MB	50x	57.5%	80.3%
SqueezeNet	Deep Compression	0.5MB	510x	57.5%	80.3%

Forrest N. Iandola « SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size»2017 https://arxiv.org/abs/1602.07360



https://fleuret.org/ee559/ee559-slides-7-2-image-classification.pdf

За чем можно наблюдать в NN

- параметры: фильтры как картинки
- внутренние активации – как картинки
- распределения активаций (на отдельных объектах)
- производные по входу
- входы, максимизирующие какой-то ответ
- «adversarial samples»

Визуализация – как понять, чему сеть учится…

Последний полносвязный слой

Можно смотреть соседей в этом признаковом пространстве

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Визуализация – как понять, чему сеть учится…

Последний полносвязный слой

Применить уменьшение размерности… t-SNE в R^2

https://cs.stanford.edu/people/karpathy/cnnembed/

Визуализация – как понять, чему сеть учится…

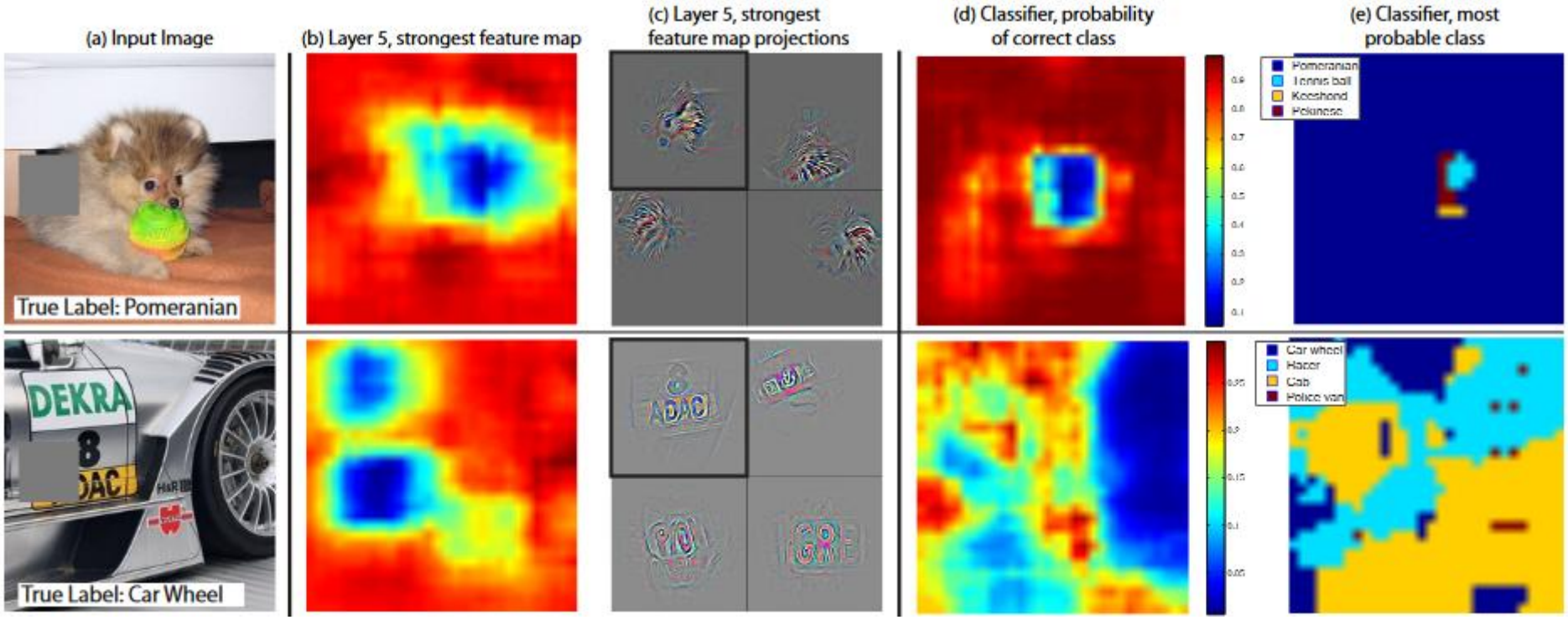
Средние слои

На каких изображениях максимальные значения активаций

https://arxiv.org/pdf/1412.6806.pdf

Визуализация – как понять, чему сеть учится…

Какие пиксели отвечают за класс – «Occlusion sensitivity»



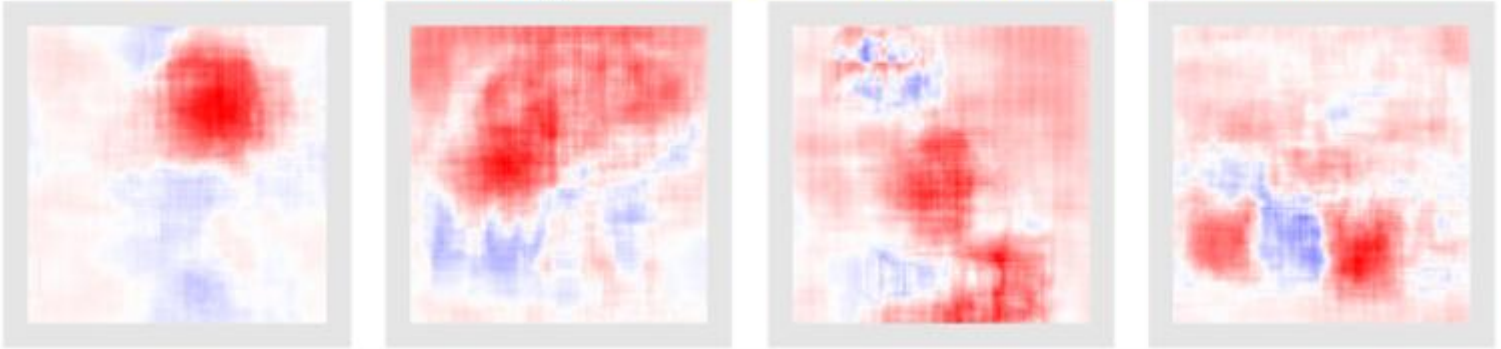
Закрывать последовательно часть изображения – 2D-гистограмма вероятности принадлежности к заданному классу при закрытии с центром в ij-м пикселе Matthew D Zeiler, Rob Fergus «Visualizing and Understanding Convolutional Networks» https://arxiv.org/pdf/1311.2901.pdf

Визуализация – как понять, чему сеть учится...

Original images

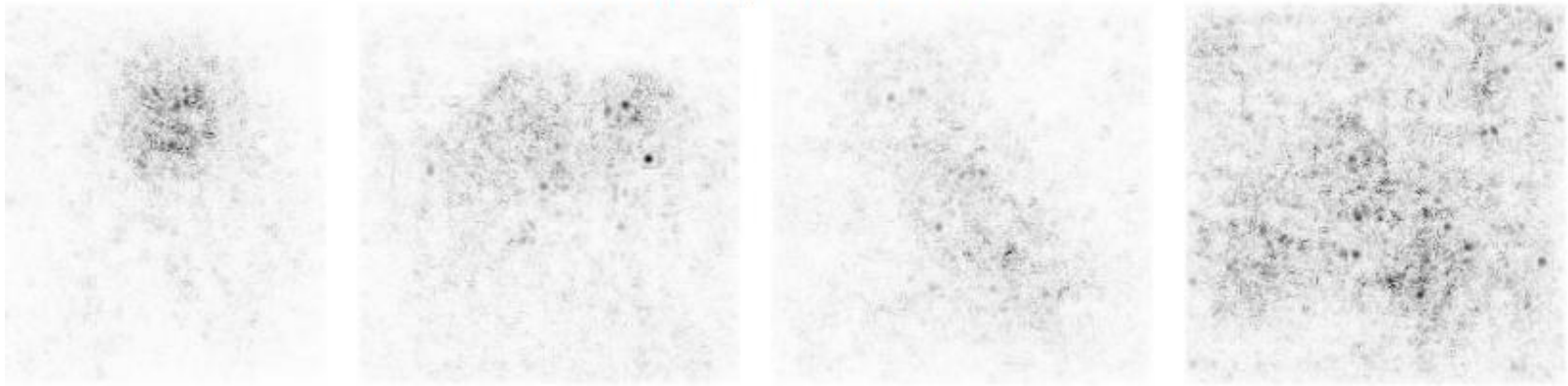
Occlusion mask 32 × 32

Occlusion sensitivity, mask 32 × 32, stride of 2, AlexNet

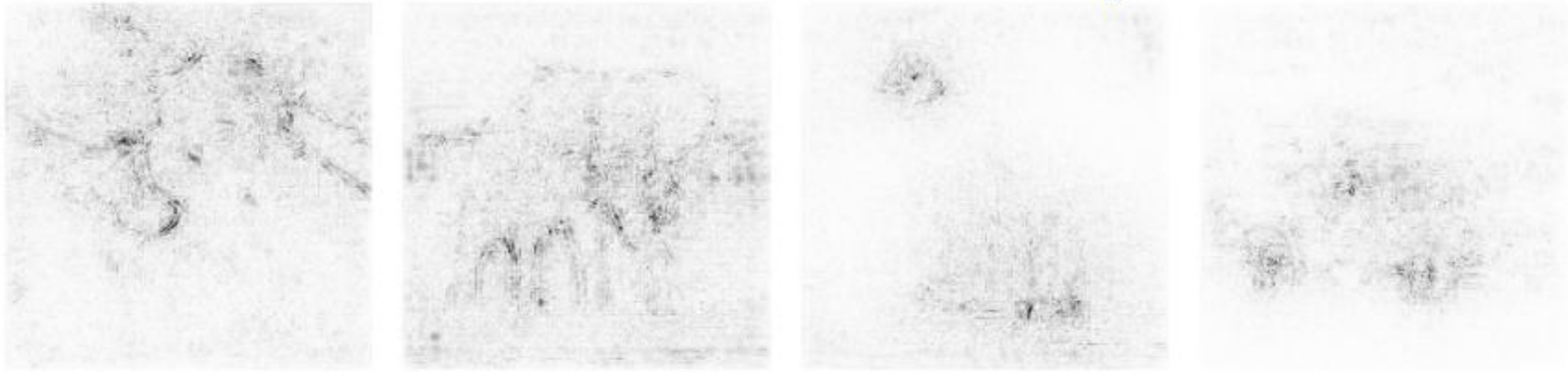


«Saliency maps» – градиенты по входу

Gradient, AlexNet

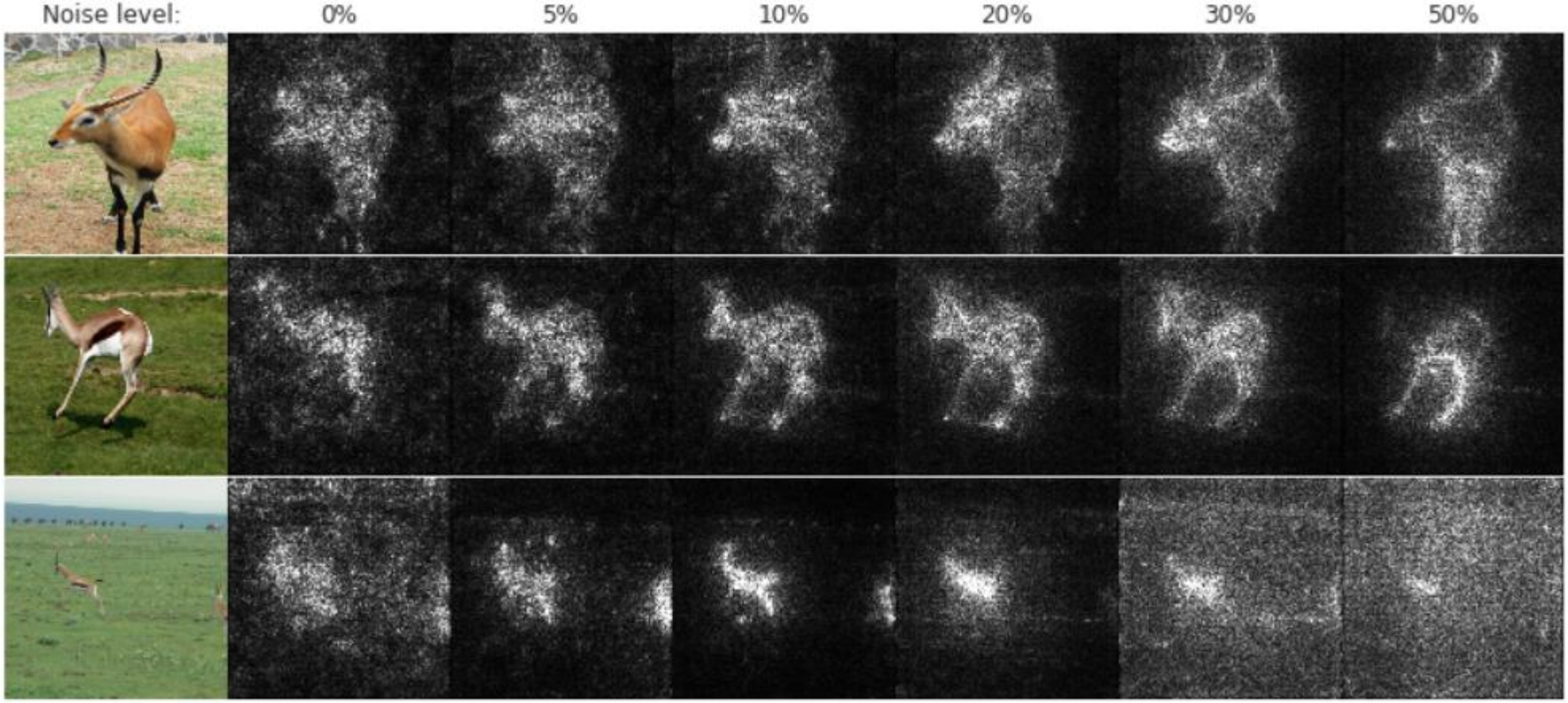


SmoothGrad, AlexNet, $\sigma = \frac{\Delta}{4}$



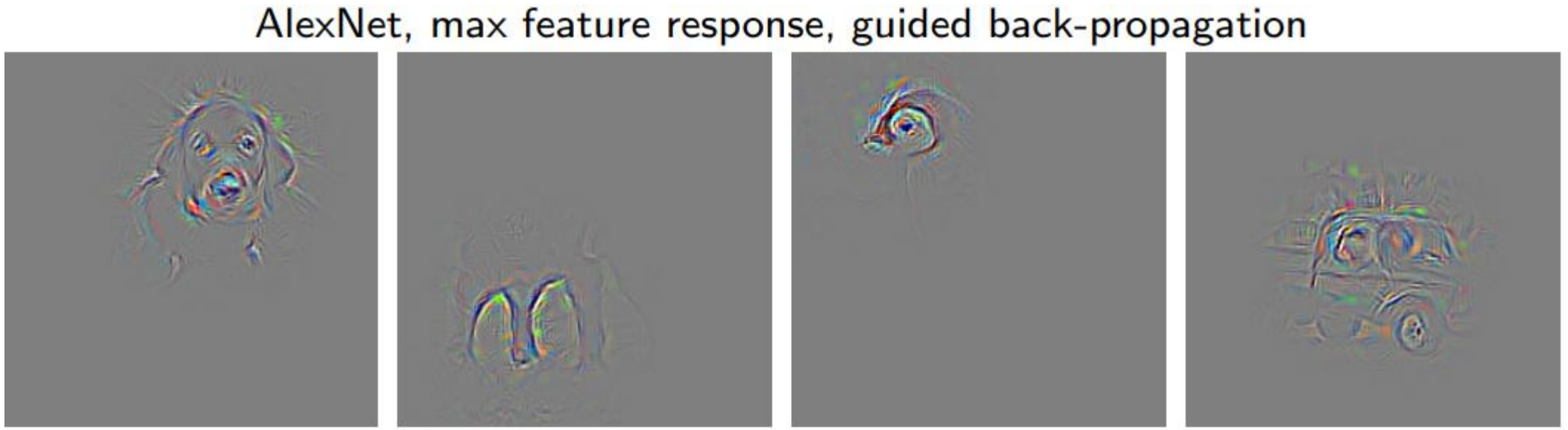
«Saliency maps» – градиенты по входу

с помощью шума можно получать более адекватные иллюстрации



Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, Martin Wattenberg «SmoothGrad: removing noise by adding noise» // https://arxiv.org/abs/1706.03825

«guided back-propagation» ~ градиенты по входу

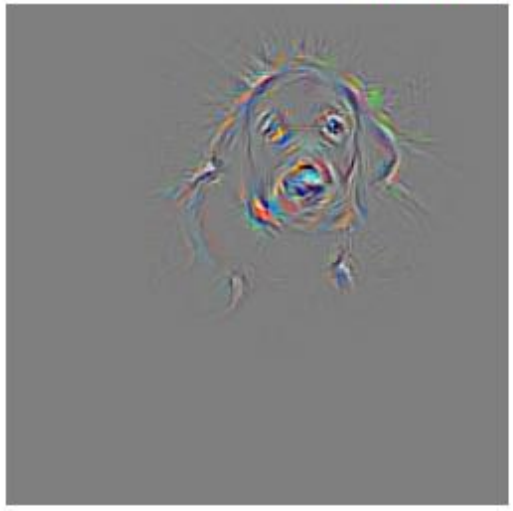
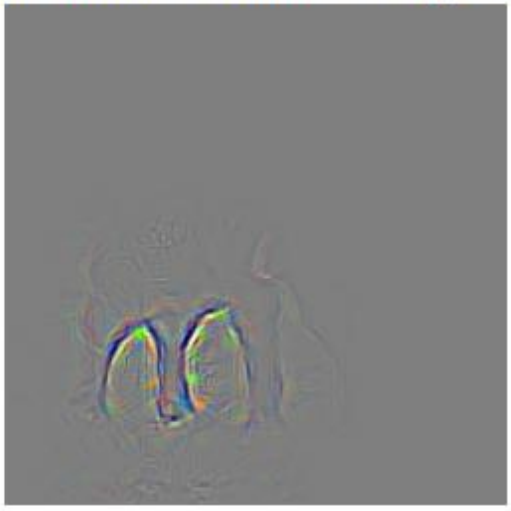
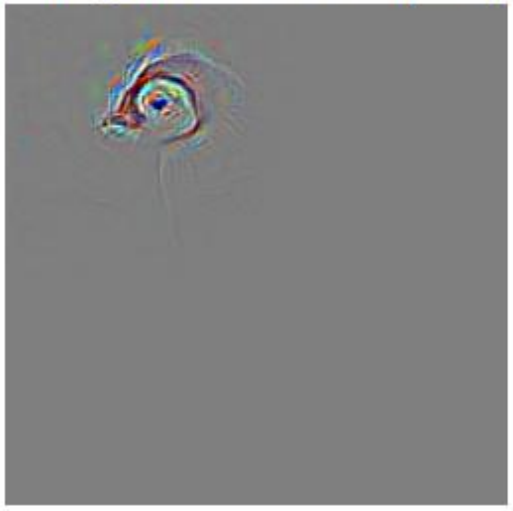


Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller «Striving for Simplicity: The All Convolutional Net» // https://arxiv.org/abs/1412.6806

Original images




AlexNet, max feature response, guided back-propagation

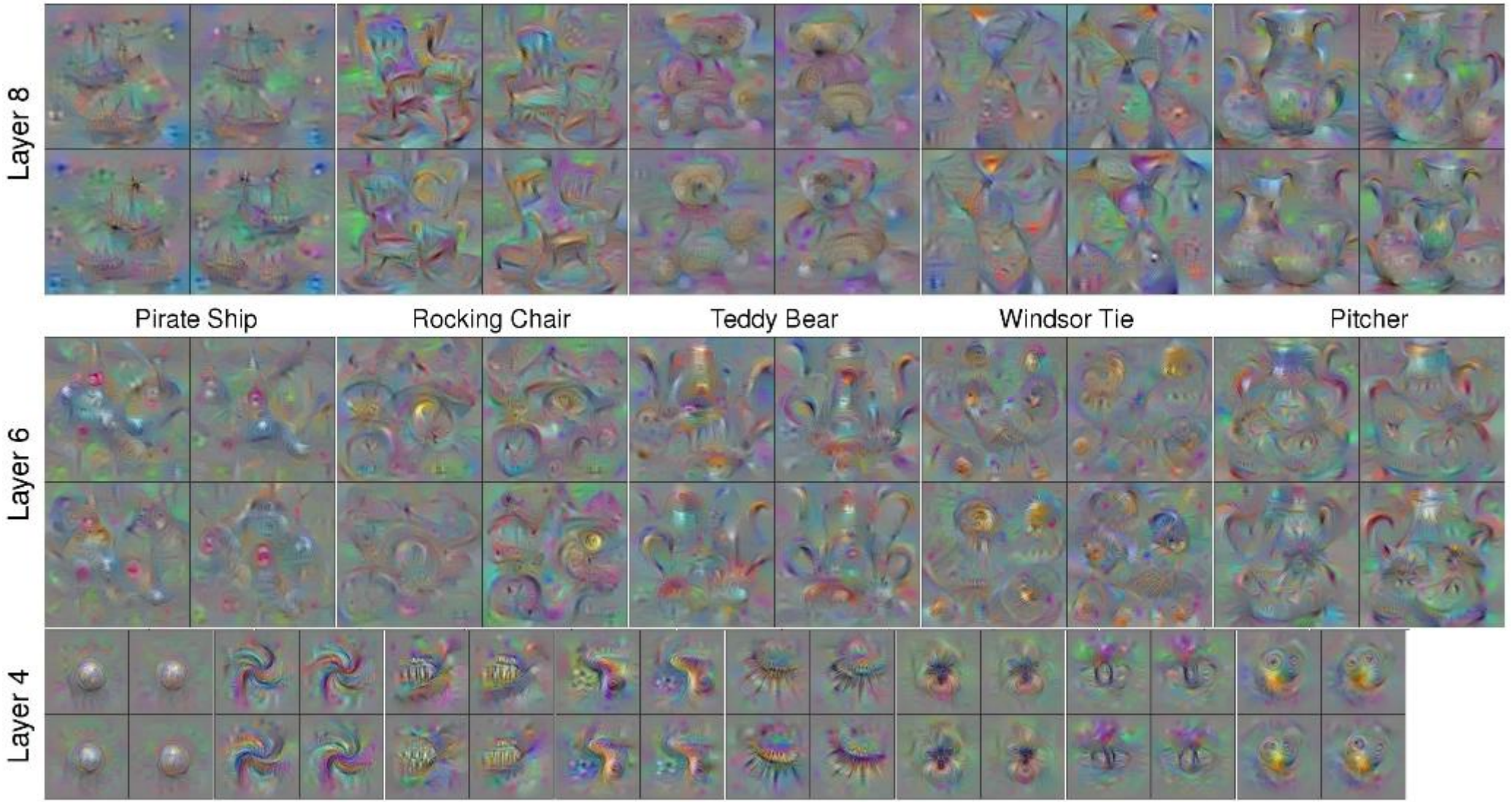

Визуализация – как понять, чему сеть учится…

Отдельные нейроны

Сгенерировать изображения,
которые максимизируют активацию выделенного нейрона
(методом обратного распространения ошибки)

Визуализация – как понять, чему сеть учится…

Отдельные нейроны



Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, Hod Lipson «Understanding Neural Networks Through Deep Visualization» // https://arxiv.org/pdf/1506.06579.pdf

Что потом обсудим

- **Решение задач компьютерного зрения с помощью DL**
- **Обманные изображения (Fooling Images)**