курс «Глубокое обучение»

Архитектуры нейронных сетей

Александр Дьяконов

22 февраля 2022 года

План

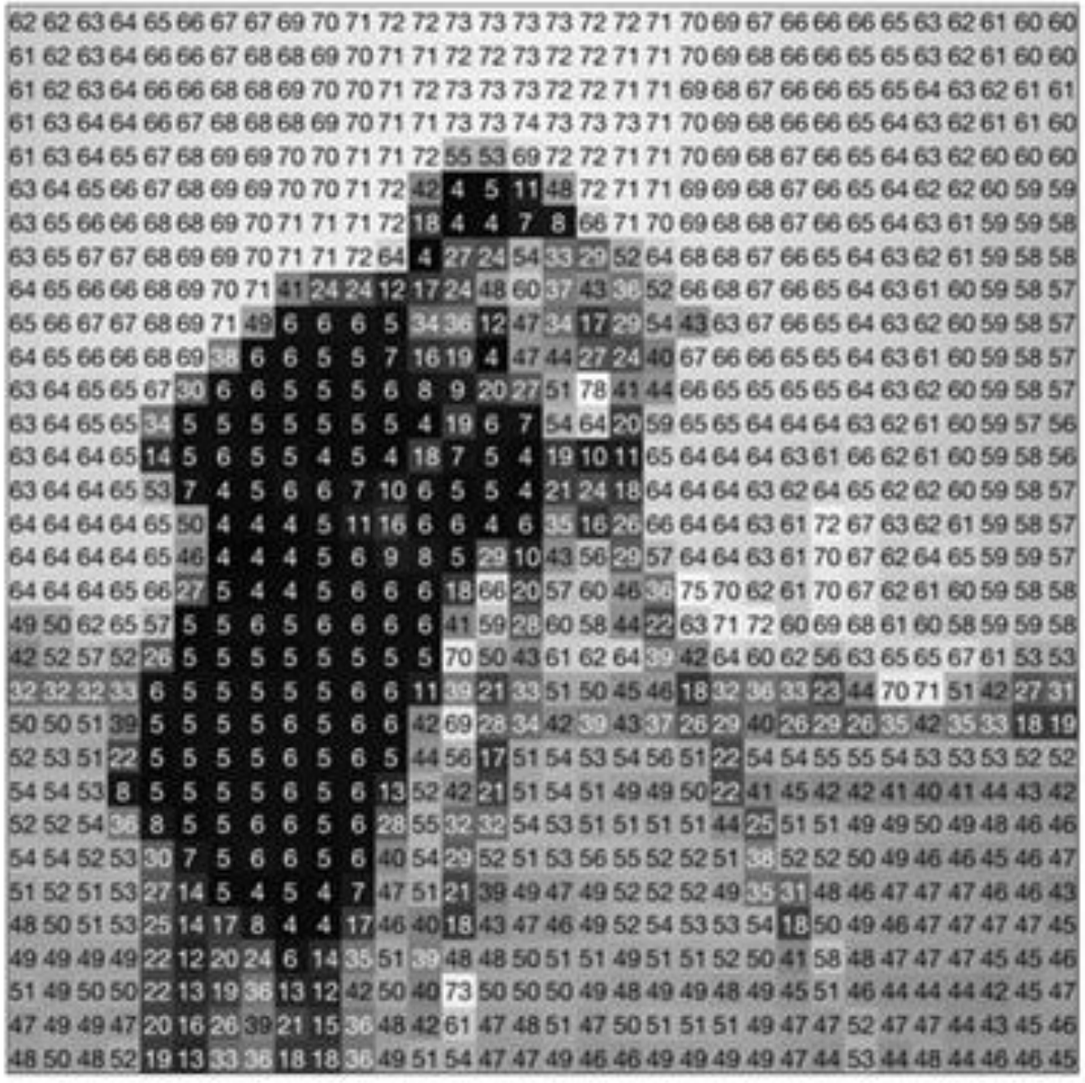
Свёрточные сети

Кодировщик-декодировщик

Рекуррентные сети

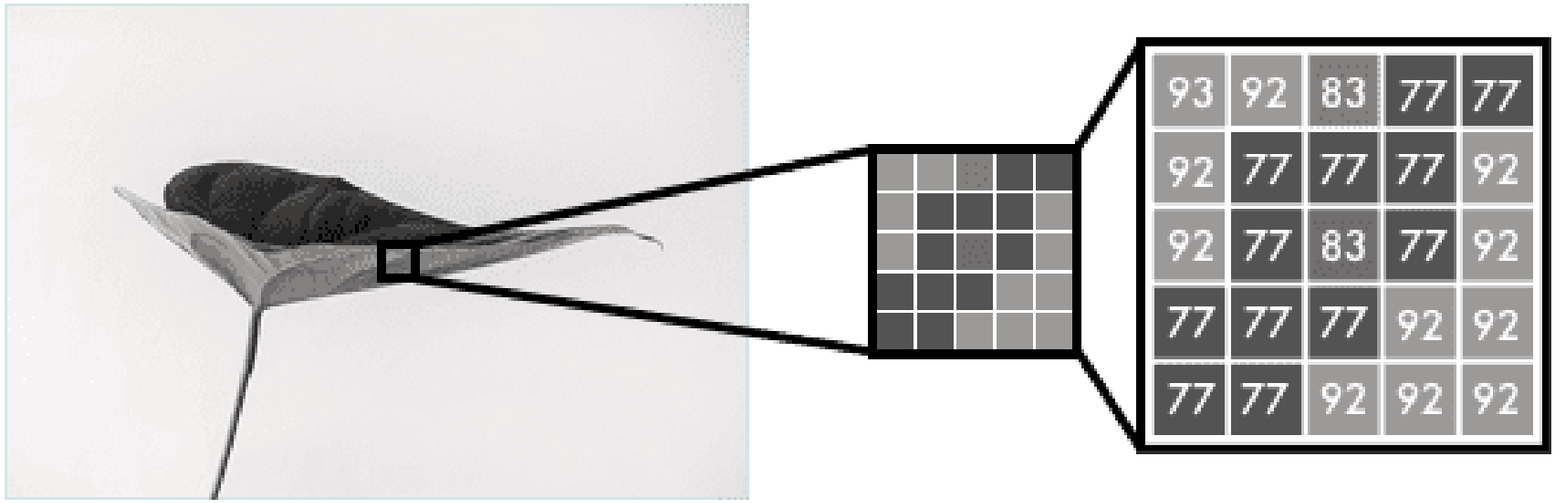
Внимание

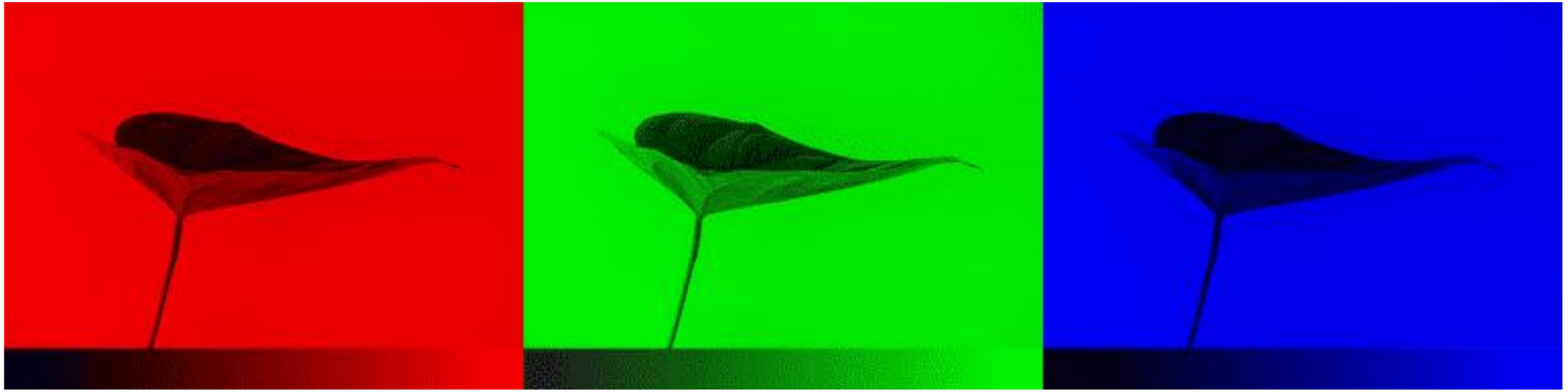
Что такое изображение – H×W-матрица



«чёрно-белое» (в градациях серого) – целочисленная матрица

Что такое изображение – трёхмерный C×H×W -тензор





цветное – 3-х мерная целочисленная матрица (тензор)

2-D свёртка (Convolution)

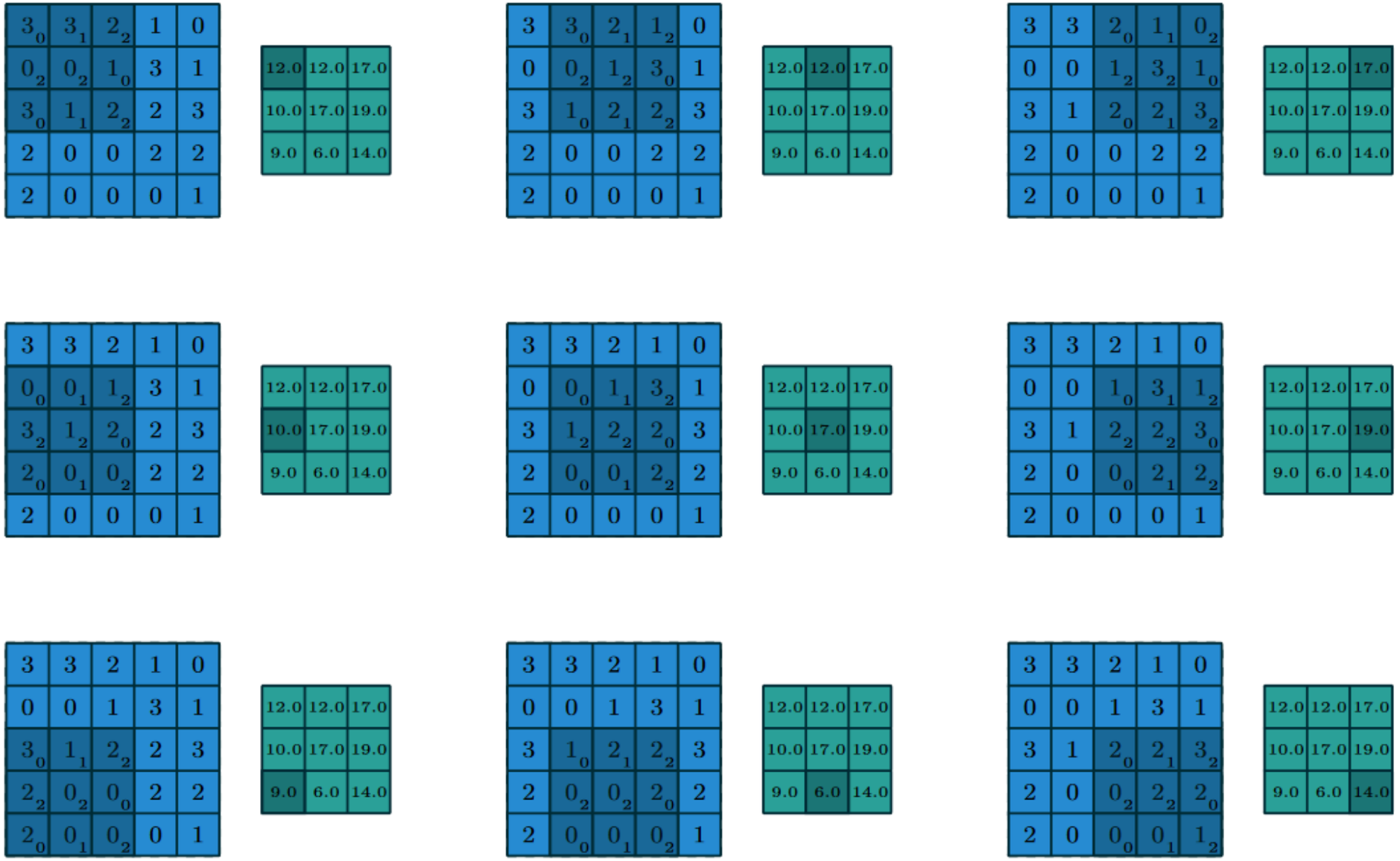
$$(I * K)_{xy} = \sum_{i=1}^{h}\sum_{j=1}^{r} K_{ij} I_{x+i-1,y+j-1}$$

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

может быть немного другая индексация

хорошее объяснение: Vincent Dumoulin, Francesco Visin «A guide to convolution arithmetic for deep learning» https://arxiv.org/pdf/1603.07285.pdf

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?

исследует локальные участки изображения – ищет паттерны

Что делает свёртка?

Фильтры в CV

- устраняют шум
- находят границы
- детектируют текстуры

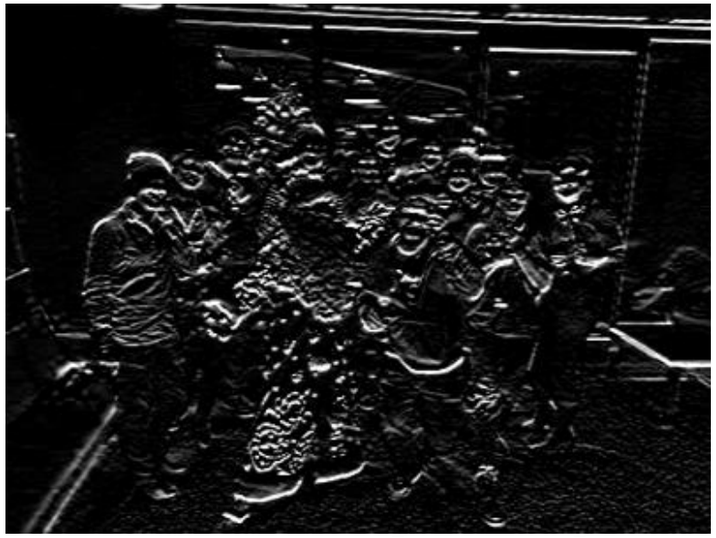
оригинал

blur

$$\begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix}$$

sharpen

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



top sobel $\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

left sobel $\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$

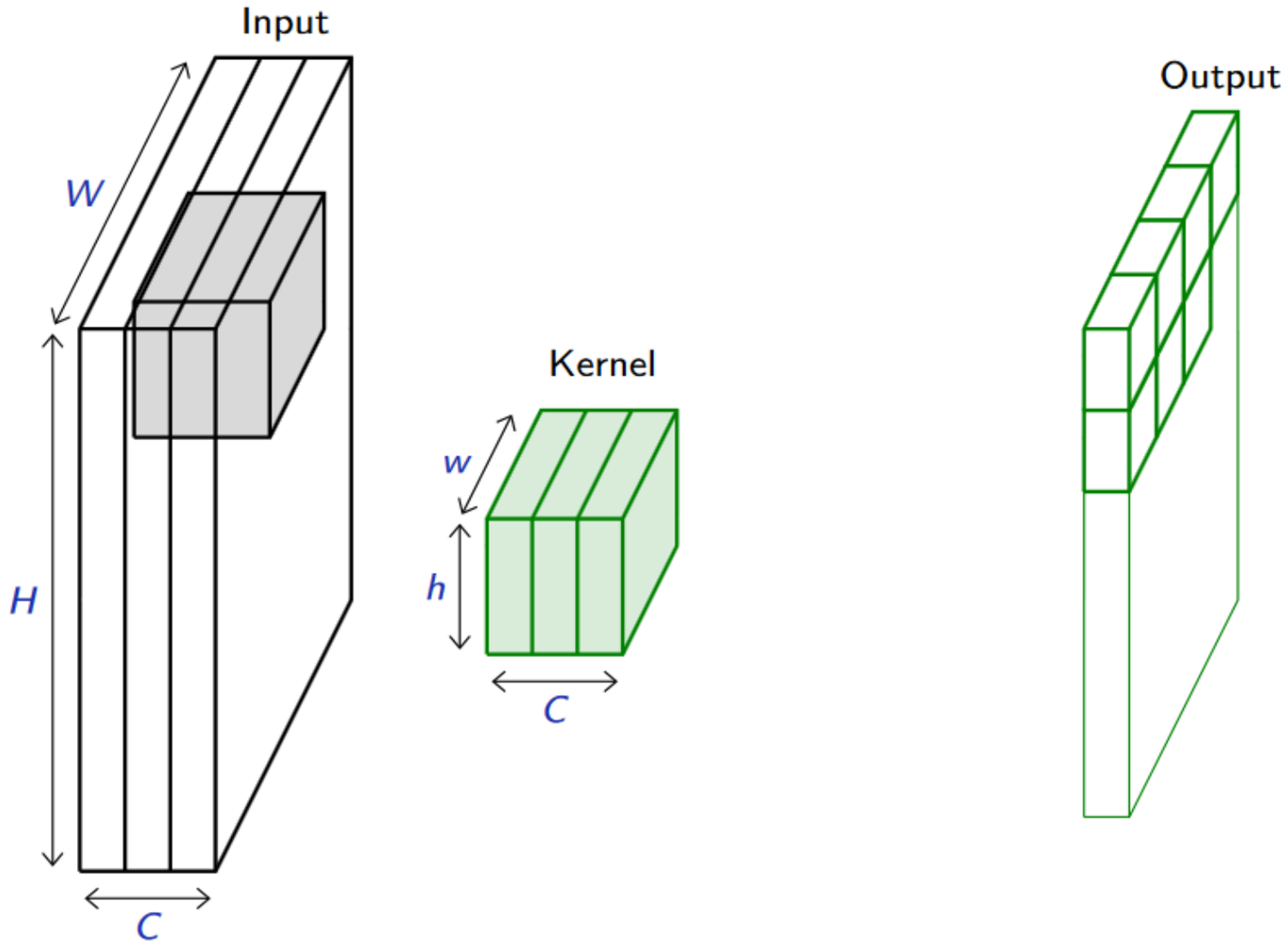
outline $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

bottom sobel $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$

right sobel $\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$

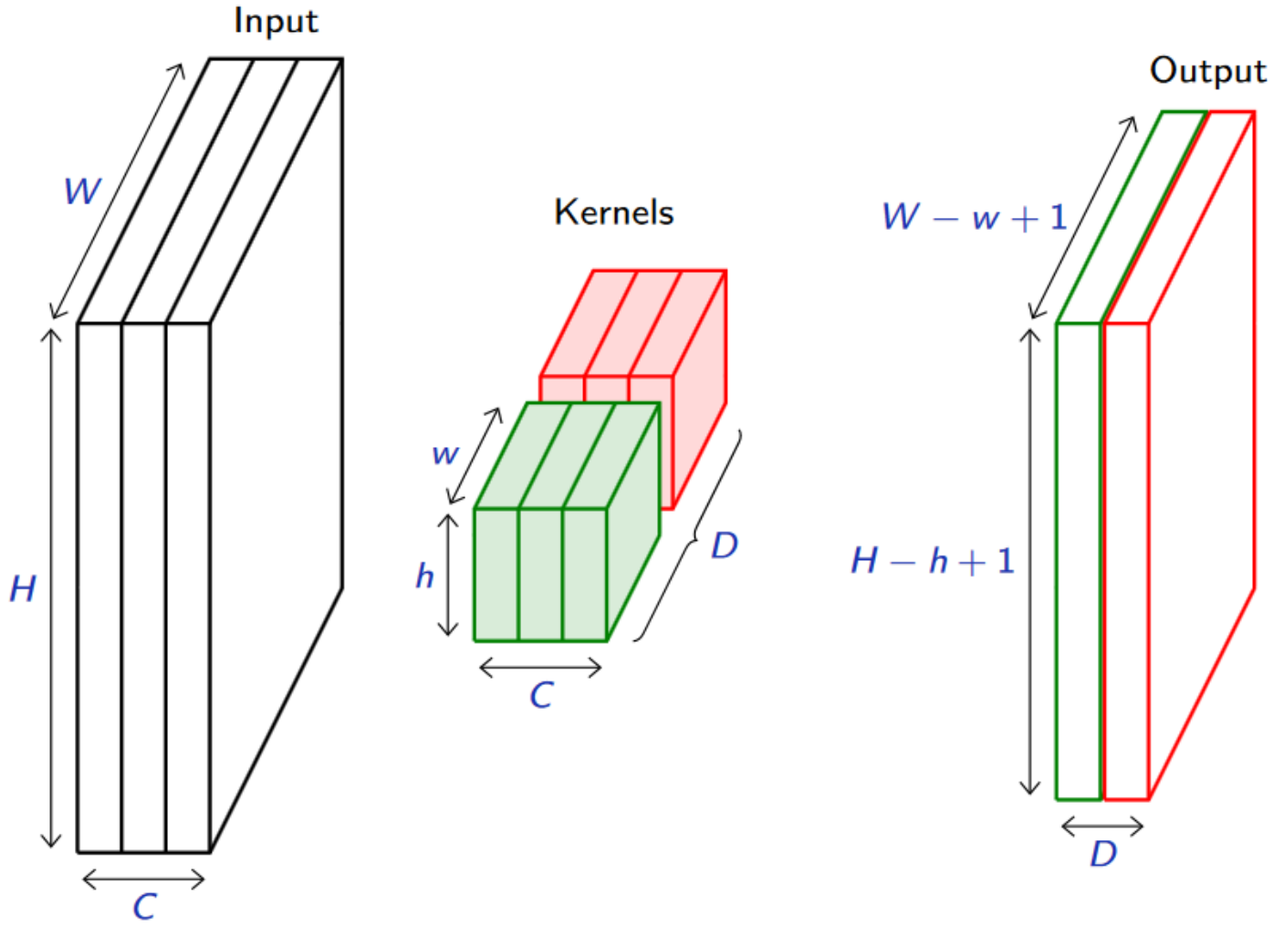
custom $\begin{bmatrix} +1 & -1 & +1 \\ -1 & 0 & -1 \\ +1 & -1 & +1 \end{bmatrix}$

Свёртка (Convolution): глубина



глубина тензора (число каналов) = глубина свёртки

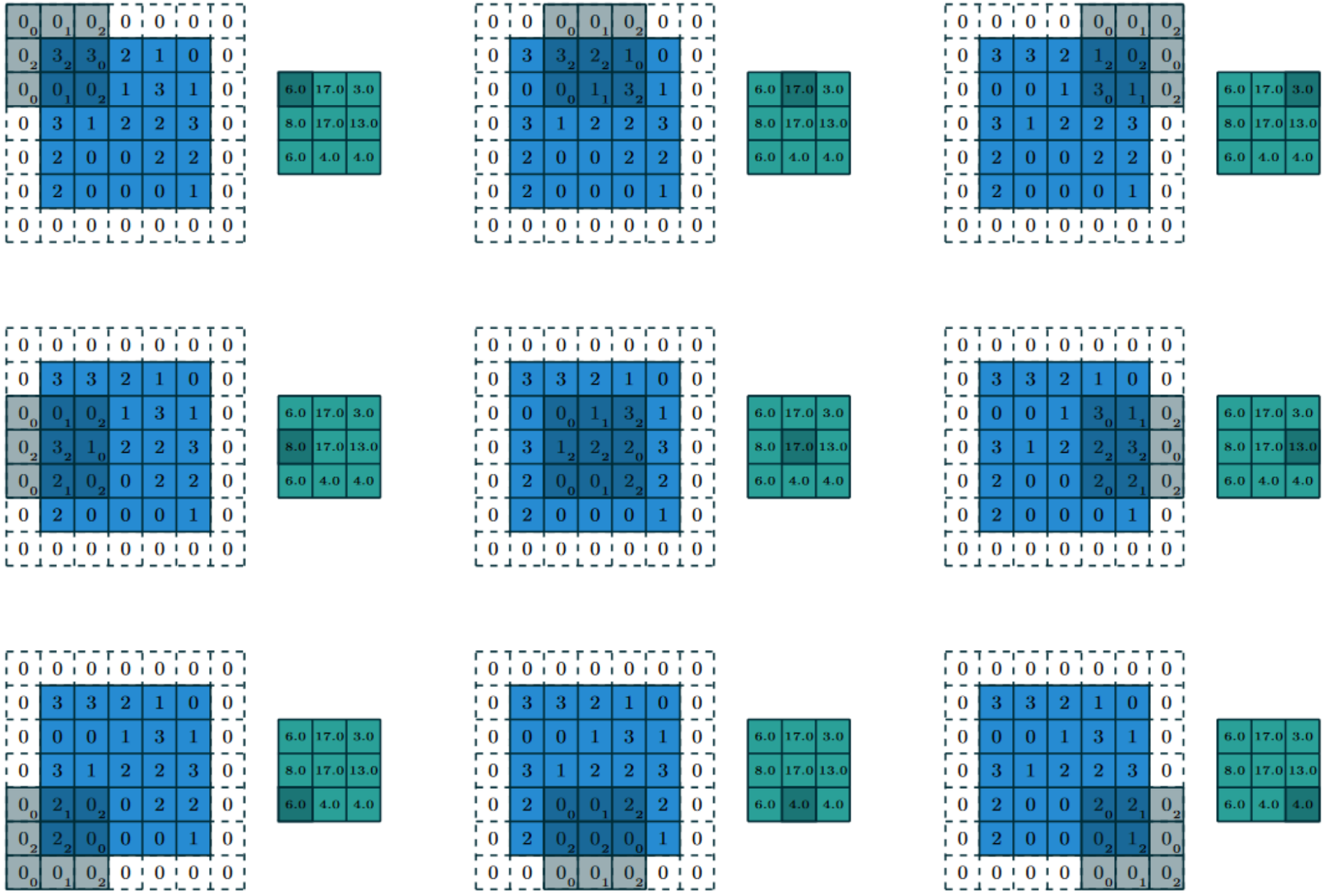
Свёртка (Convolution): применение нескольких свёрток



каждая свёртка – 1 «лист» на выходе, k свёрток – k-канальный выход

получаем на выходе тензор, глубина = число применяемых свёрток

свёрточный слой (для картинок) – 4D-массив $C_{out} \times C_{in} \times h \times w$


6.0 17.0 3.0
8.0 17.0 13.0
6.0 4.0 4.0

Свёртка (Convolution): минутка кода

```
torch.nn.Conv2d(in_channels: int,
                out_channels: int,
                kernel_size: Union[T, Tuple[T, T]],
                stride: Union[T, Tuple[T, T]] = 1,
                padding: Union[T, Tuple[T, T]] = 0,
                dilation: Union[T, Tuple[T, T]] = 1,
                groups: int = 1,
                bias: bool = True,
                padding_mode: str = 'zeros') # 'reflect','replicate','circular'
input = torch.randn(20, 16, 50, 100)
m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2),
              dilation=(3, 1))
output = m(input)
```

in_channels, out_channels – количество каналов на входе и выходе – должны делиться на groups!

kernel_size – размеры ядра

stride – смещение (можно понижать разрешение)

padding – отступы

dilation – расстояние между точками ядра (увеличивает область зависимости)

groups – опр. связи между входом и выходом

Минутка кода: свёртка (Convolution)

Shape:

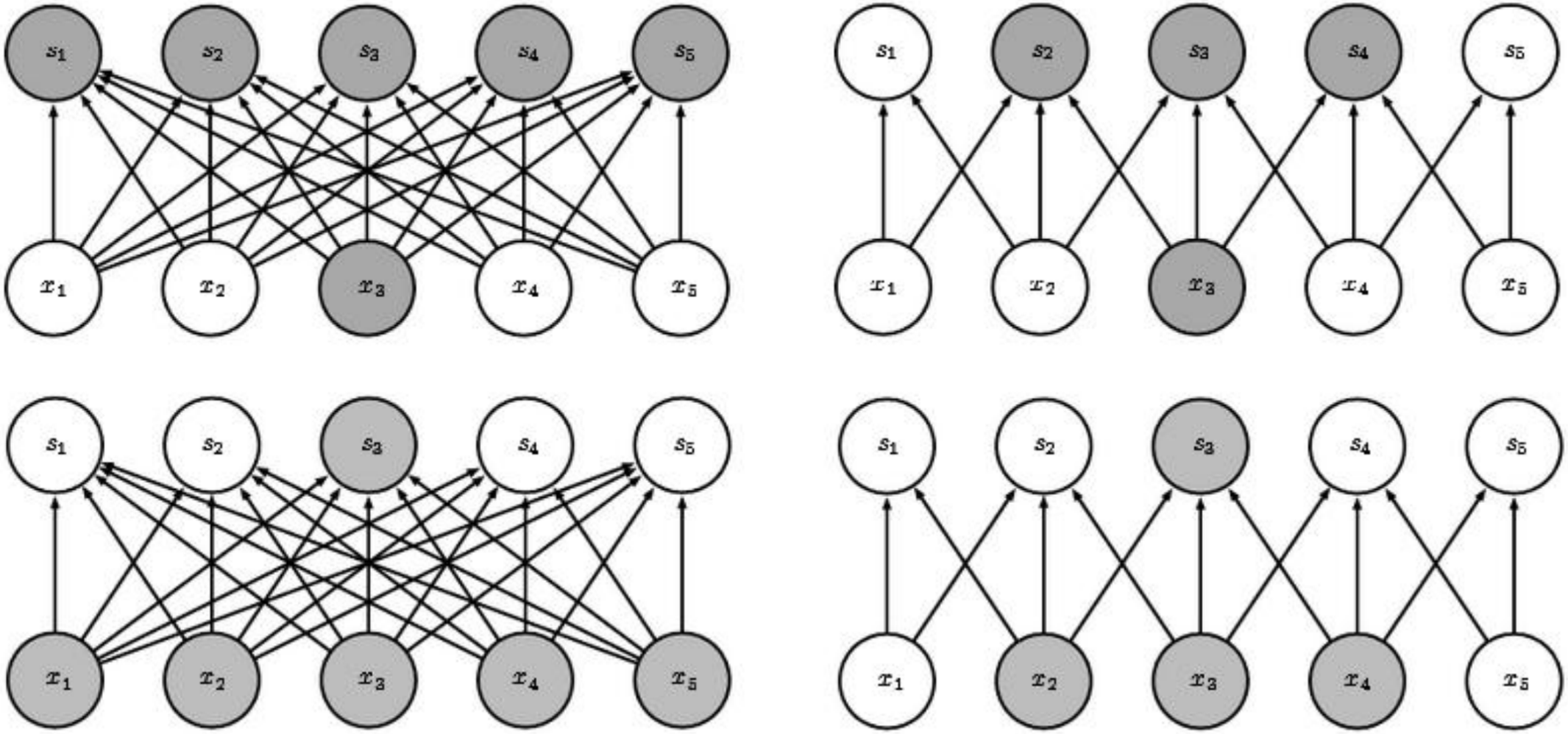
- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

размеры выхода (в разных реализациях – по-разному)

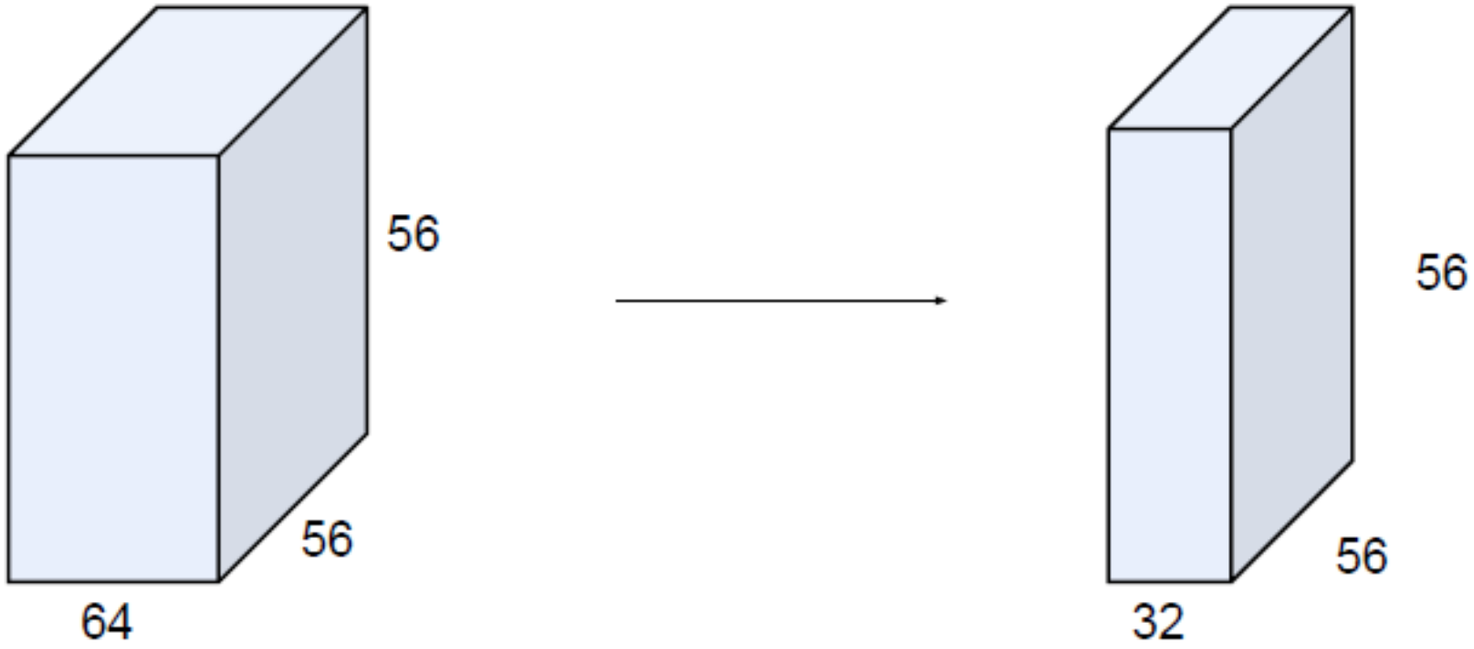
Разреженные взаимодействия (sparse interactions)



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

Смысл свёрток 1×1 (Pointwise Convolution)

применение 32 свёрток 64×1×1:



Преобразование признаков!

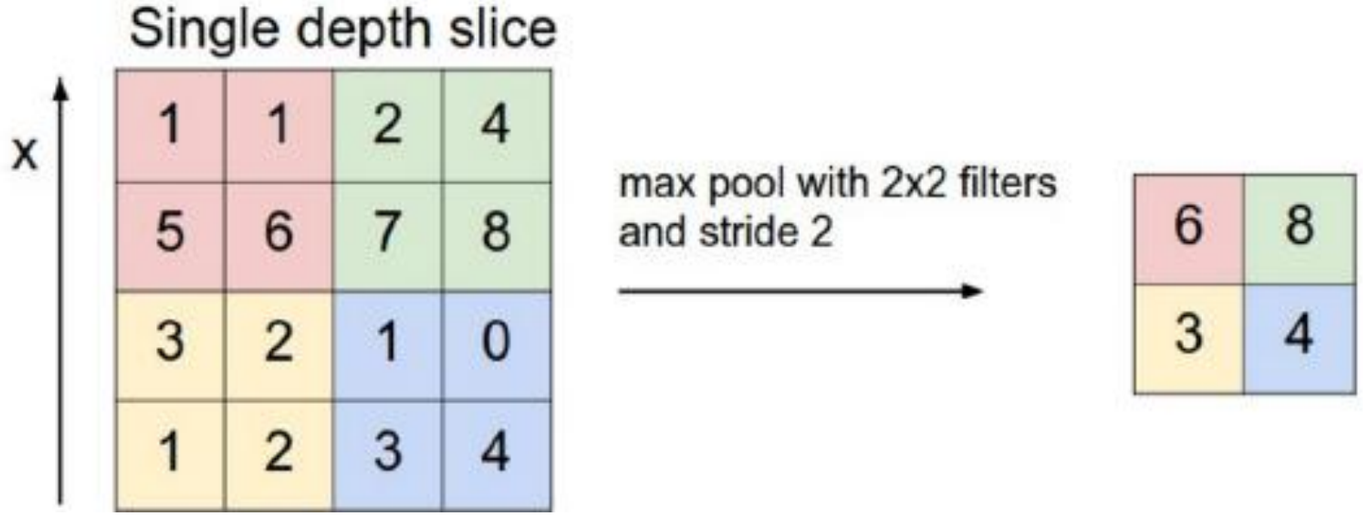
будет часто использоваться – это своеобразная мини-нейронка

+ изменение числа каналов

+ «узкое горло» в НС

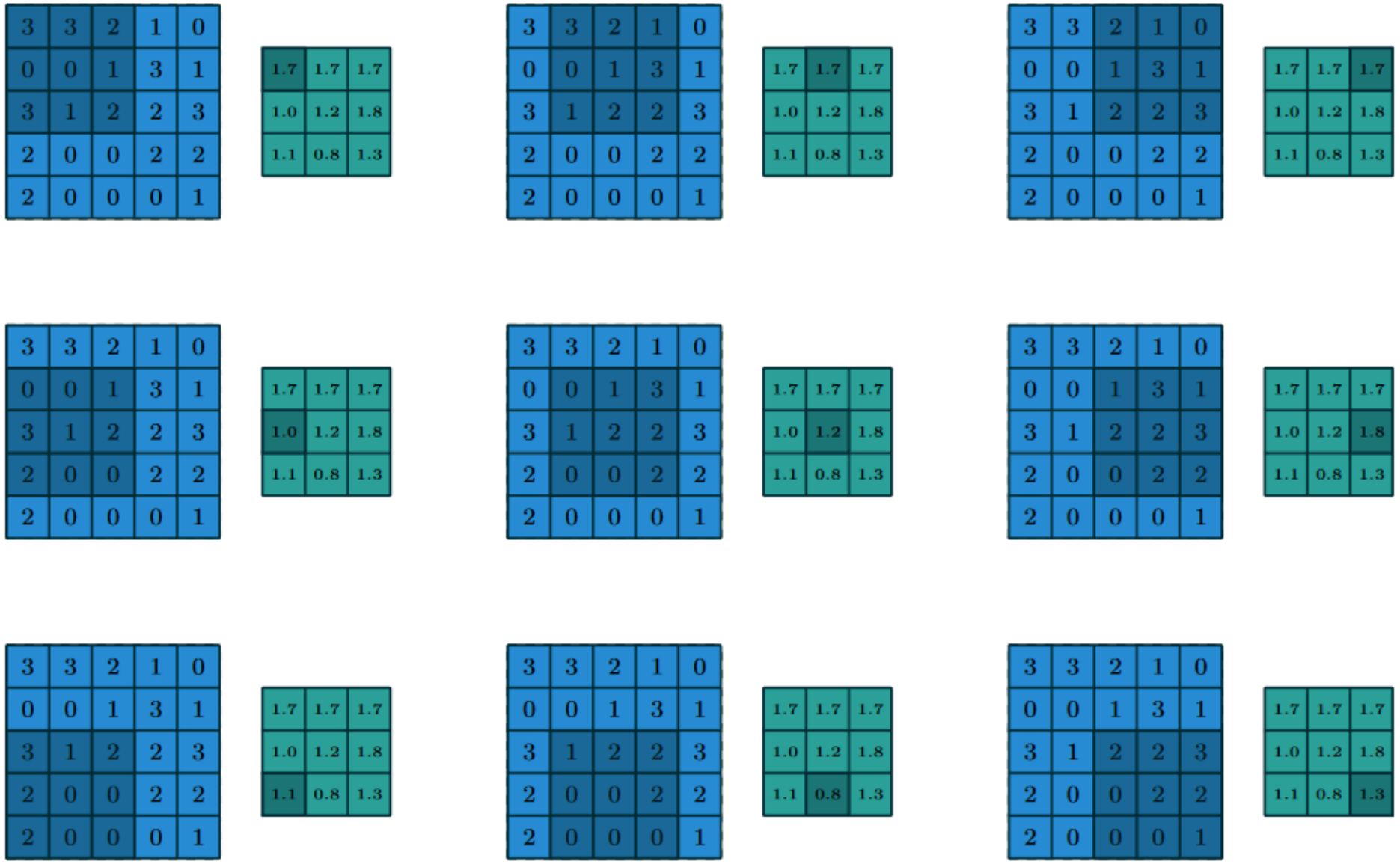
Pooling (агрегация, субдискретизация / subsampling)

для каждого признака канала надо определить,
нашли ли паттерн
используем функцию агрегации (mean, max, …)



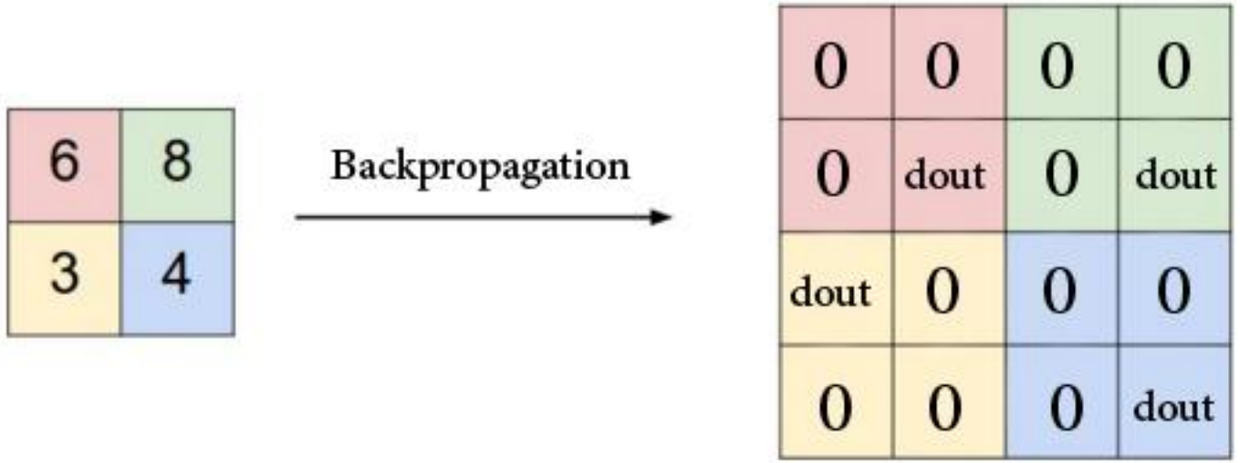
делается независимо по каналам ⇒ сохраняет число каналов (глубину тензора)

Агрегация (Pooling) усреднением



Агрегация (Pooling): дифференцирование

При дифференцировании возвращают градиент в позициях максимумов



Почему… $$\frac{\partial \max[f(x,w), g(x,w)]}{\partial w} = \begin{cases} \frac{\partial f(x,w)}{\partial w}, & f(x,w) \geq g(x,w), \\ \frac{\partial g(x,w)}{\partial w}, & f(x,w) < g(x,w). \end{cases}$$

https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/pooling_layer

Минутка кода: агрегация (Pooling)

```
torch.nn.MaxPool2d(kernel_size: Union[T, Tuple[T, ...]],
                   stride: Optional[Union[T, Tuple[T, ...]]] = None,
                   padding: Union[T, Tuple[T, ...]] = 0,
                   dilation: Union[T, Tuple[T, ...]] = 1,
                   return_indices: bool = False,
                   ceil_mode: bool = False) # как вычислять размеры результата

input = torch.randn(20, 16, 50, 32)
m = nn.MaxPool2d((3, 2), stride=(2, 1))
output = m(input)
```

Shape:

- Input: (N, C, H_{in}, W_{in})
- Output: (N, C, H_{out}, W_{out}), where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 * \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 * \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Устройство слоя свёрточной НС:

свёрточная часть: [свёртка $\rightarrow$ нелинейность $\rightarrow$ пулинг] $\times$ k

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»

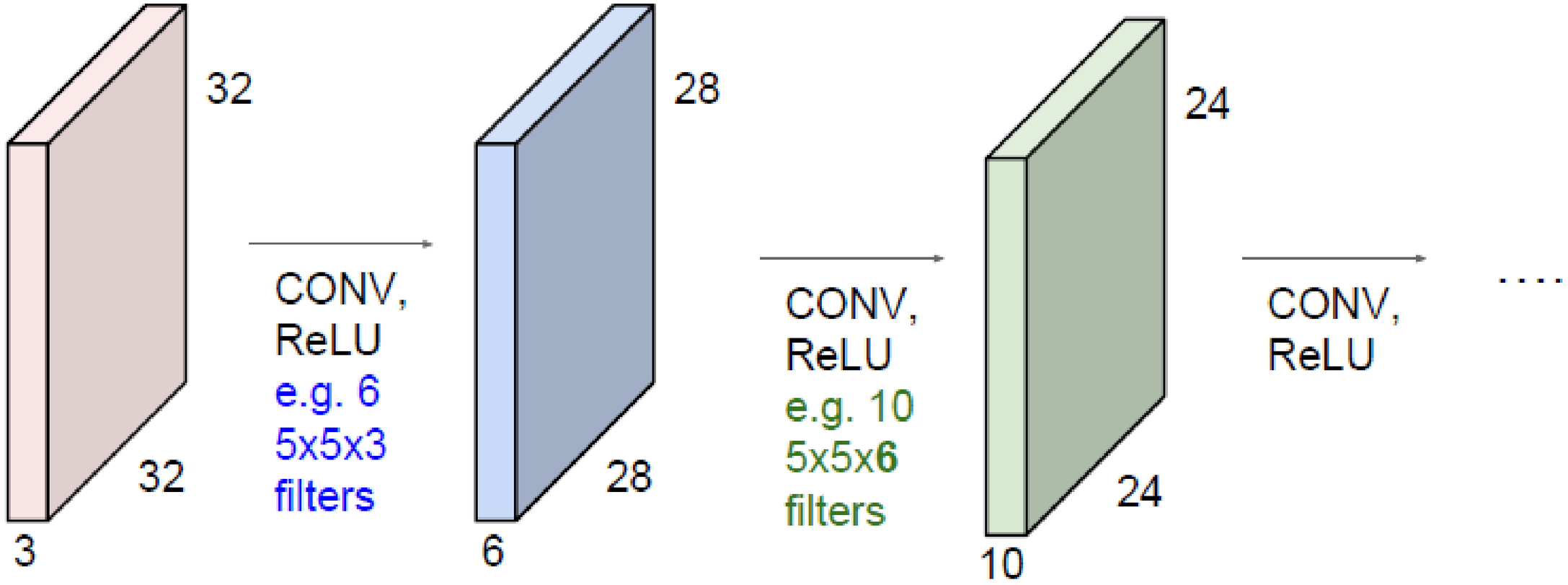
$\Rightarrow$ мало параметров

- **инвариантные преобразования (equivariant representations)**

инвариантность относительно сдвига

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

Свёрточная НС: тензор → тензор



Каждый тензор:

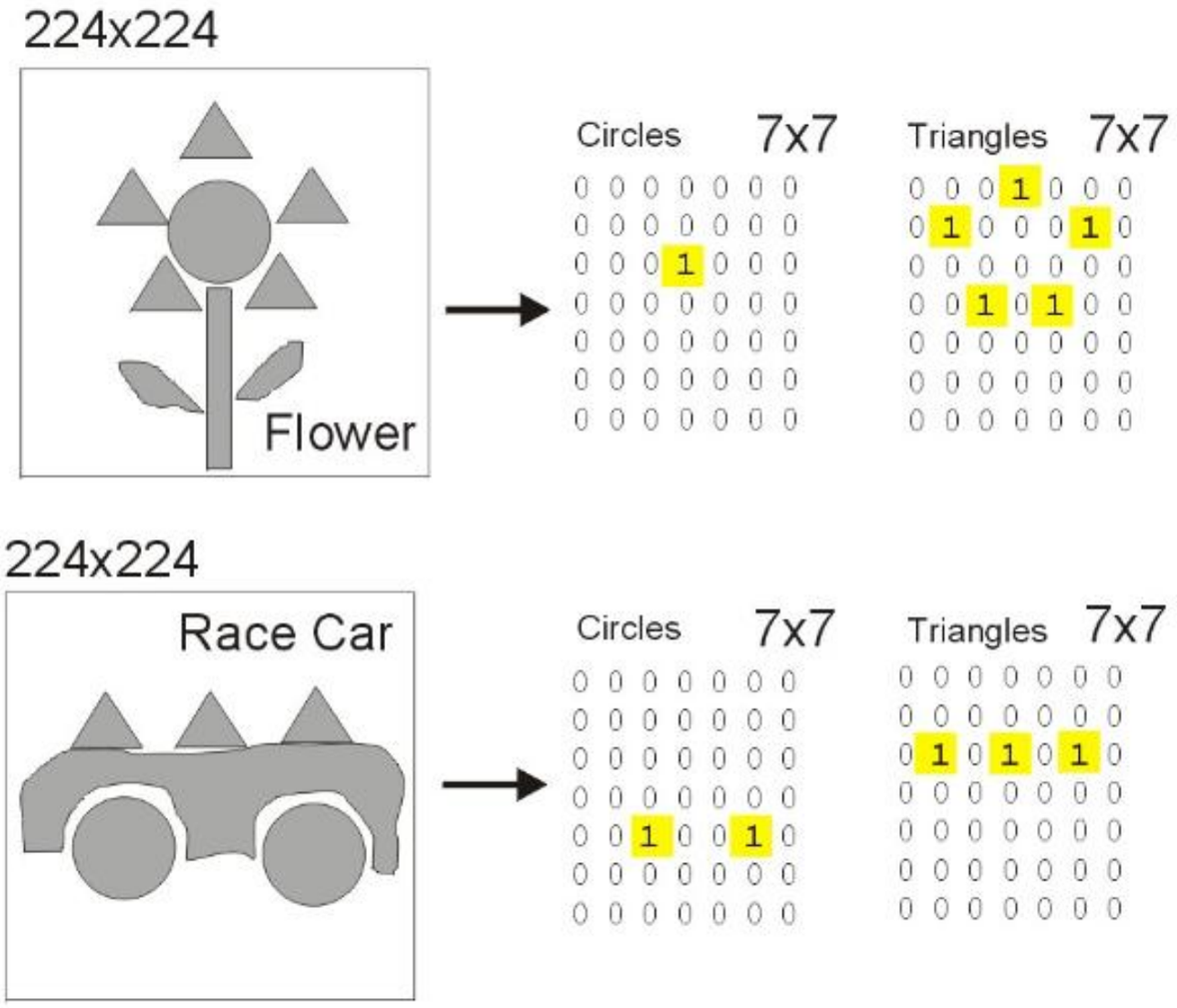
 # признаков / каналов (глубина) × ширина × высота

Свёрточная НС: тензор → тензор

```
f = nn.Conv2d(in_channels=1, out_channels=1, kernel_size=3)
x = torch.randn(1, 1, 28, 28)
print (x.shape)
x = f(x)
print (x.shape)
x = F.max_pool2d(x, kernel_size=2)
print (x.shape)
x = f(x)
print (x.shape)
x = F.max_pool2d(x, kernel_size=2)
print (x.shape)
```

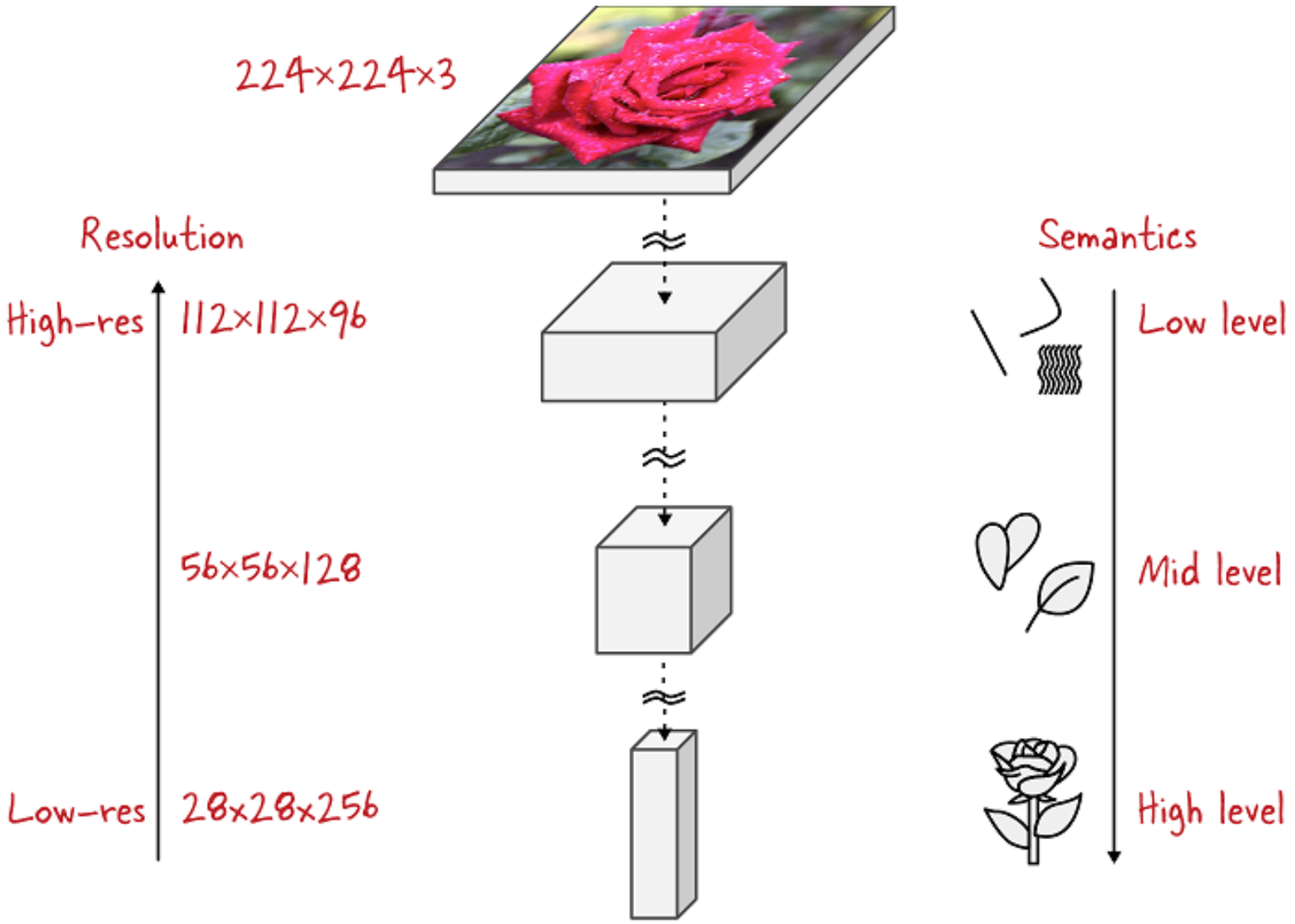
```
torch.Size([1, 1, 28, 28])
torch.Size([1, 1, 26, 26])
torch.Size([1, 1, 13, 13])
torch.Size([1, 1, 11, 11])
torch.Size([1, 1, 5, 5])
```

Визуализация признаков: за что могут отвечать свёртки последующих слоёв…



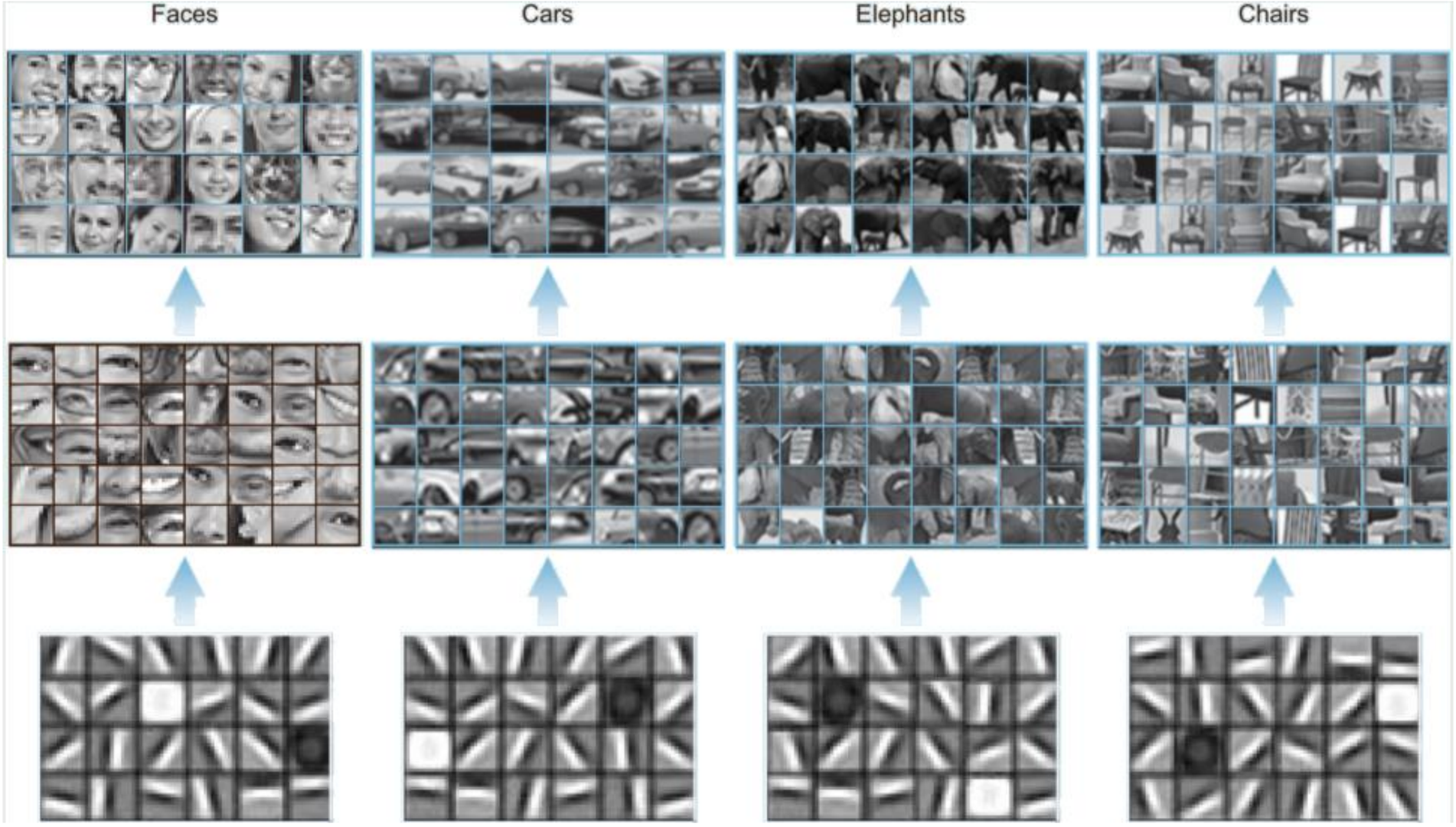
https://www.kaggle.com/c/siim-isic-melanoma-classification/discussion/160147

Визуализация признаков



[Practical Machine Learning for Computer Vision]

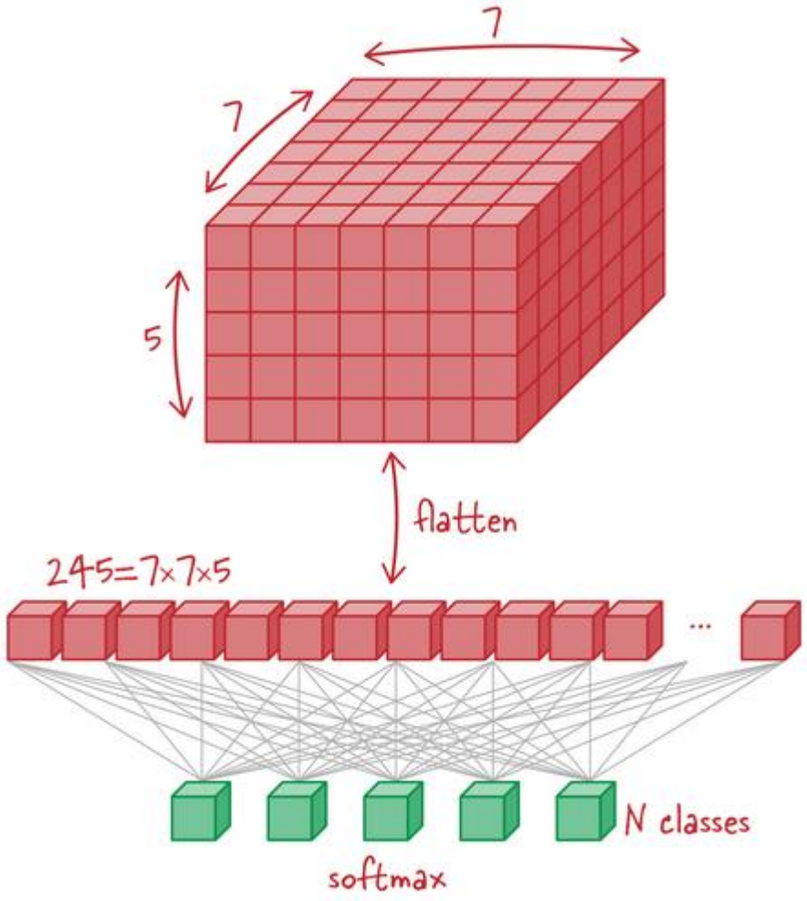
Визуализация признаков

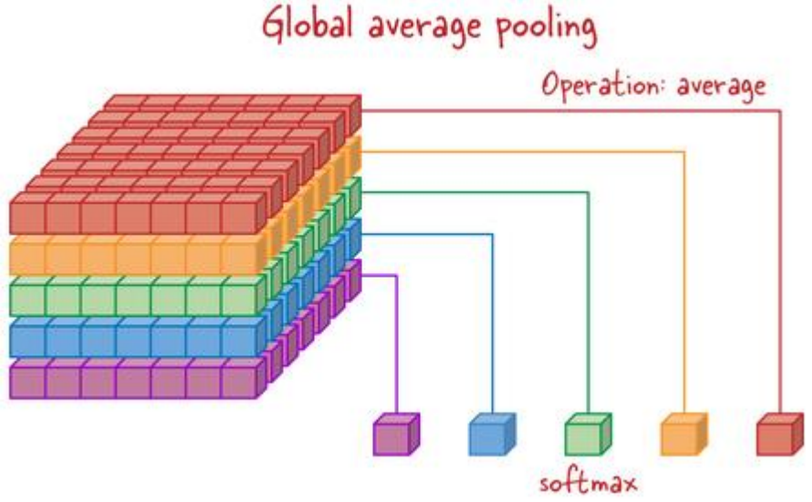


[Mohamed Elgendy]

Последние слои CNN – векторизация / глобальный пулинг

как перейти от H×W-пространства к пространству «однородных признаков»



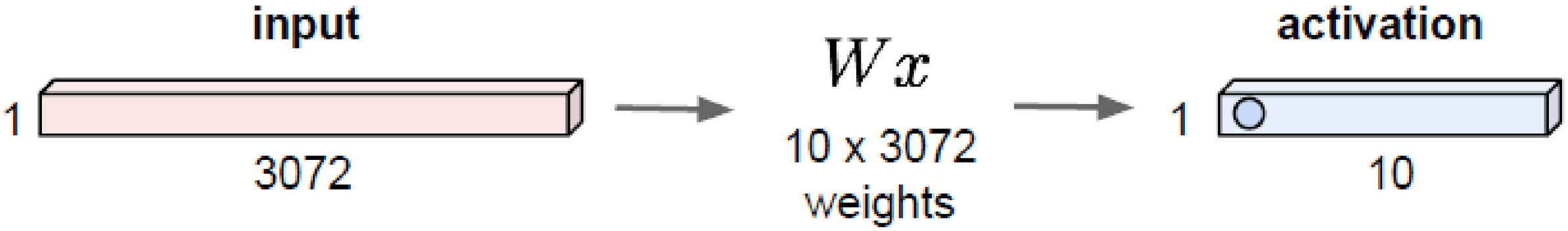


Совсем пропадает пространственная информация

[Practical Machine Learning for Computer Vision]

Последние слои CNN – «полносвязная часть»

тензор 3×32×32 → 3072D-вектор → линейный слой → активация:

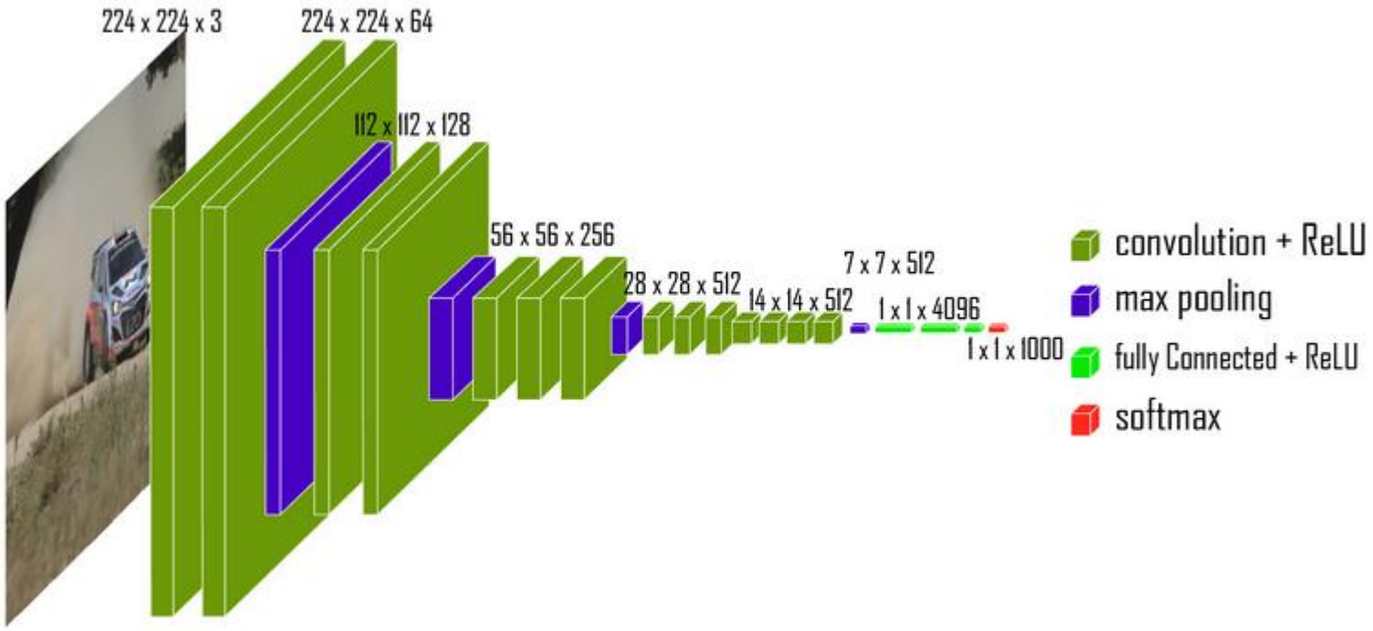


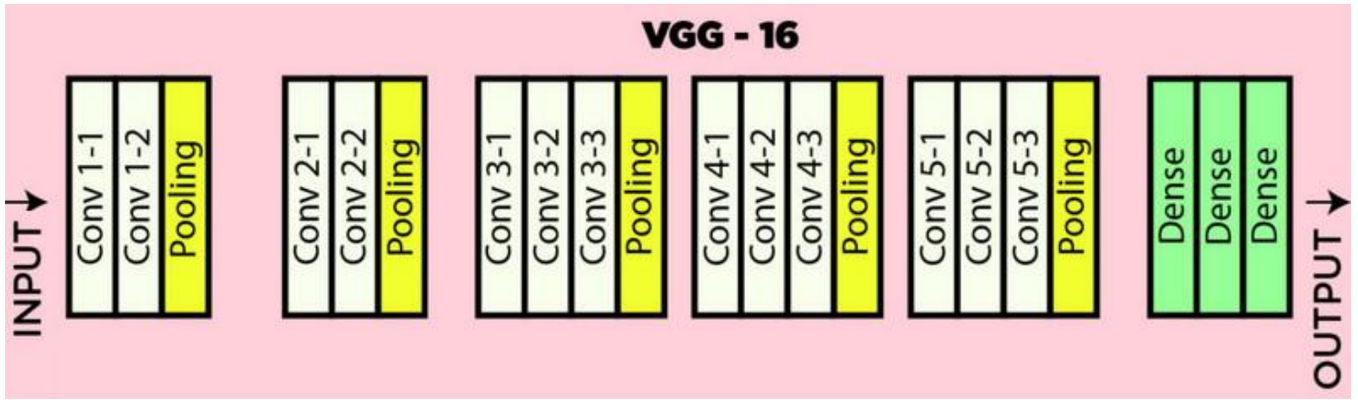
в конце свёрточной сети «обычные» полносвязные слои

для решения задачи классификации / регрессии

есть специальные сети без этих слоёв (дальше)

Архитектура CNN



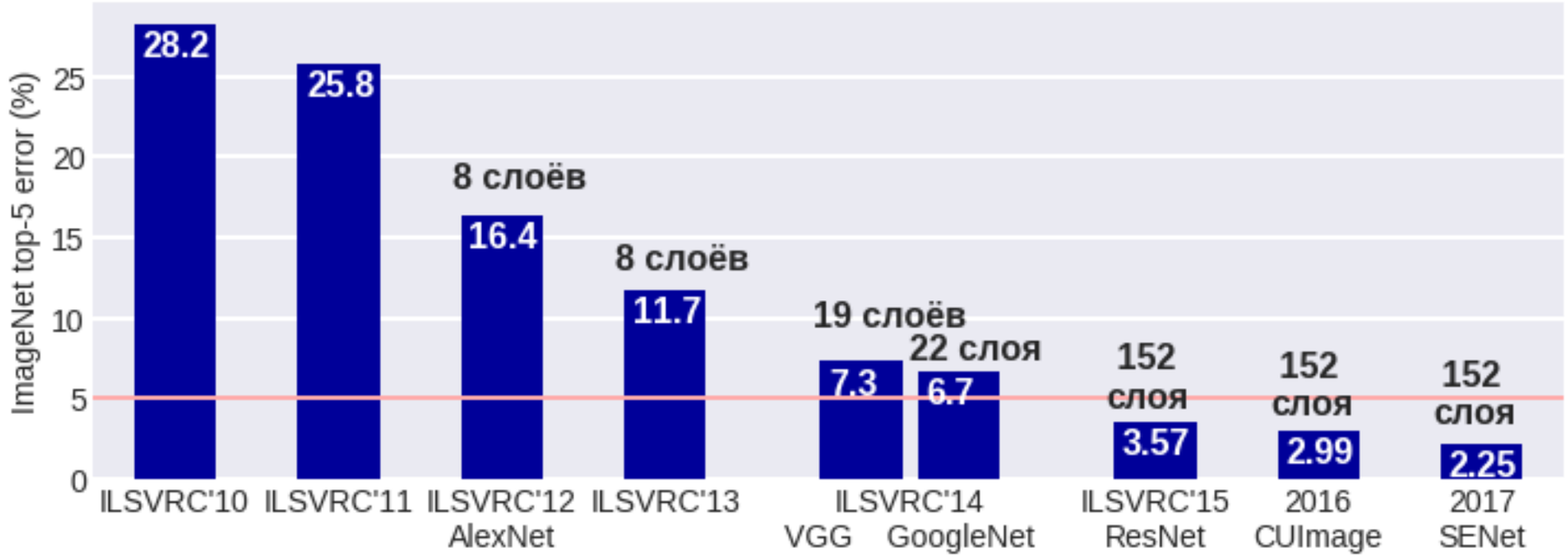


https://www.geeksforgeeks.org/vgg-16-cnn-model/

Минутка кода

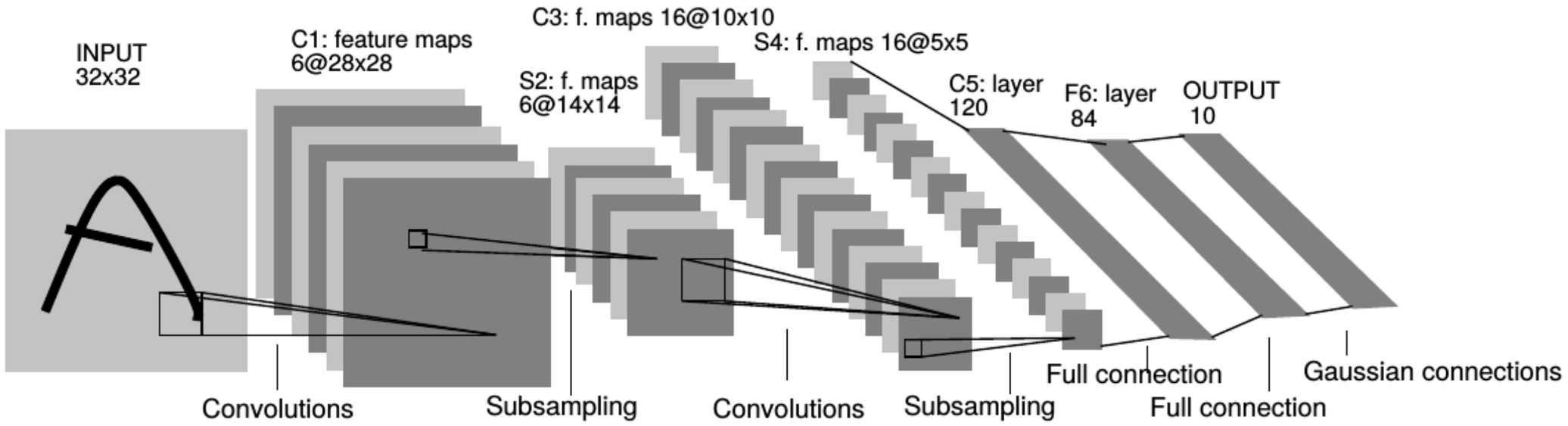
```
class CNN(nn.Module):
    def __init__(self, input_size, n_feature, output_size):
        super(CNN, self).__init__()
        self.n_feature = n_feature
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=n_feature, kernel_size=5)
        self.conv2 = nn.Conv2d(n_feature, n_feature, kernel_size=5)
        self.fc1 = nn.Linear(n_feature*4*4, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x, verbose=False):
        x = self.conv1(x)
        x = F.relu(x)
        x = F.max_pool2d(x, kernel_size=2)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, kernel_size=2)
        x = x.view(-1, self.n_feature*4*4)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.log_softmax(x, dim=1)
        return x
```

Революция в машинном обучении



ошибка человека – 5.1

LeNet-5 (1998)



5 = 2 свёрточных слоя + 3 полносвязных

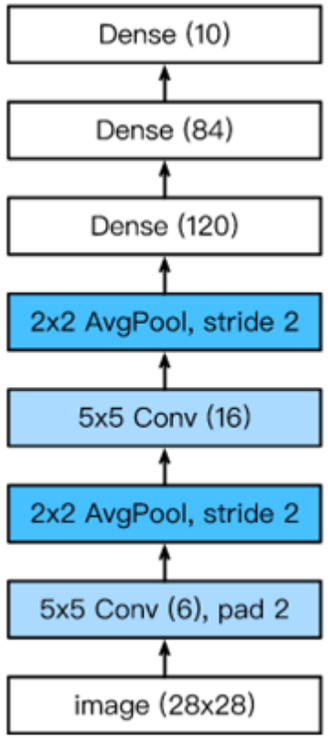
свёртки 5×5

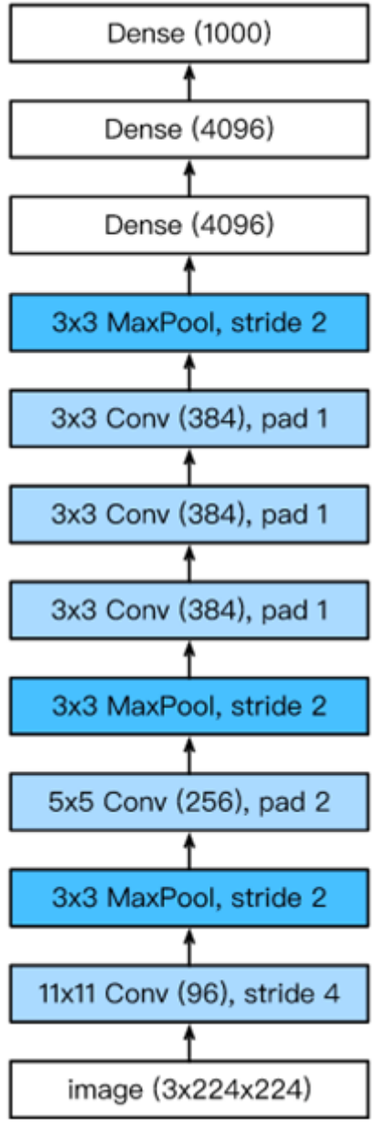
C@H×W

третий слой можно считать свёрточным, а можно полносвязным

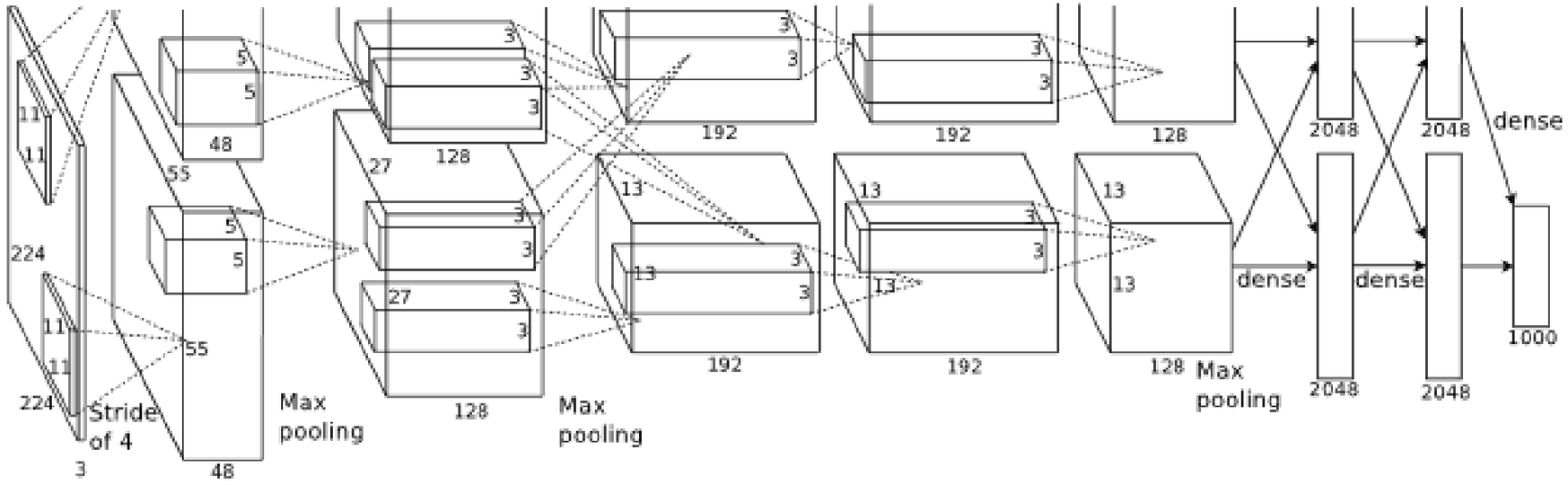
(там 5×5-свёртка)

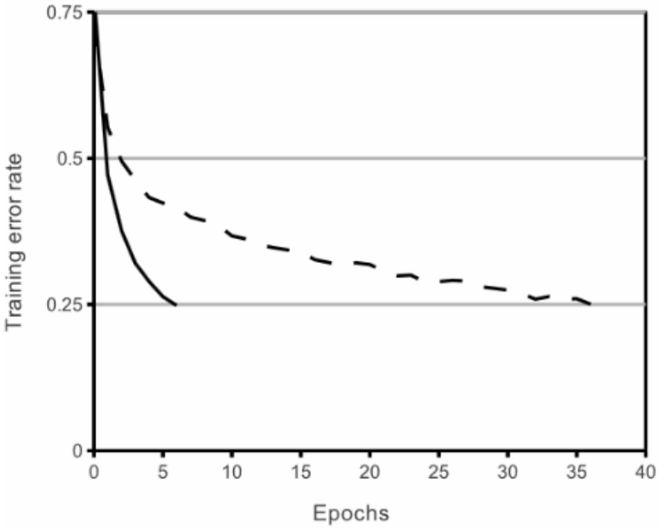
LeNet-5 → AlexNet (2012)





AlexNet (2012)



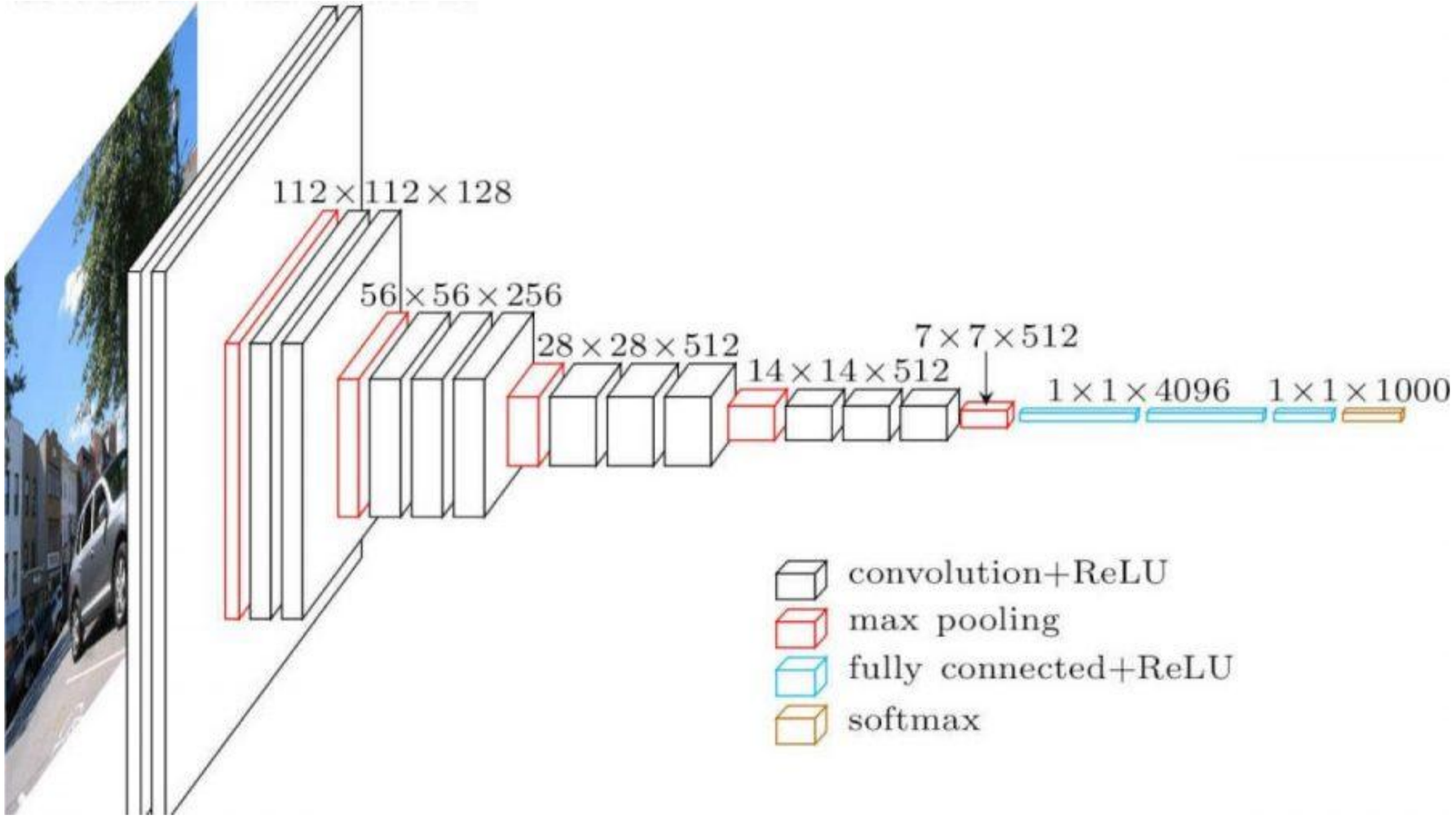


- **ReLU – после ∀ conv и dense слоя (см. рис, скорость 6×)**
  - **MaxPool (вместо AvgPool), полно-связные слои**
    - **Data augmentation**
  - **Dropout 0.5 (но и время обучения 2×)**
  - **Batch size = 128, SGD Momentum = 0.9**
    - **60M параметров / 650K нейронов**
    - **1 неделя на 2 GPU (50x над CPU)**
      - **7 скрытых слоёв**

Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

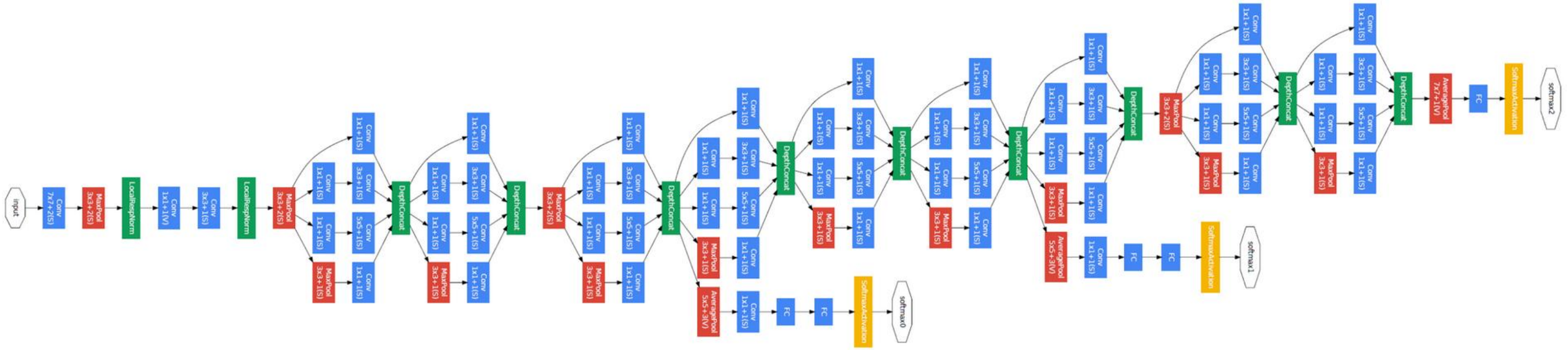
Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton “ImageNet Classification with Deep Convolutional Neural Networks” // https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

VGG (2014, VGG group, Oxford)



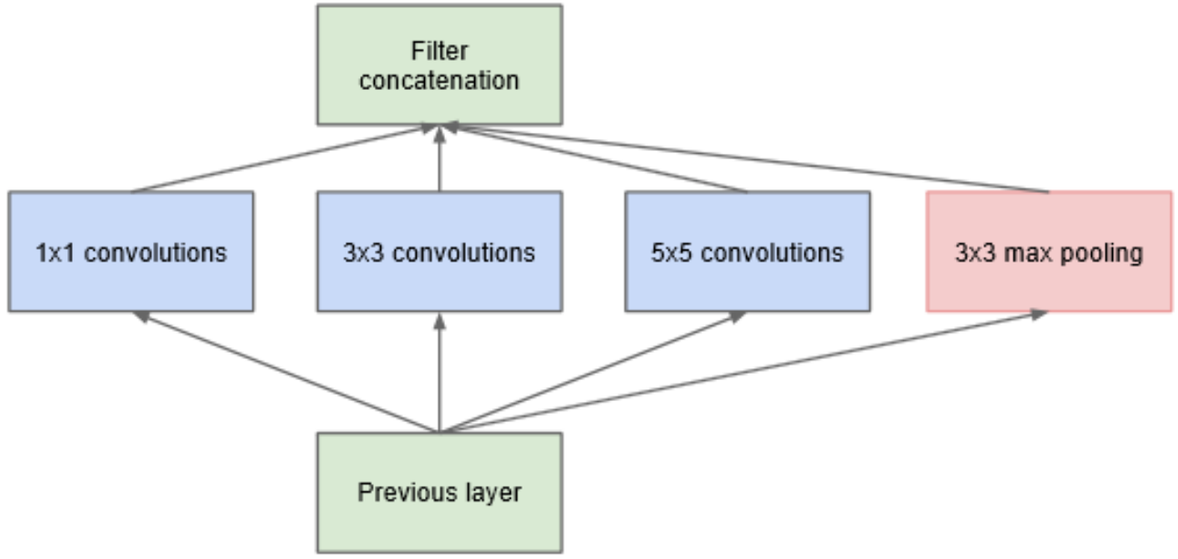
K. Simonyan, A. Zisserman «Very Deep Convolutional Networks for Large-Scale Image Recognition» https://arxiv.org/pdf/1409.1556.pdf

GoogLeNet / Inception (2014)

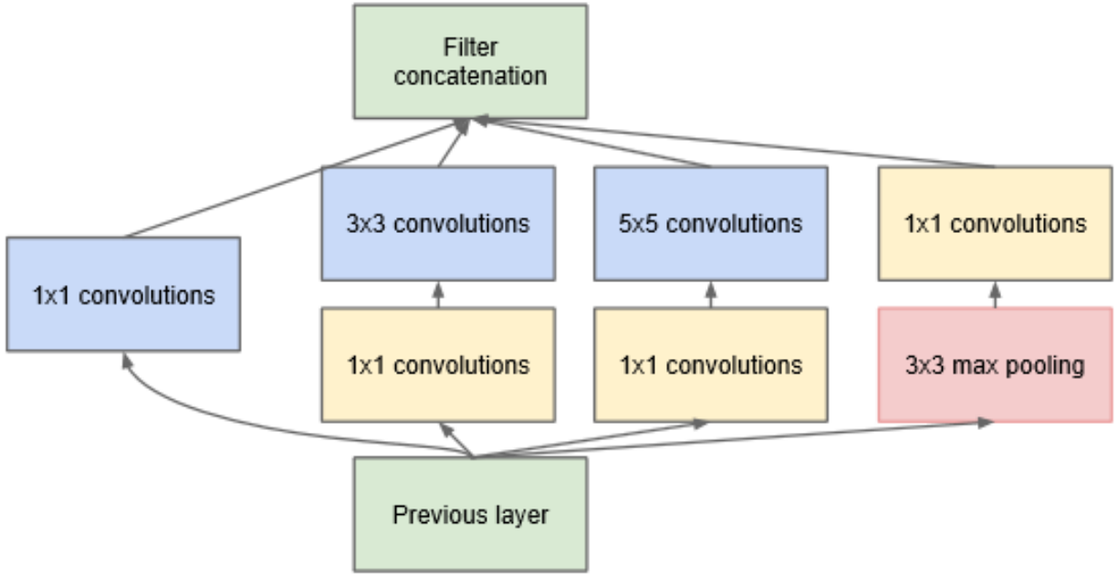


- **«конструктор» НС – «Modular Architecture»**
- **22 слоя – нет полносвязных**
- **Модуль «Inception», 1×1-свёртки,**
- **5M параметров (меньше!)**
- **дополнительные выходы классификации** (с весом 0.3 к общей ошибке)
- **тоже ансамбль (из 7)**
- **Global Average Pooling** (немного улучшает качество)

Модуль «Inception»



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Изначальная идея – разные свёртки + пулинг

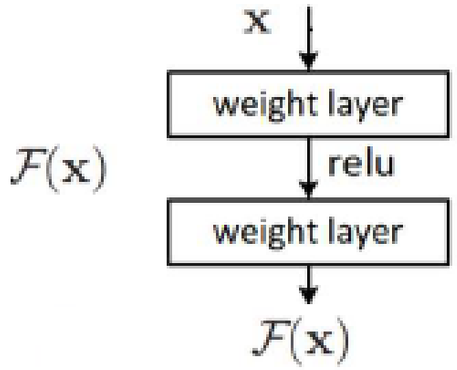
1×1-свёртки существенно уменьшают число параметров!

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich «Going Deeper with Convolutions» // https://arxiv.org/abs/1409.4842

ResNet = Residual Network (2015)

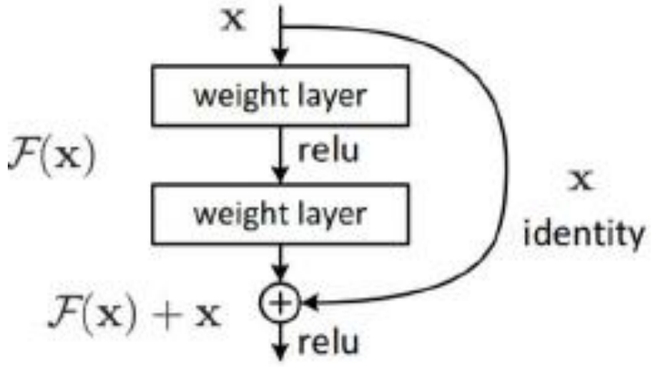
$$y = f(x)$$

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



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

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



skip (shortcut) connections

упрощение реализации тождественной функции,
по крайней мере, через два слоя

Есть ещё «highway networks» – прокидывание связей с настраиваемым «гейтом»

Просто добавление слоёв не помогает!

Добавлять надо по-умному…

He et al. «Deep Residual Learning for Image Recognition» https://arxiv.org/pdf/1512.03385.pdf

ResNet = Residual Network (2015)

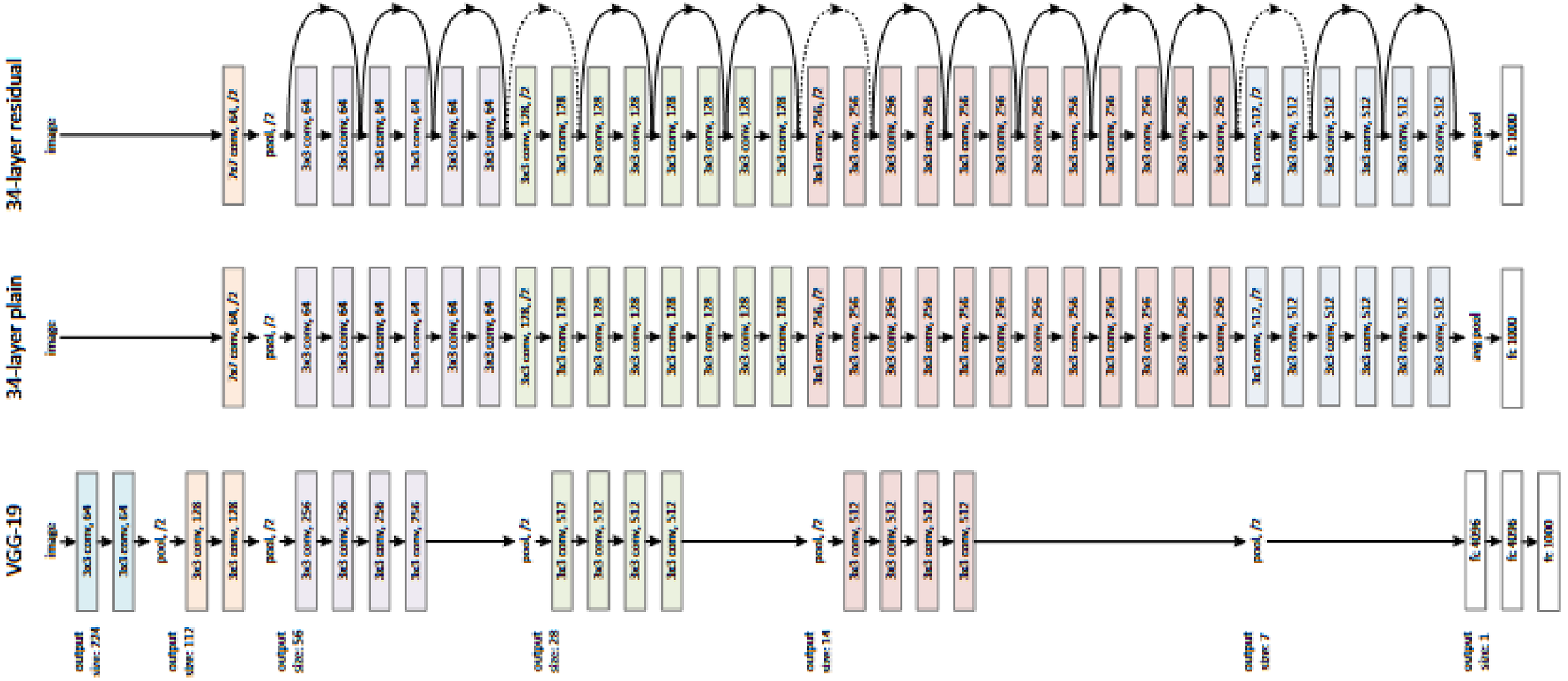
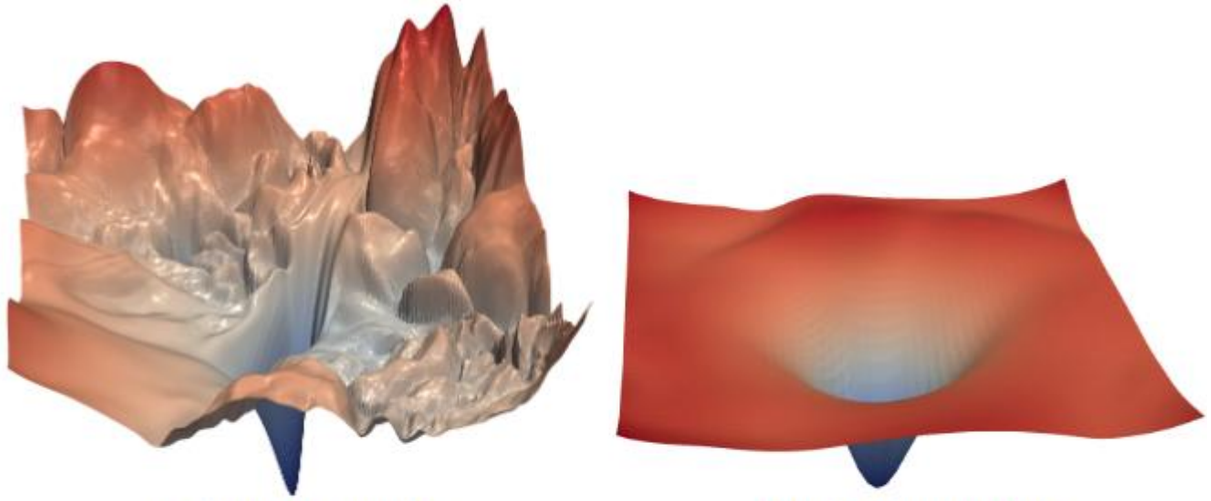


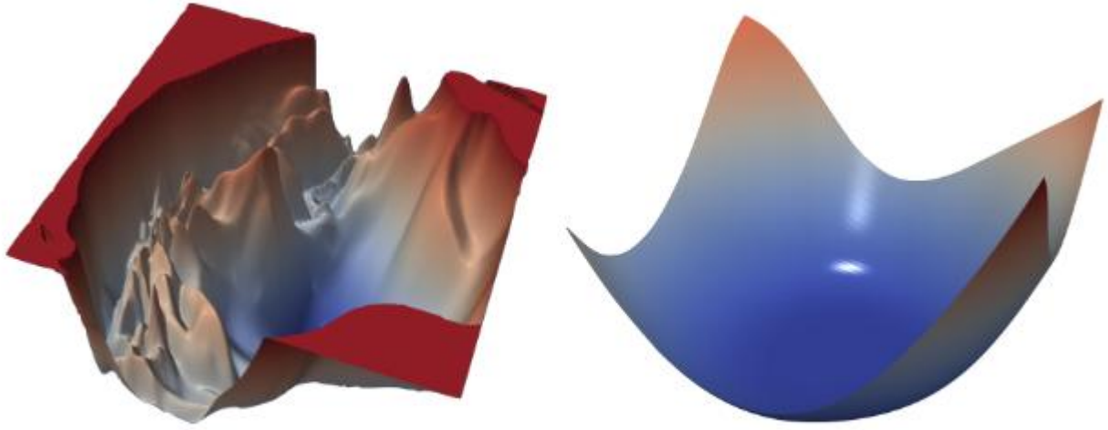
Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Эффект прокидывания связей



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.



(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

Figure 4: The loss surfaces of ResNet-110-noshort and DenseNet for CIFAR-10.

глубина и ширина «улучшают» поверхность функции ошибки

правильная оптимизация позволяет «правильно» идти по поверхности

«Visualizing the Loss Landscape of Neural Nets» https://arxiv.org/abs/1712.09913

Семантическая сегментация

В итоге что-то такое… «Deconvolution Network»

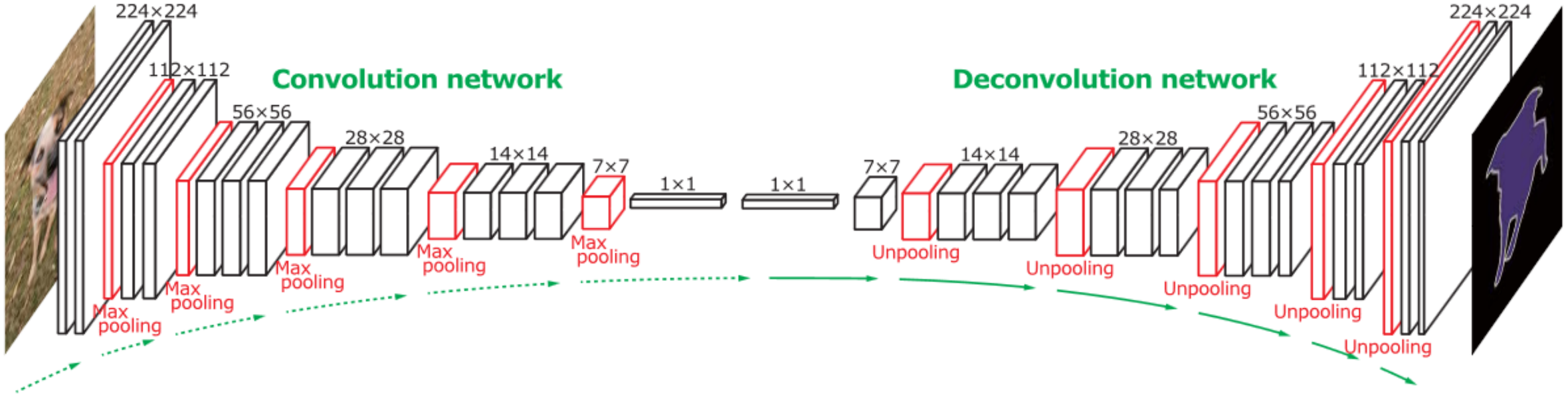


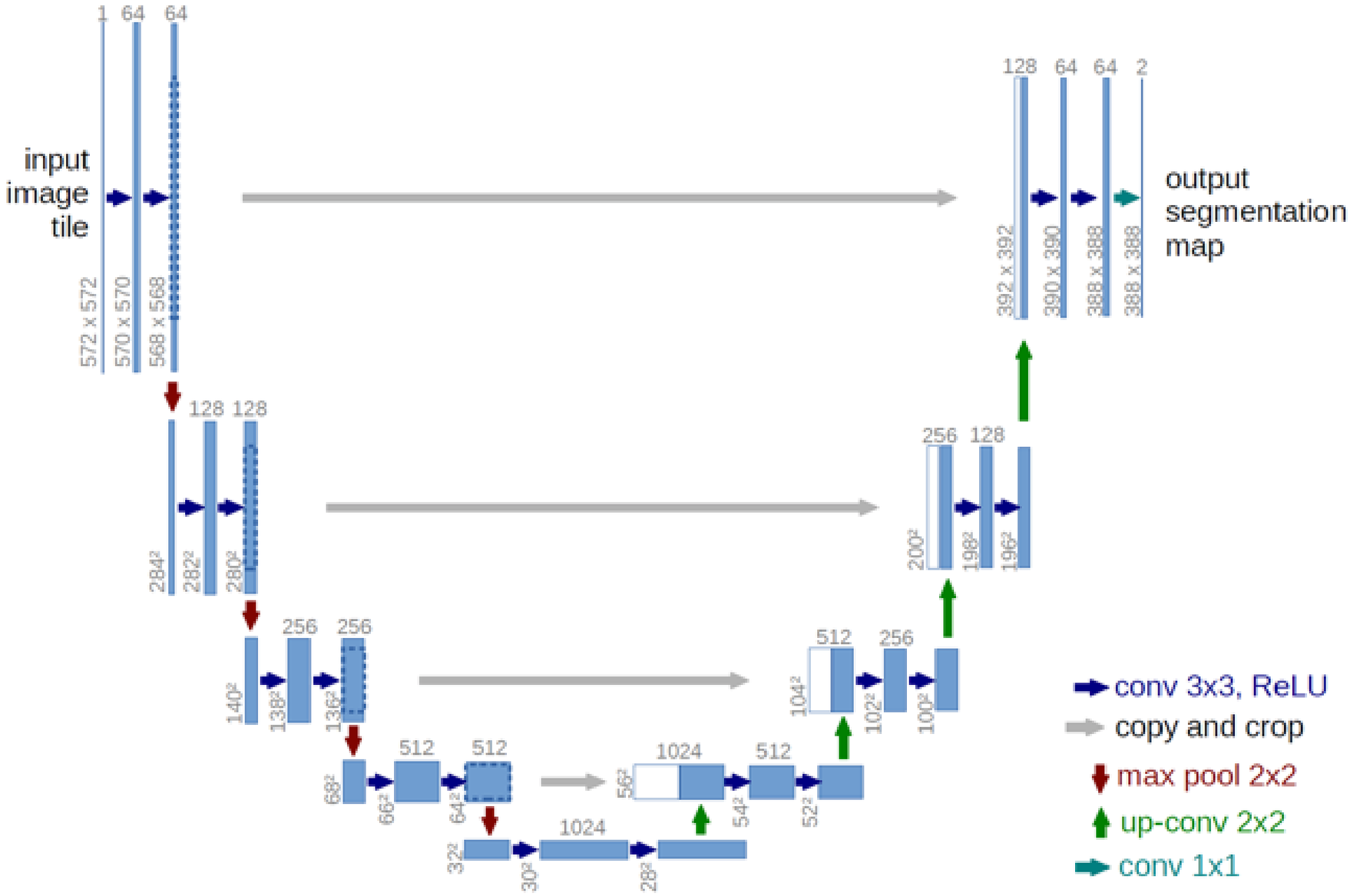
Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

кодировщик из VGG16

уменьшаются размеры, но увеличивается число каналов!

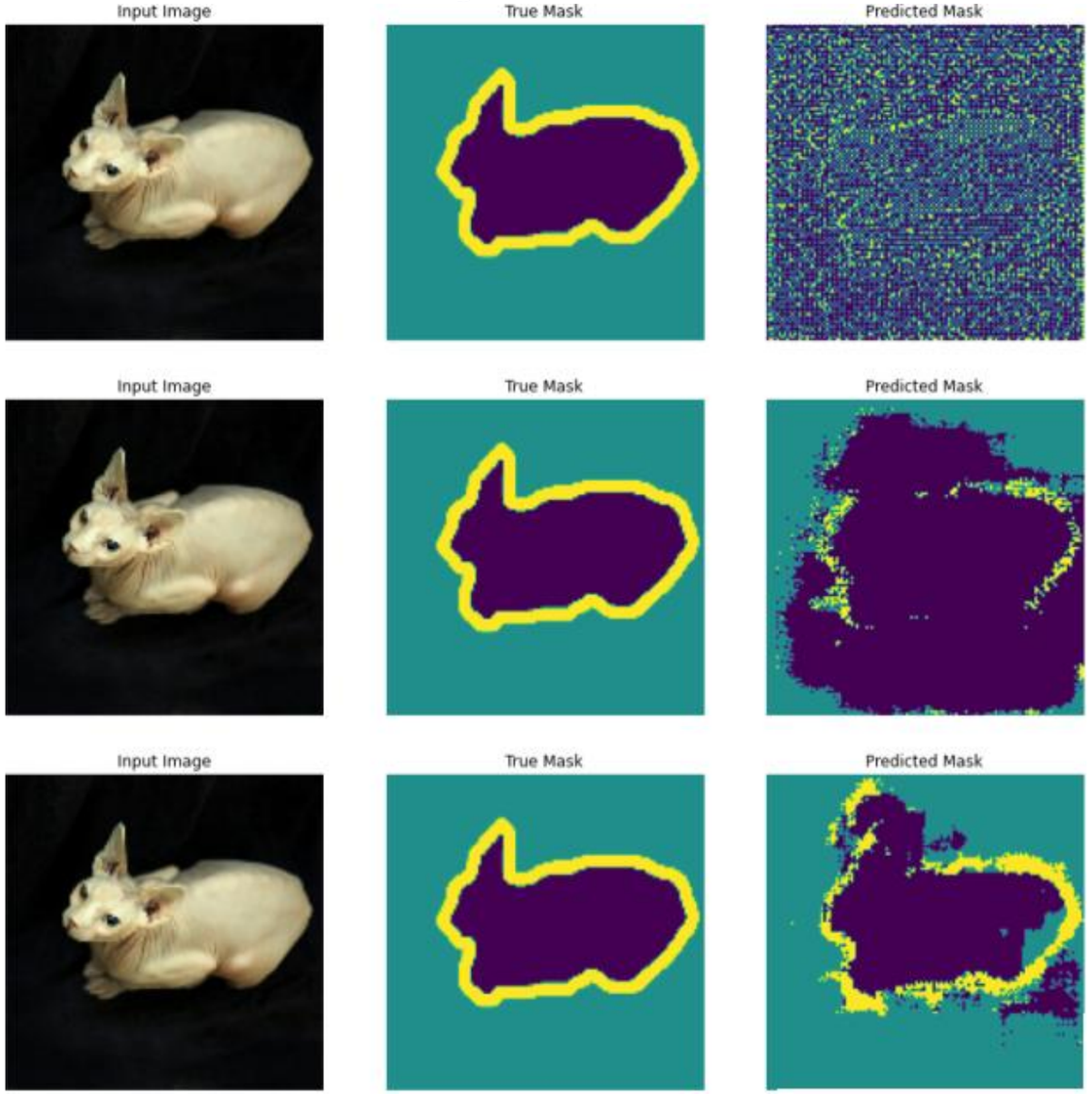
https://arxiv.org/pdf/1505.04366.pdf

U-Net (самая популярная)



«U-Net: Convolutional Networks for Biomedical Image Segmentation» [Ronneberger O. и др., 2015 https://arxiv.org/abs/1505.04597»

Как обучается U-net



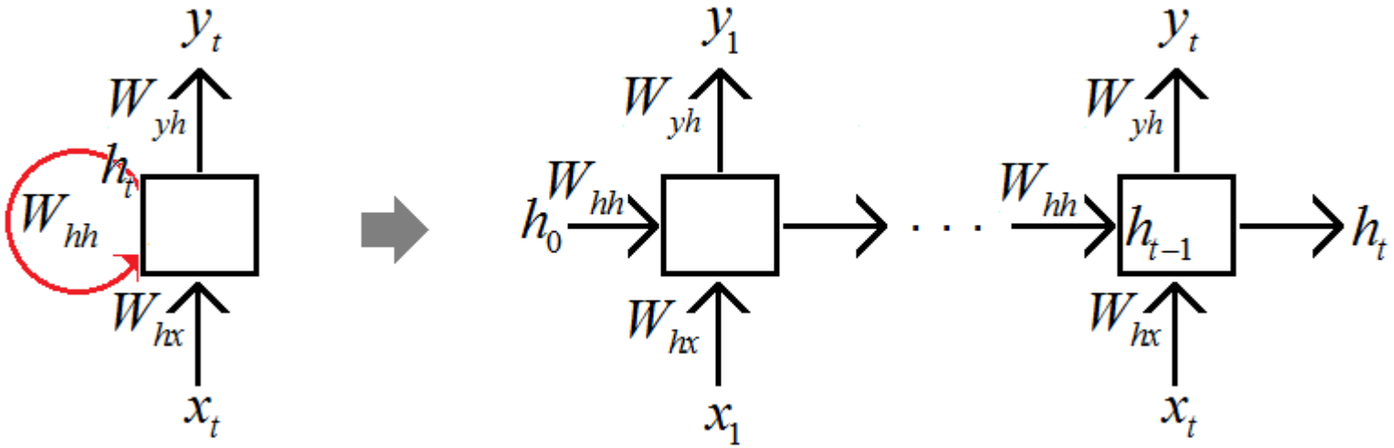
[Practical Machine Learning for Computer Vision]

Рекуррентная нейросеть (RNN = Recurrent Neural Network)

– для обработки / генерации последовательностей

использование выхода (output) / скрытого состояния (hidden state)

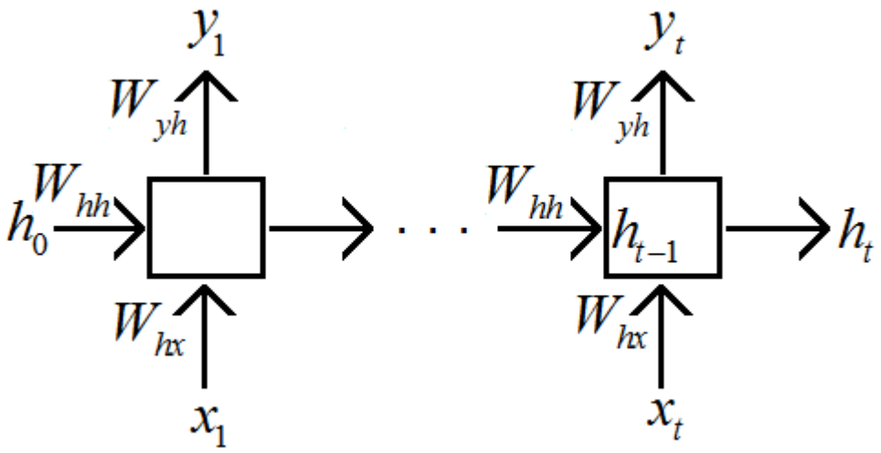
легко масштабируется при увеличении длины последовательностей



$$p(x_1,\ldots,x_T)=\prod_{t=1}^{T} p(x_t \mid x_{t-1},\ldots,x_1)$$

http://www.jefkine.com/general/2018/05/21/2018-05-21-vanishing-and-exploding-gradient-problems/

RNN: базовый блок (Vanilla RNN)



$$h_0 = \text{init}()$$

$$h_1 = \sigma(W_{hh}h_0 + W_{hx}x_1 + b_h)$$

$$\dots$$

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$

$$y_t = g(W_{yh}h_t + b_y)$$

пока рассматриваем однослойную сеть

линейный слой + нелинейность

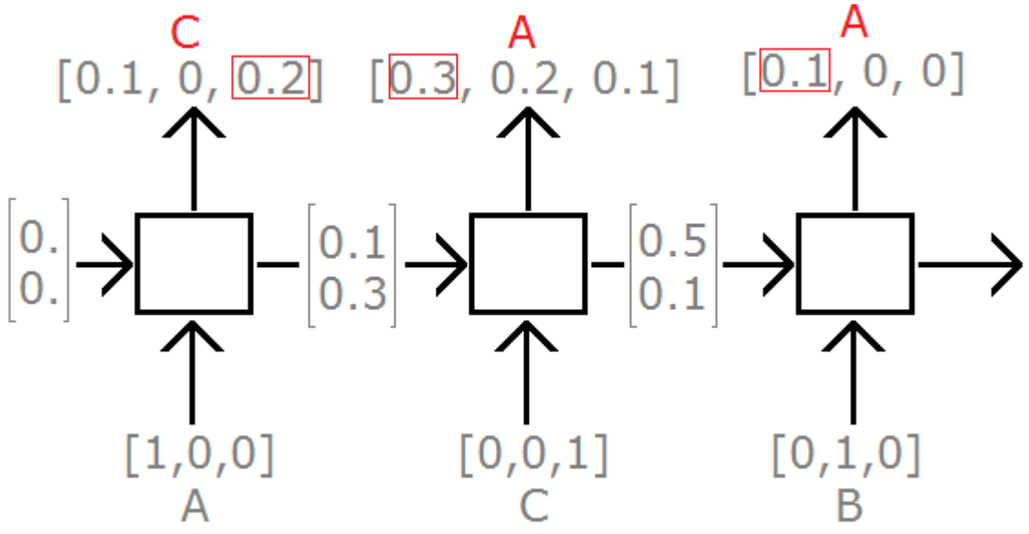
свободный член для простоты можно убрать

индексы могут быть другие

RNN: форма записи

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \quad \sim \quad h_t = \sigma\left([W_{hh}, W_{hx}]\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_h\right) \equiv \sigma\left(W[h_{t-1}; x_t] + b_h\right)$$

это обычная однослойная сеть



потенциальные проблемы:

- **забывание**

должны помнить начало последовательности

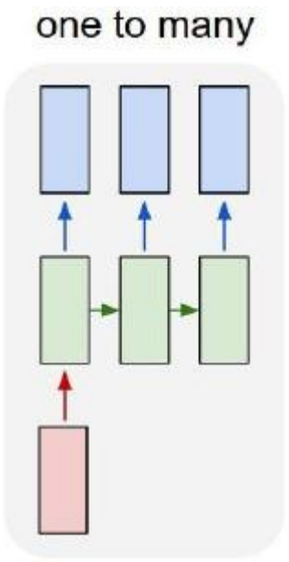
- **градиенты**

дальше разберём

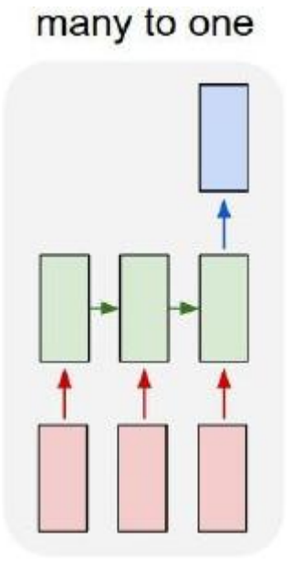
здесь на вход посл-ть и на выходе посл-ть

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

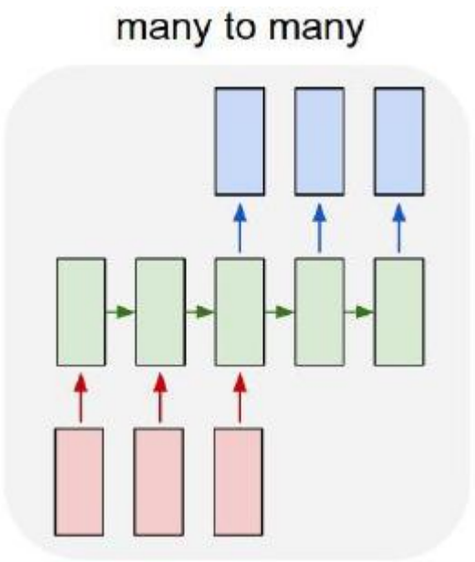
Применение RNN



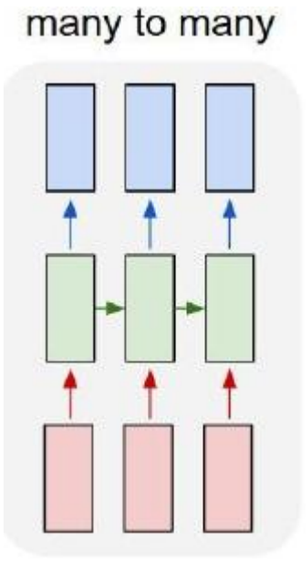
описание изображения



тема / настроение текста



машинный перевод



классификация фреймов видео

Можно по-разному собирать блоки –
для решения разных задач

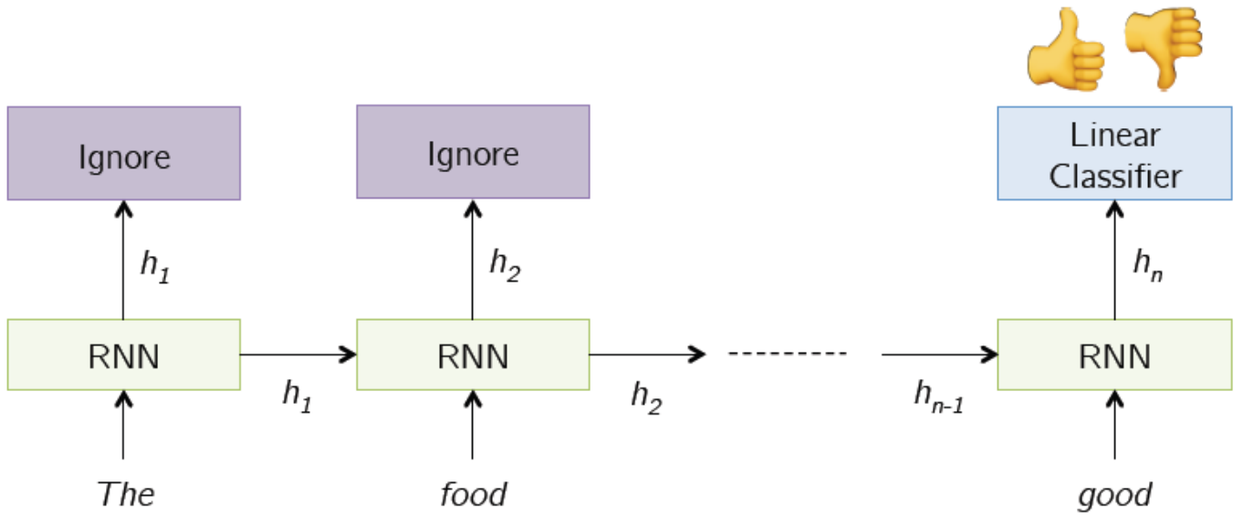
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN: как решать задачи классификации

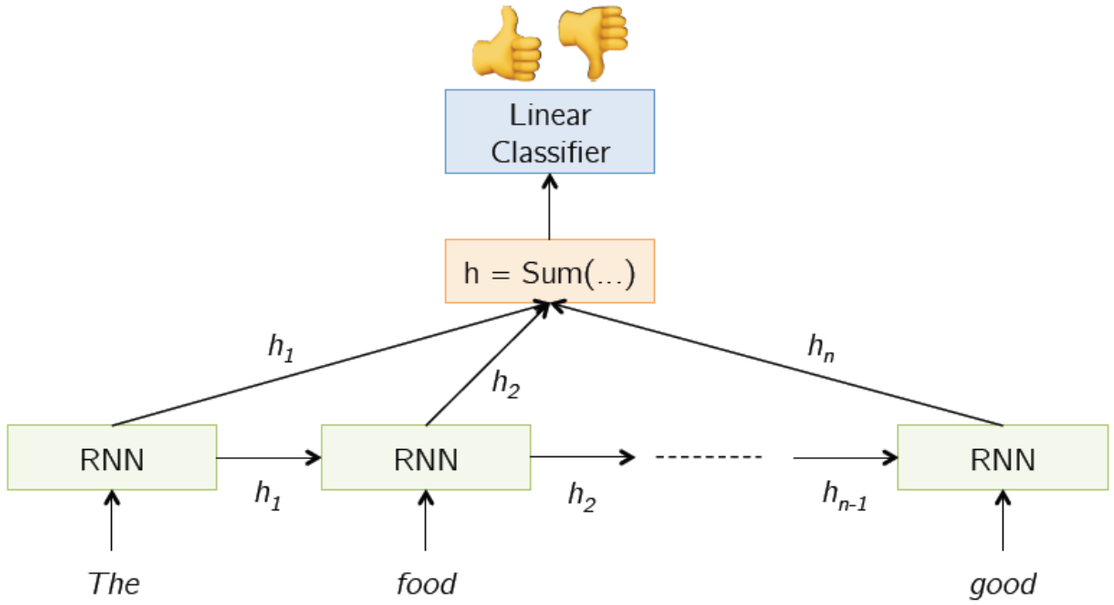
пример: тональность сообщения

нужен выход фиксированной длины

Первый способ



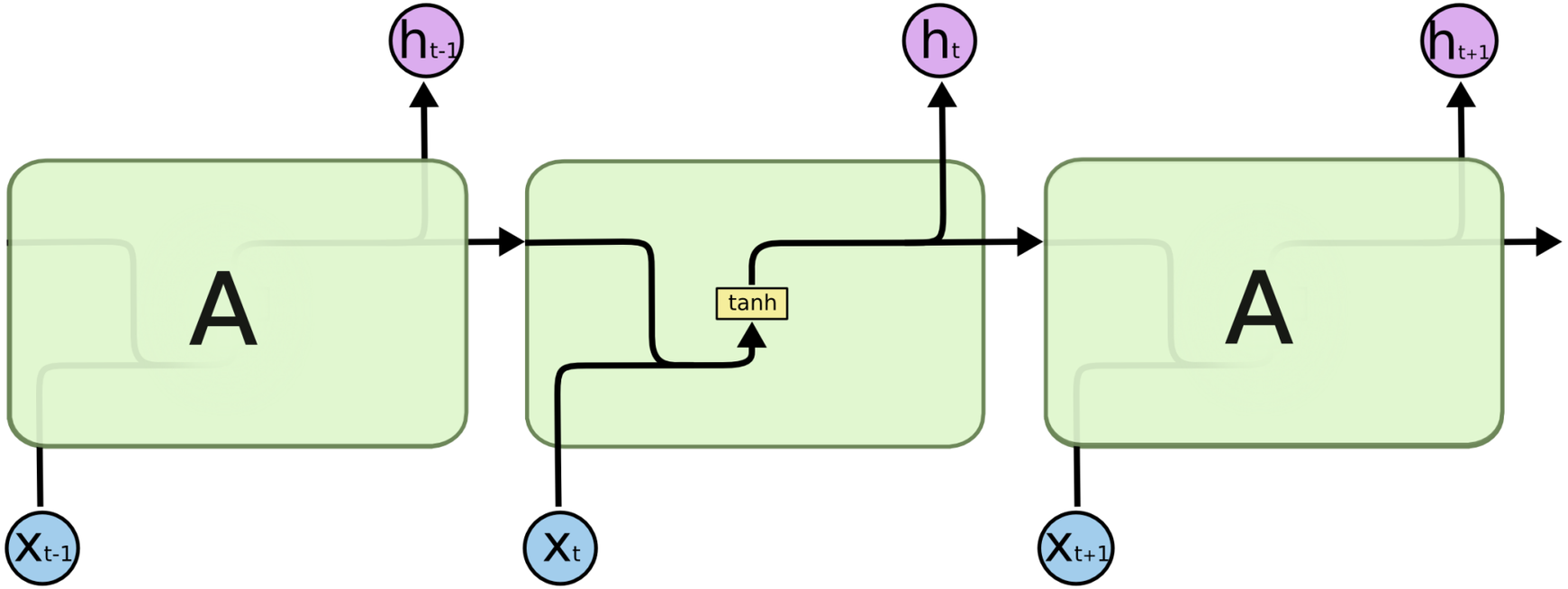
Второй способ



классификатор может быть устроен сложнее (k-слойный)

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

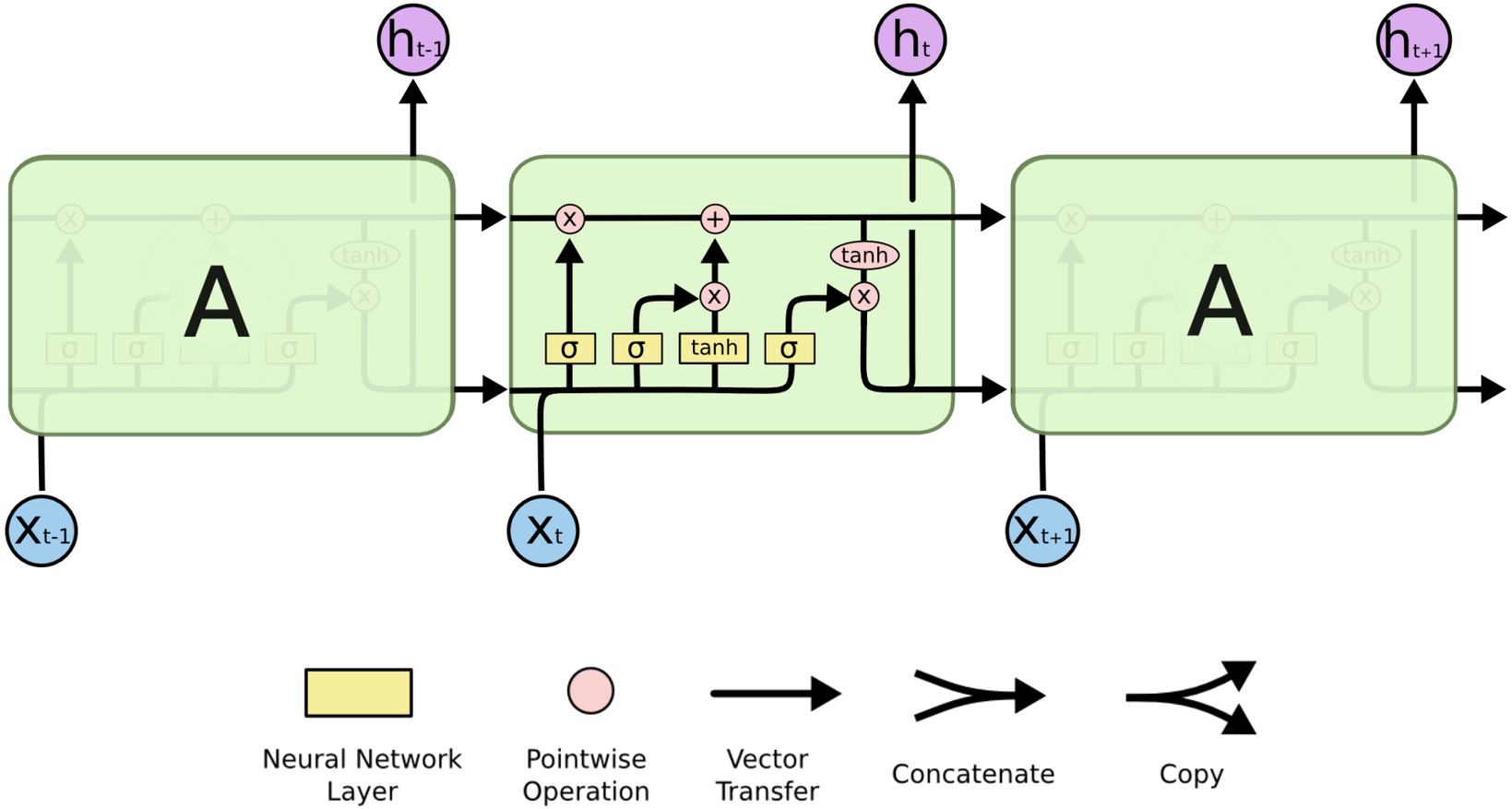
Стандартная RNN



LSTM (Long Short Term Memory)

здесь другой базовый блок (не обозначено умножения на матрицы):

два состояния – cell state / hidden state (выход ячейки)

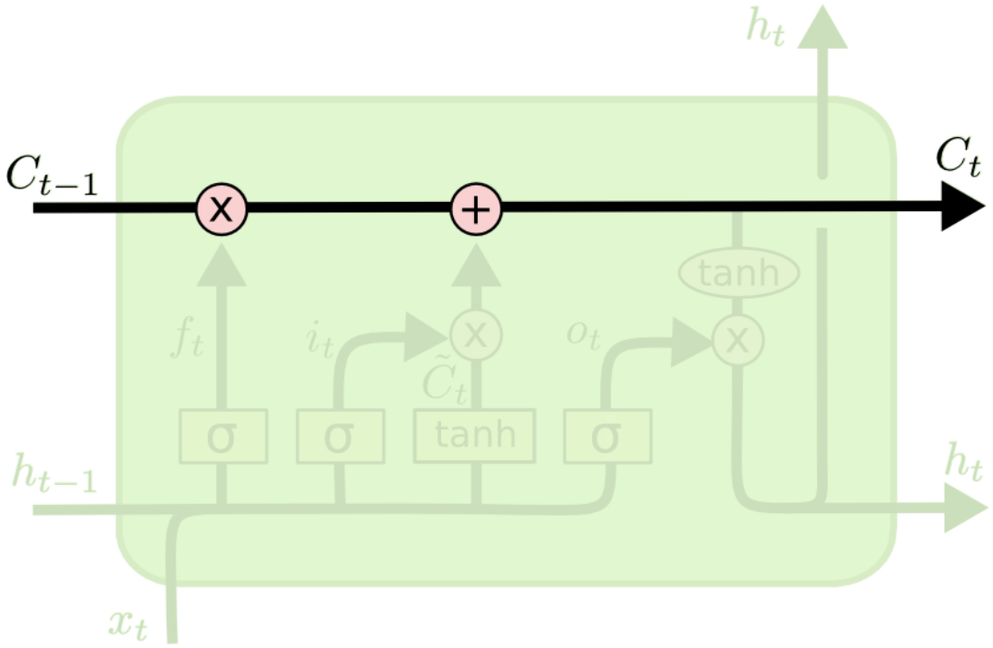


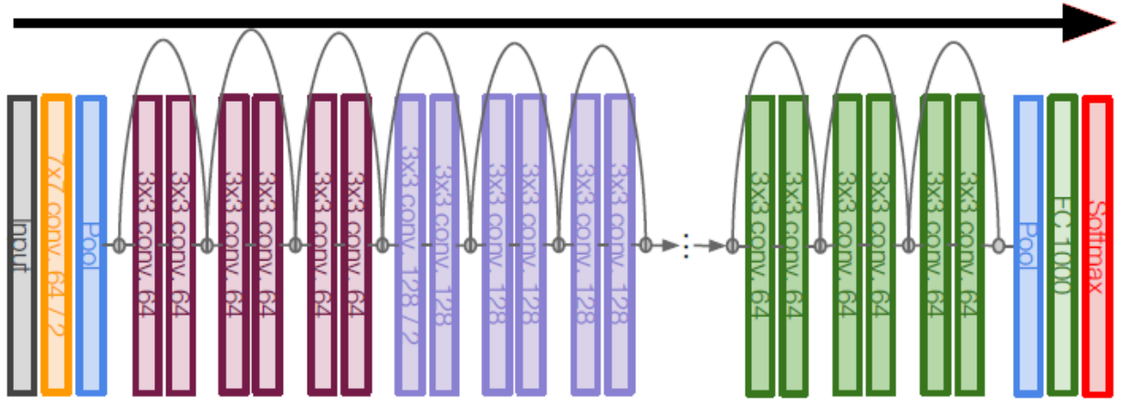
S. Hochreiter, J. Schmidhuber «Long short-term memory» // Neural Computation — 1997. — V. 9, № 8, P. 1735—1780.

Ключевая идея LSTM

«состояние ячейки/блока» – проходит через все блоки

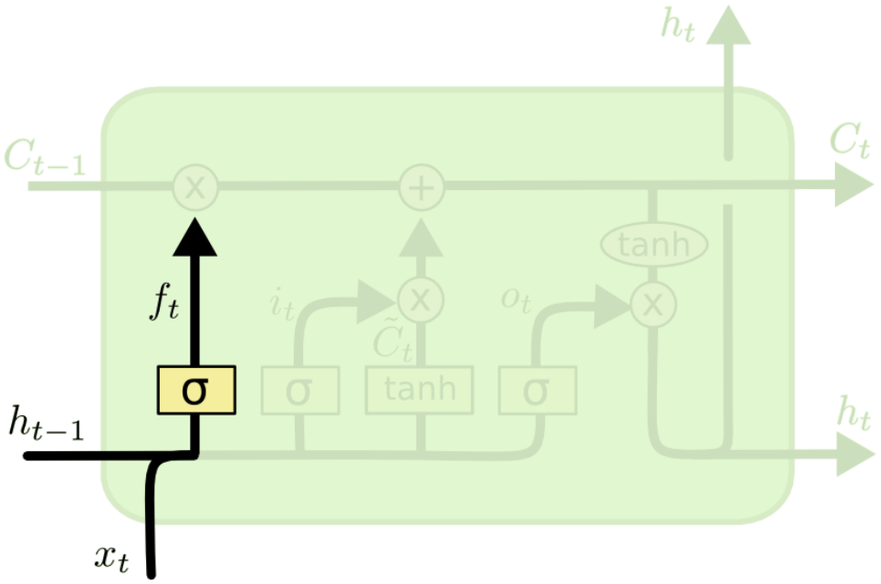
«shortcut connection»





- **память** – перенос информации, которая должна «слабо меняться»
- **борьба с затухающим градиентом** свободно протекает, как в ResNet

Забывающий гейт (Forget Gate)



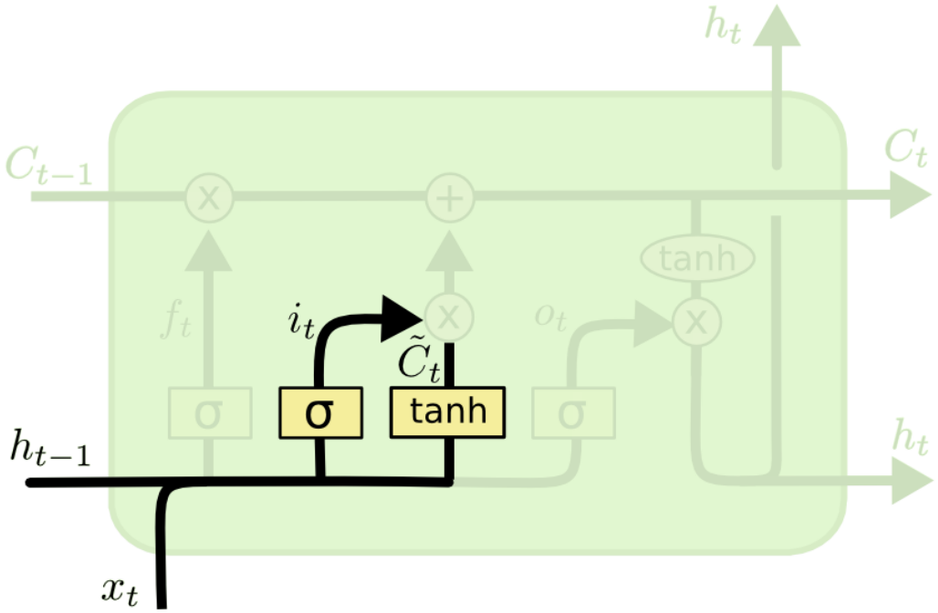
$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f)$$

если = 1 – передаём полностью состояние блока

если = 0 – то забываем предыдущее состояние

строго равенства не будут выполняться

Входной гейт (Input Gate)



входной гейт:

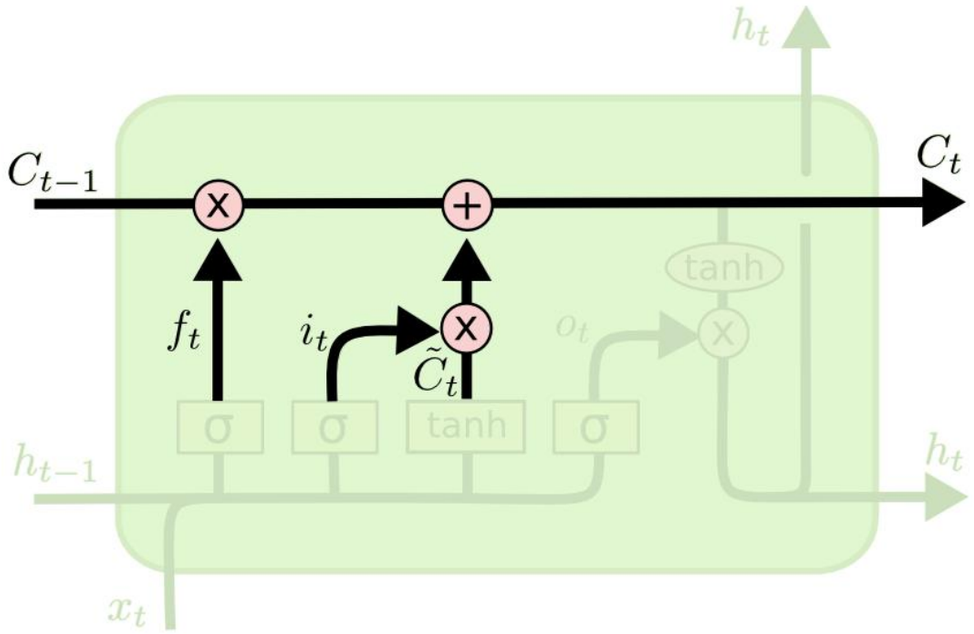
$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)$$

текущее состояние:

$$\tilde{C}_t = \tanh(W_C[h_{t-1}; x_t] + b_C)$$

Какую новую информацию учитываем в состоянии…

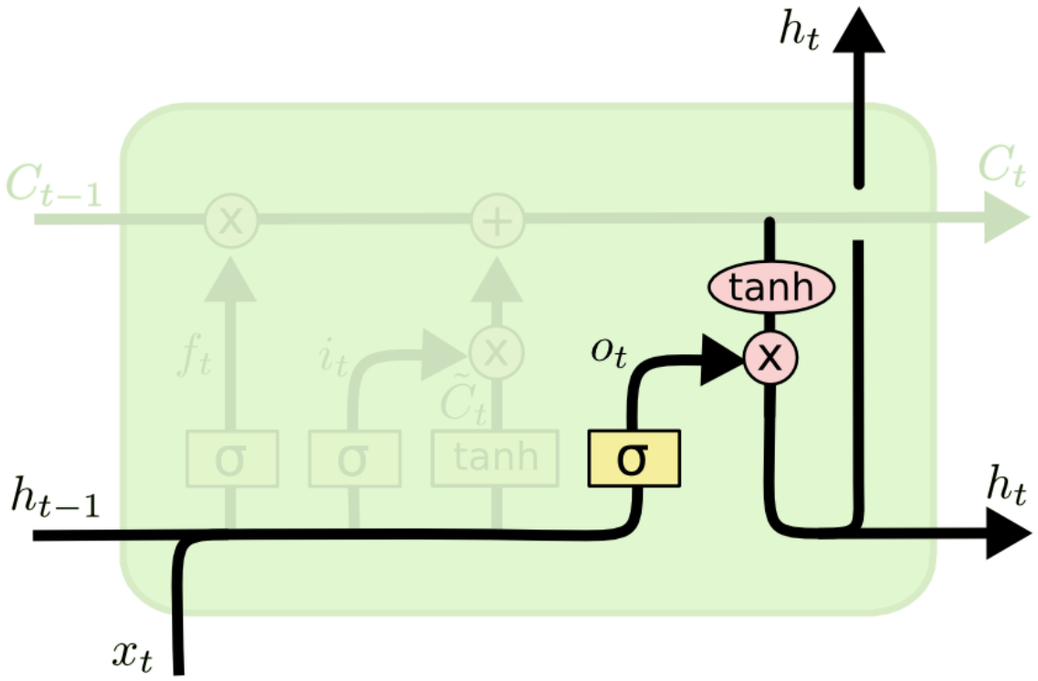
Обновление состояния (Cell update)



$$C_t = f_t C_{t-1} + i_t \tilde{C}_t$$

новое состояние = (старое состояние | гейт) + (посчитанное состояние | гейт)

Выходной гейт (Output Gate)



выходной гейт:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

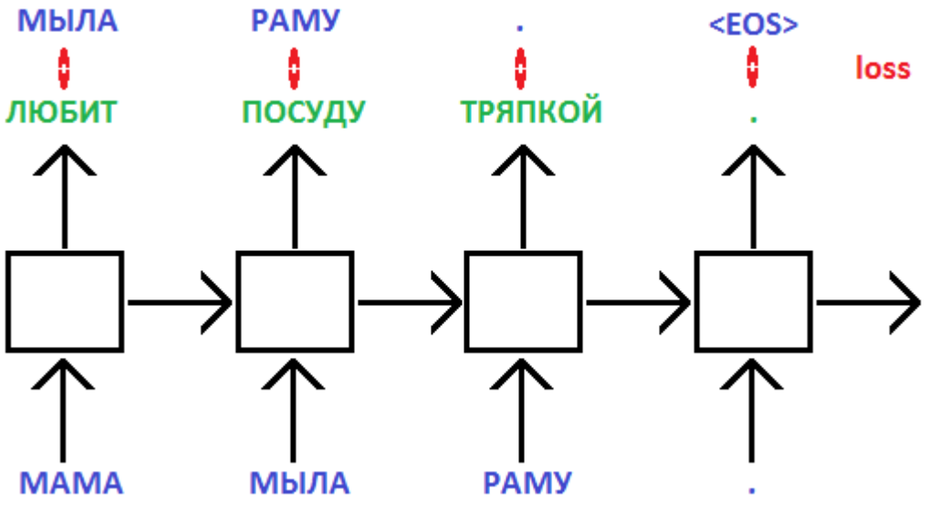
скрытое состояние:

$$h_t = o_t \tanh(C_t)$$

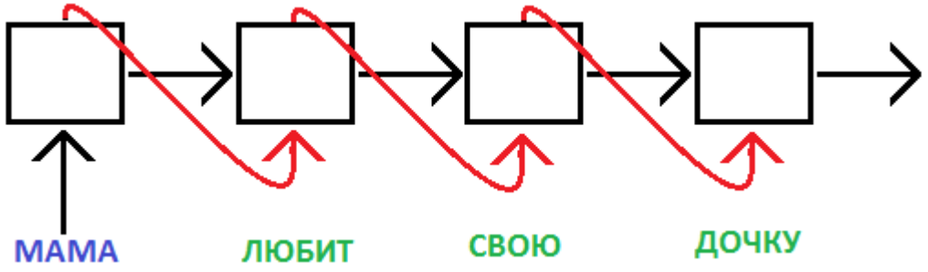
Приёмы обучения: метод форсирования учителя (teacher forcing)

как обучать модель, которая предсказывает следующий элемент последовательности

обучение



тест



применяются не только в RNN, но и в трансформерах

Приёмы обучения: метод форсирования учителя (teacher forcing)

Вместо выхода модели на предыдущем шаге подаём истинную метку

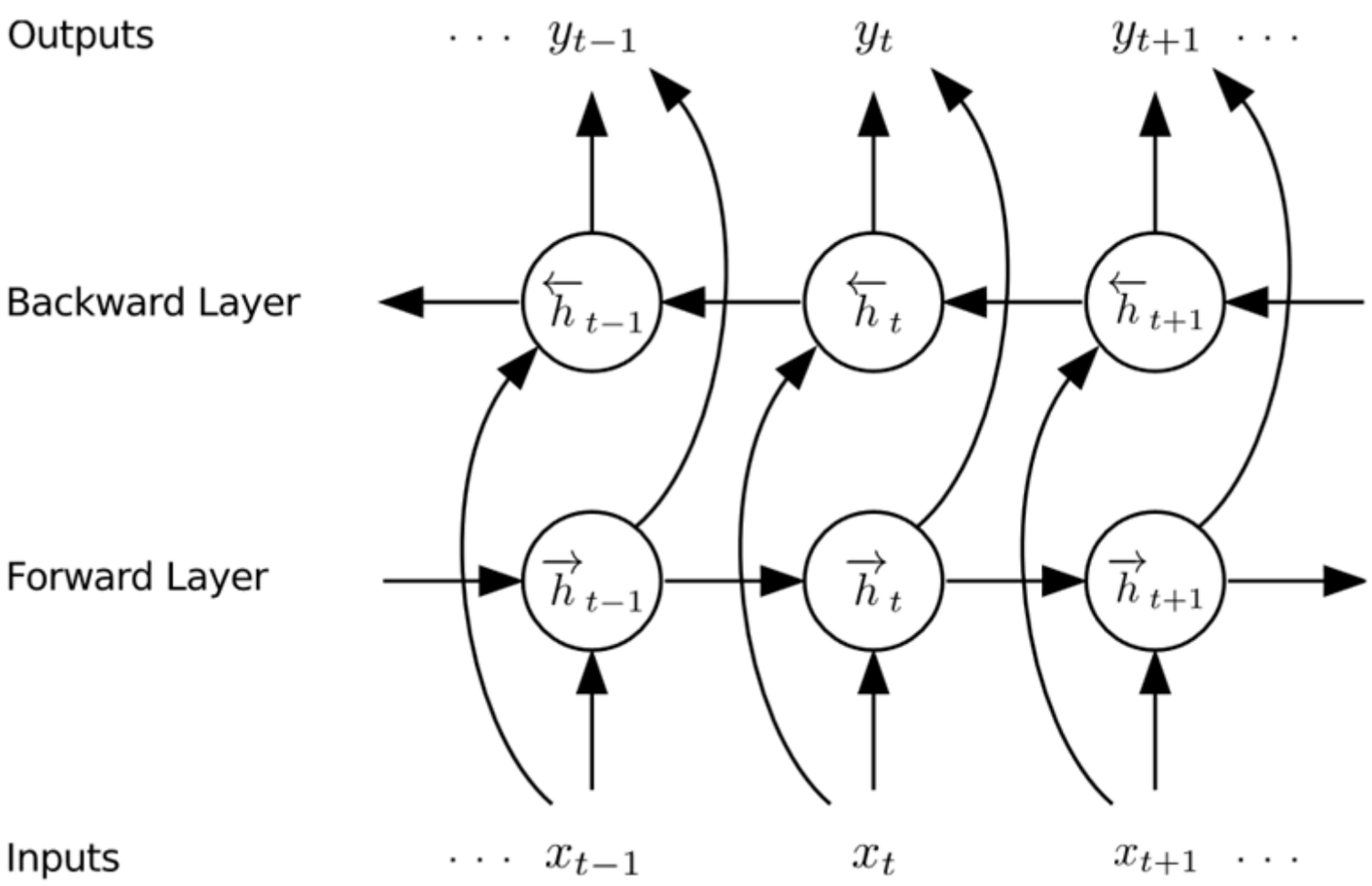
\+ быстрее сходится

\+ можно использовать для предтренировки

– то что видит при тестировании и обучении может отличаться

– накапливается ошибка

Двунаправленные (Bidirectional) RNN



$$\vec{h}_t = \sigma(\vec{W}_{hh}\vec{h}_{t-1} + \vec{W}_{hx}x_t)$$

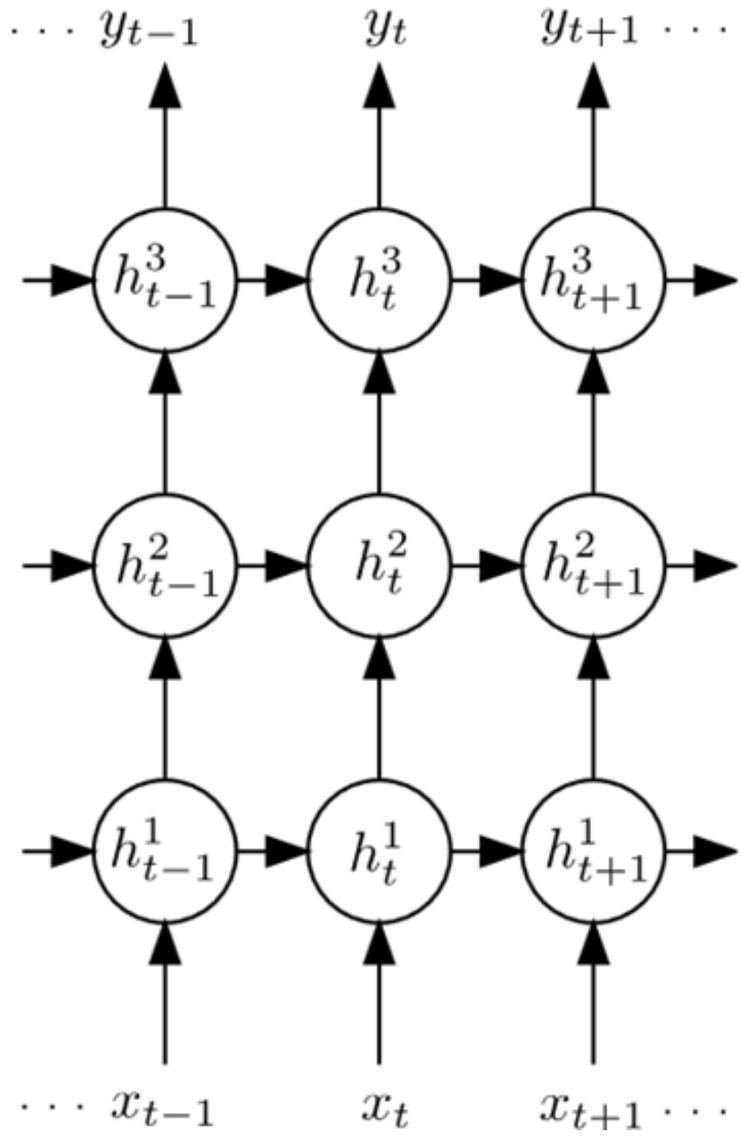
$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}_{hh}\overleftarrow{h}_{t+1} + \overleftarrow{W}_{hx}x_t)$$

$$y_t = \vec{W}_{yh}\vec{h}_t + \overleftarrow{W}_{yh}\overleftarrow{h}_t + b_y$$

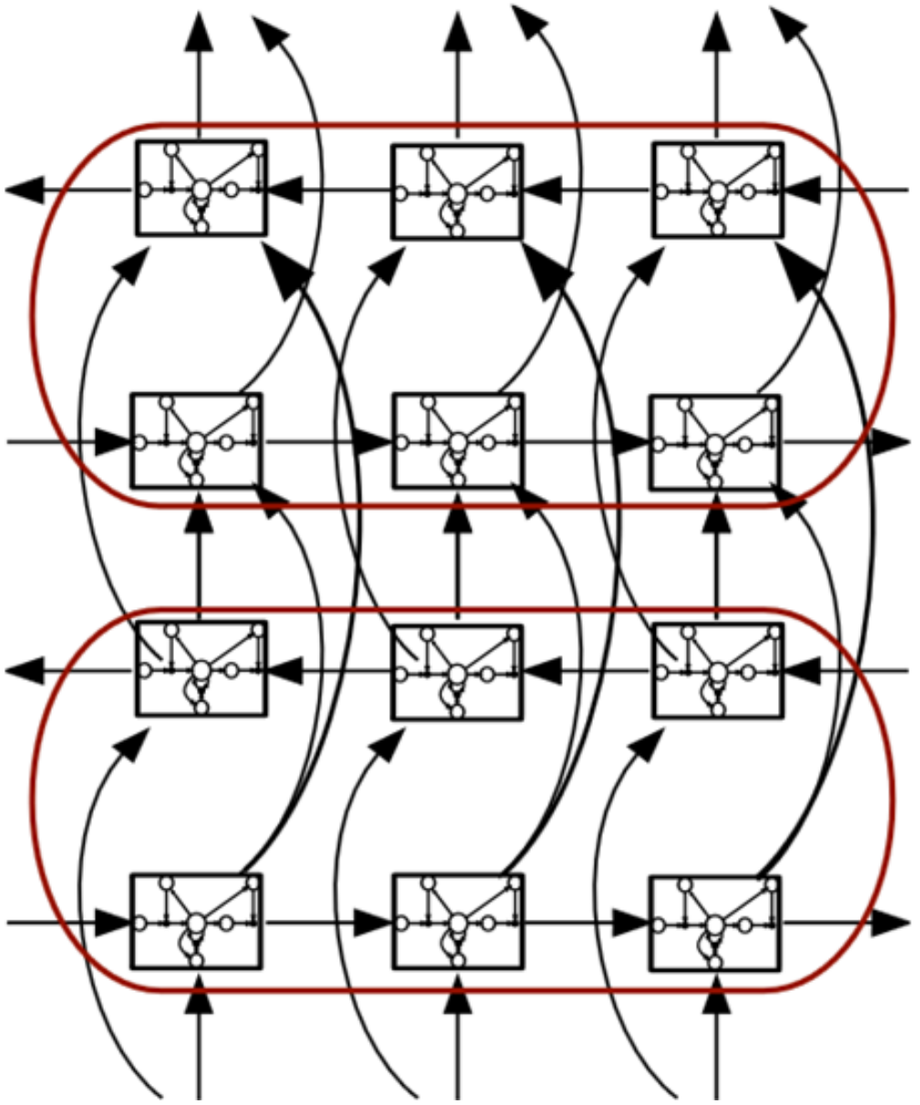
нужна вся последовательность (не всегда есть)

распознавание рукописных текстов, распознавание речи, биоинформатика

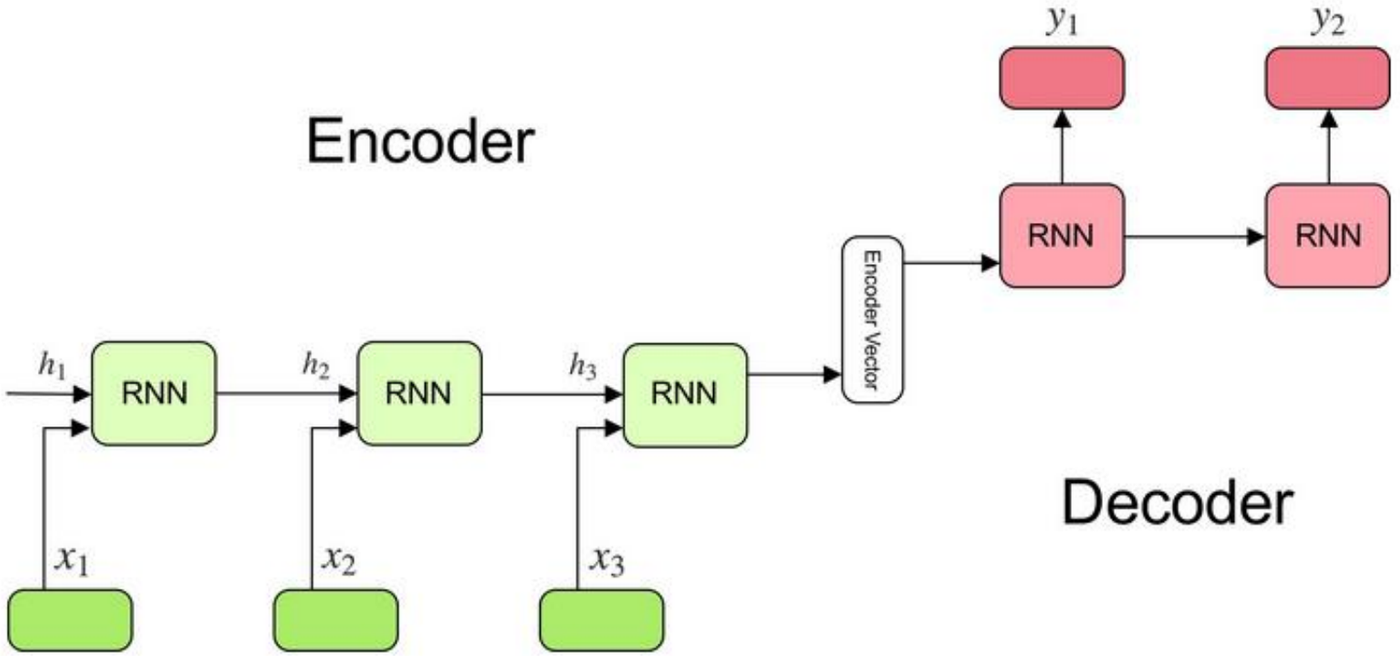
Глубокие (Deep) RNN / Multi-layer RNN / stacked RNN



Глубокие двунаправленные (Bidirectional) RNN



Модель seq2seq



http://www.davidsbatista.net/blog/2020/01/25/Attention-seq2seq/

Sutskever I. «Sequence to Sequence Learning with Neural Networks», 2014 // https://arxiv.org/abs/1409.3215

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho et. al. «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation» https://www.aclweb.org/anthology/D14-1179/

Модель seq2seq: как переводить последовательность → последовательность

Многослойная (4 слоя) LSTM

размерность представления = 1000

входной словарь = 160,000

выходной словарь = 80,000

кодировщик (encoder) – декодировщик (decoder)

кодировщик: входная последовательность → вектор

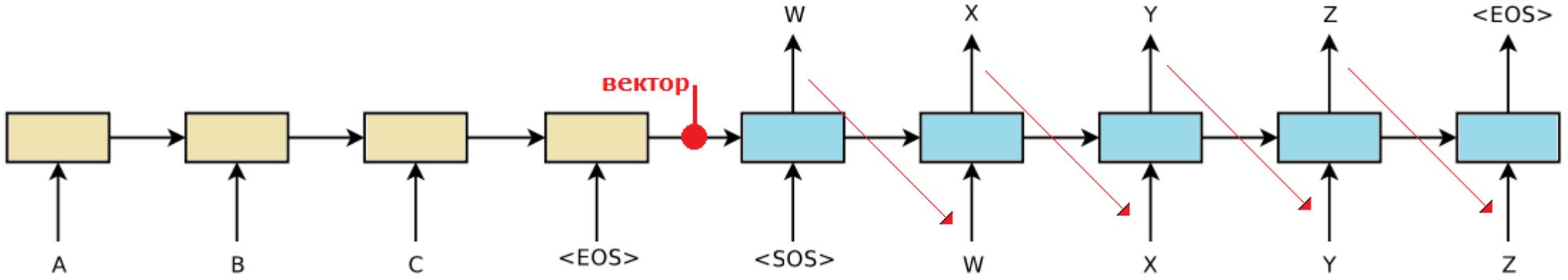
декодировщик: вектор → таргетная последовательность

Это разные LSTM, у них разные параметры!

Интересно: в задаче перевода качество повышало инвертирование порядка входа!

Модель seq2seq

здесь декодировщик называют также **языковой моделью**



при работе (inference) – подаём на вход сгенерированное
при обучении – среднее ошибок на всех выходах (ex negative log prob)

тонкости:
на рисунке в декодировщике передаётся только его внутреннее состояние
выход кодировщика передаётся лишь первому элементу
можно его передавать всем! Чтобы информация о входе была у всех

Модель seq2seq

Внутреннее представление предложений!

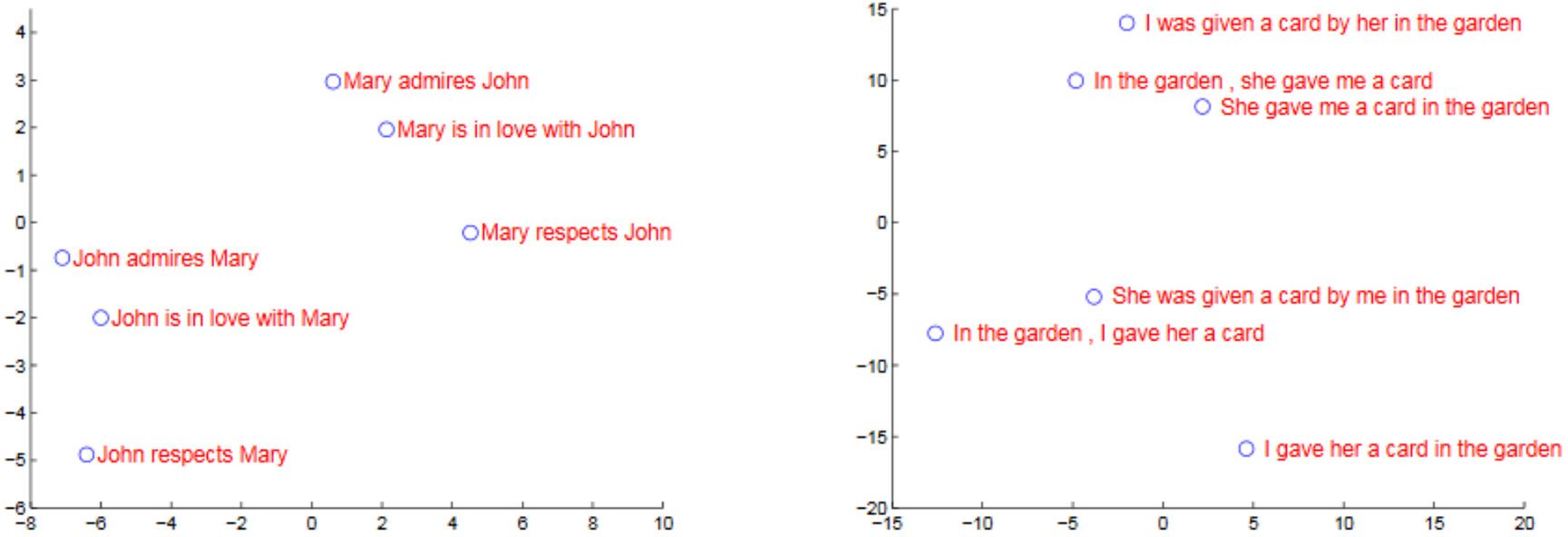


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

left-to-right beam-search decode

если выбираем лучшего следующего, не обязательно максимизируем качество

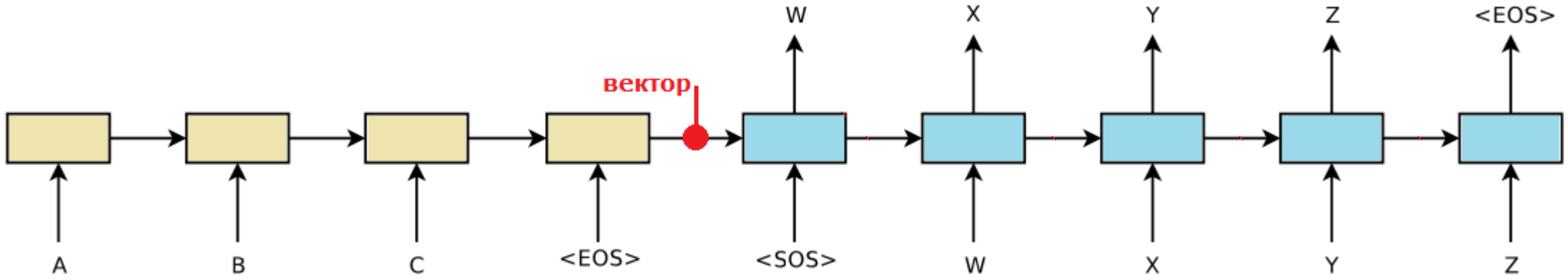
Обучение 10 дней

Тоже хороши ансамбли

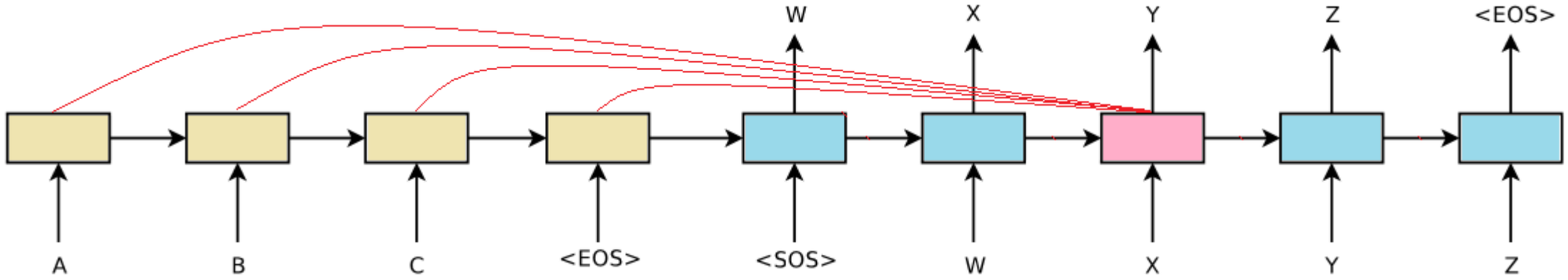
Обобщения seq2seq

На одном нейроне вся информация о тексте… плохо

(особенно для длинных последовательностей)



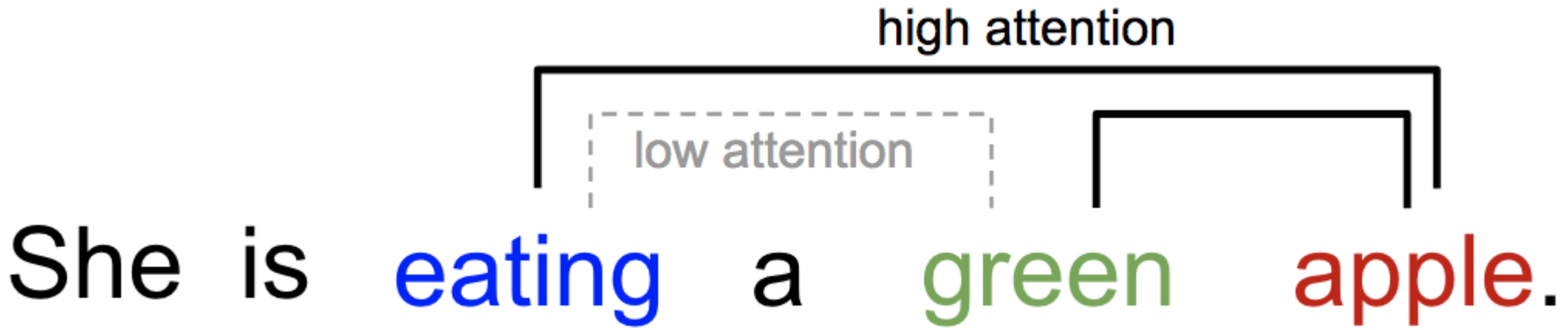
Решение – механизм внимания



Bahdanau et al. 2015 «Neural Machine Translation by Jointly Learning to Align and Translate» // ICLR 2015 https://arxiv.org/pdf/1409.0473.pdf

Механизм внимания

Концепция: есть взаимосвязи между словами



кодировщик передаёт в декодировщик не только одно состояние,
а состояния всех токенов!

– но для этого нужен механизм пулинга посл-ти состояний любой длины

выход – пулинг по схожести

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#born-for-translation

Механизм внимания

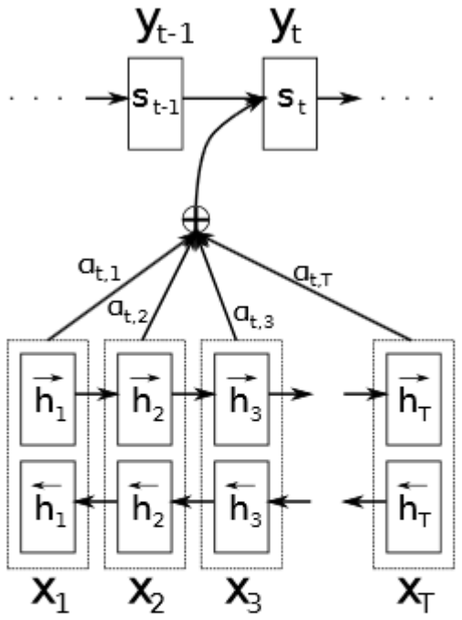


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Не будем пытаться закодировать всё предложение одним вектором!

Добавляется контекстный вектор (конкатенируется)

$$c_i = \sum_j \alpha_{ij} h_j$$

веса (softmax)

$$\alpha_{ij} = \exp(e_{ij}) / \sum_k \exp(e_{ik})$$

Насколько соответствуют состояния

$$e_{ij} = a(s_{i-1}, h_j)$$

Учитываются не только слова ДО, но и ПОСЛЕ!

Конкатенация состояния ДО и состояния ПОСЛЕ

Bidirectional RNN (BiRNN)

Механизм внимания: получаем интерпретацию и выравнивание (alignment)

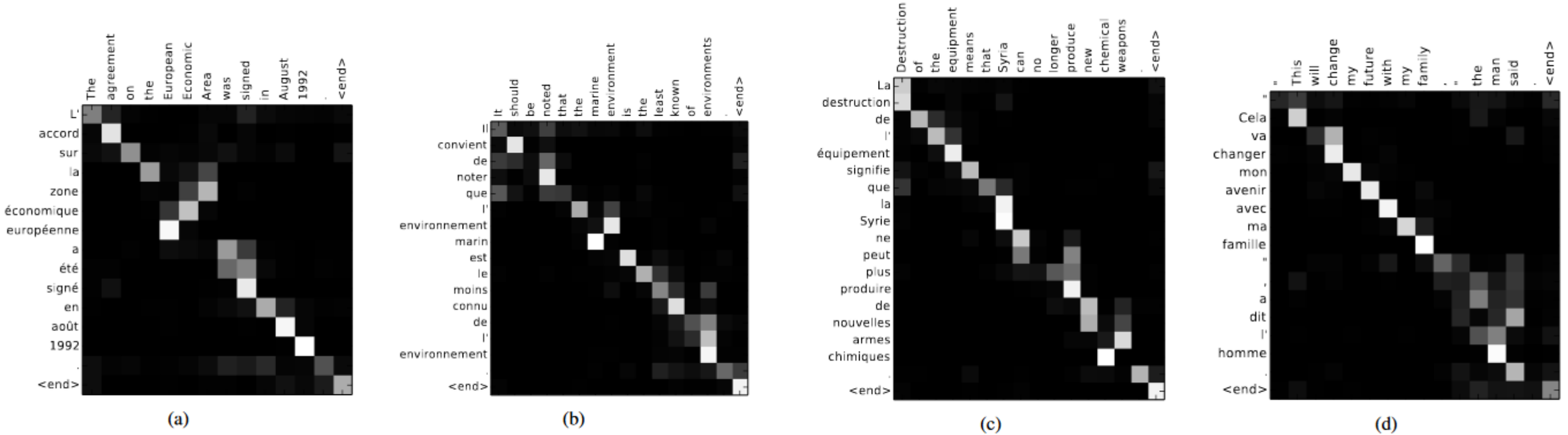


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Bahdanau D. и др. «Neural Machine Translation by Jointly Learning to Align and Translate» // https://arxiv.org/abs/1409.0473

Механизм внимания: решение проблемы «узкого горла»

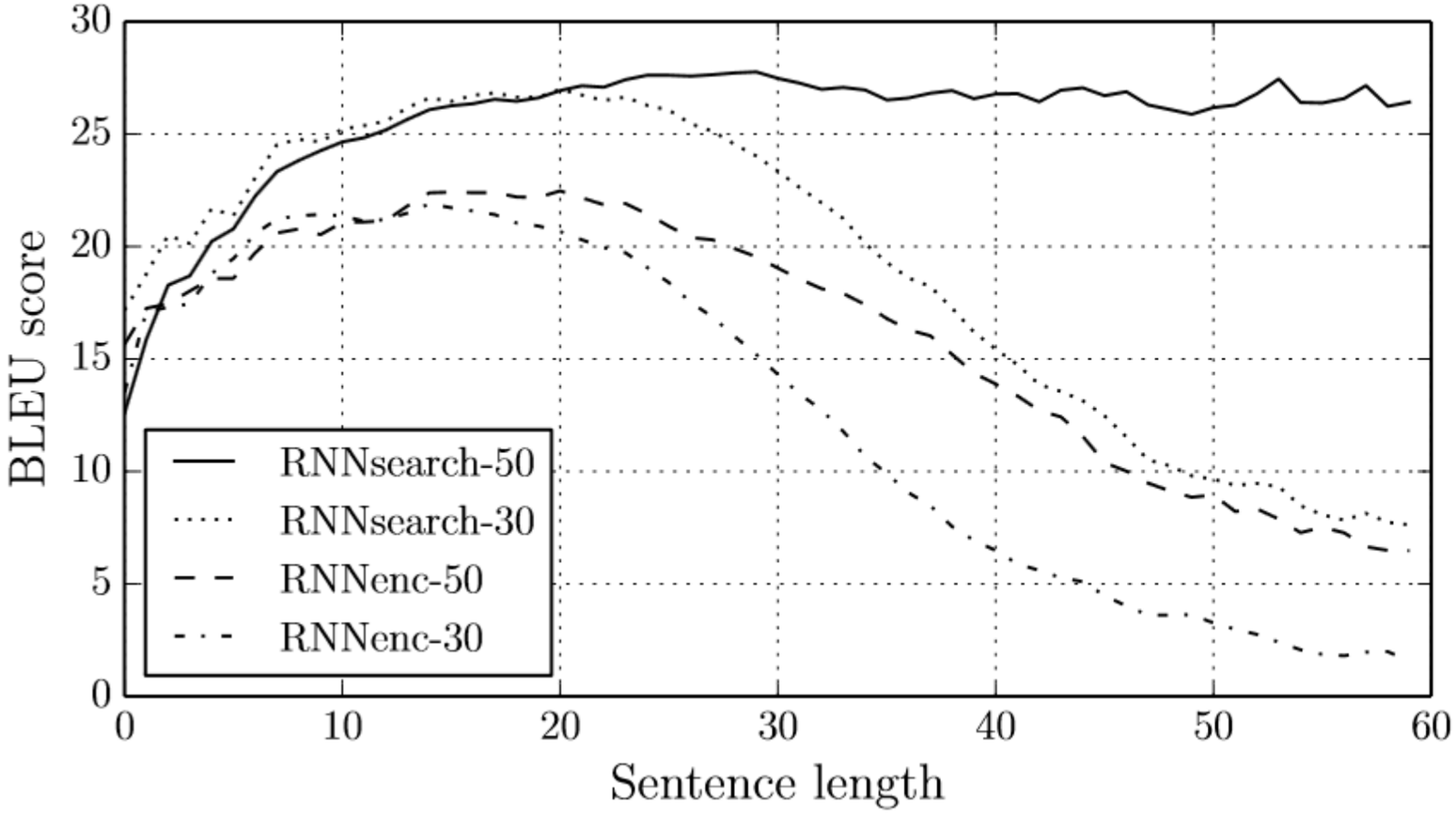
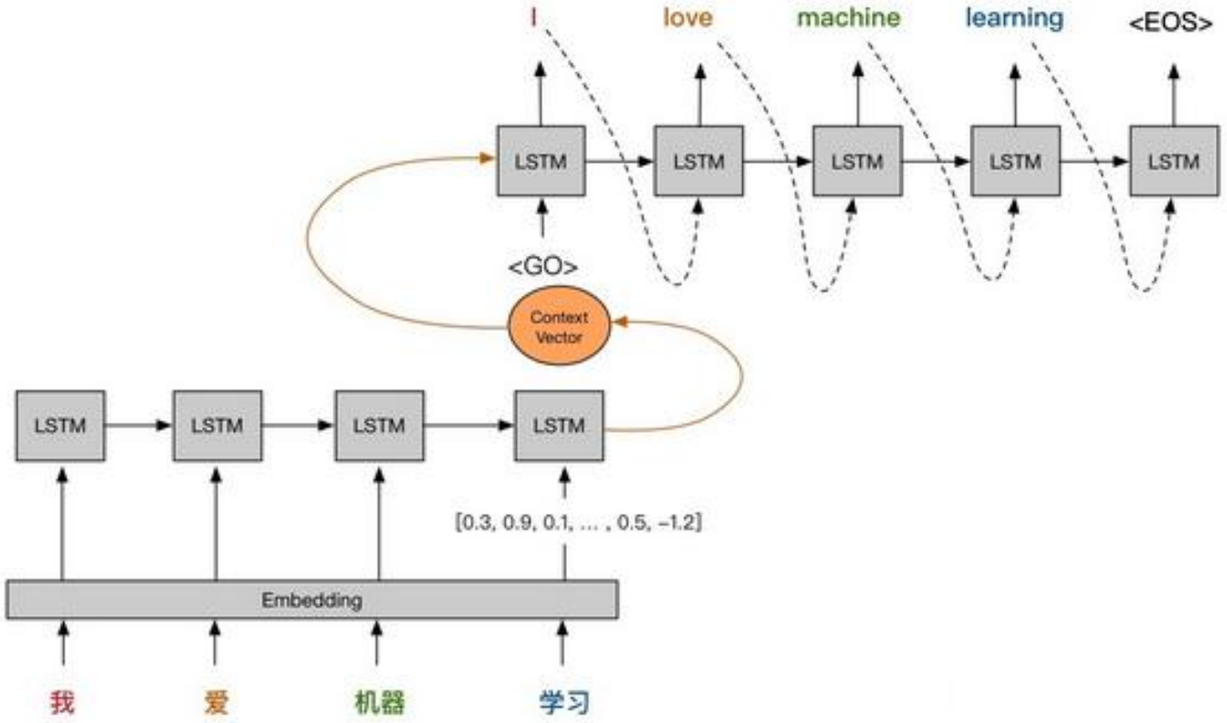
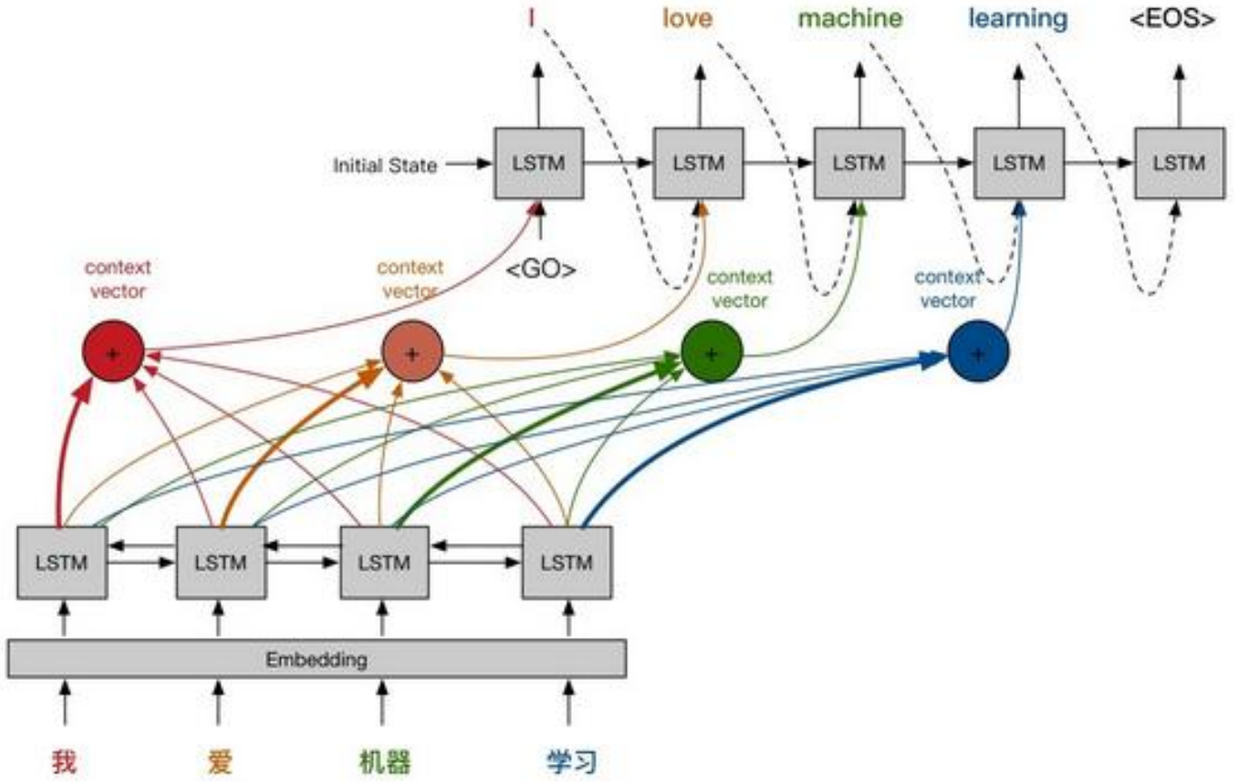


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

seq2seq vs attention





Внимание – техника вычисления взвешенной суммы значений (values) по запросу (query)

~ техника получения описания (representation) фиксированного размера по запросу

https://zhuanlan.zhihu.com/p/37290775

Плюсы механизма внимания (Attention)

- улучшает качество перевода (и не только)
  - решает проблему «узкого горла»
  - появляется интерпретируемость
- решает проблему «затухания сигнала» / исчезающего градиента
- получаем выравнивание (alignment) «бесплатно» в переводе

Итог

В изображениях свёртки – естественная операция

- классическая линейная операция
- поиск паттернов
- реализация фильтра
- разделение параметров
- реализация разреженных взаимодействий (sparse interactions)

Естественное устройство CNN: n×[conv + activ + pool] + k×FC

В отличие от классического CV не придумываем фильтры

Они обучаются сами!

Свёртка – первый пример разделения весов.

Есть способы экономии параметров – и ими пользуются!

Есть «обратные» (транспонированные) свёртки

Итог

Стандартная архитектура «кодировщик-декодировщик»

- сегментация
- машинный перевод

RNN «развёртываются» в вычислительный граф
но есть проблема с распространением градиента
LSTM – одно из решений

Механизм внимания позволяет обрабатывать большие последовательности