

Глубинное обучение

Лекция 3: Основные виды нейросетей (CNN и RNN)

Лектор: Антон Осокин


ФКН ВШЭ, 2019



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

План лекции

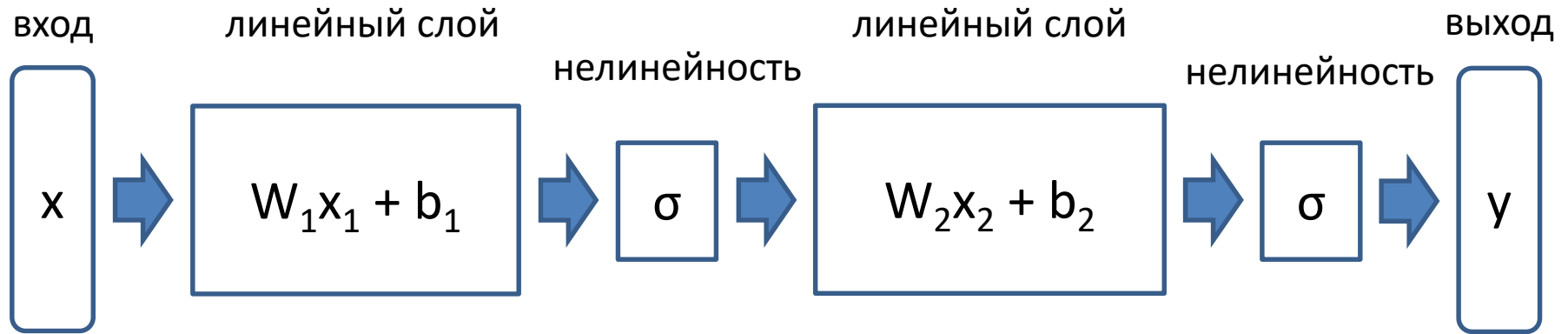
- CNN: Свёрточные нейросети
 - Основные операции: свёртка, пулинг
 - Основные архитектуры
 - LeNet, AlexNet, VGG,
 - Inception, ResNet, DenseNet
- RNN: Рекуррентные нейросети
 - Рекуррентные блоки
 - Виды моделей
 - LSTM, GRU



Как заставить
backprop
работать?

Чтобы градиенты
доходили!

Обычные нейросети



Недостатки:

- Слишком много параметров
 - Картинка 100 x 100 x 3 – первая размерность 30к
 - Сигнал 1000 x 10 – первая размерность 10к
- Быстро переобучаются, нужно много данных
- Как применять к входам разного размера?

Нужны более эффективные параметризации!

Свёрточные сети (ConvNet, CNN)

CNN – модель для данных с пространственной структурой (картинка – 2D, сигнал – 1D, видео – 3D)

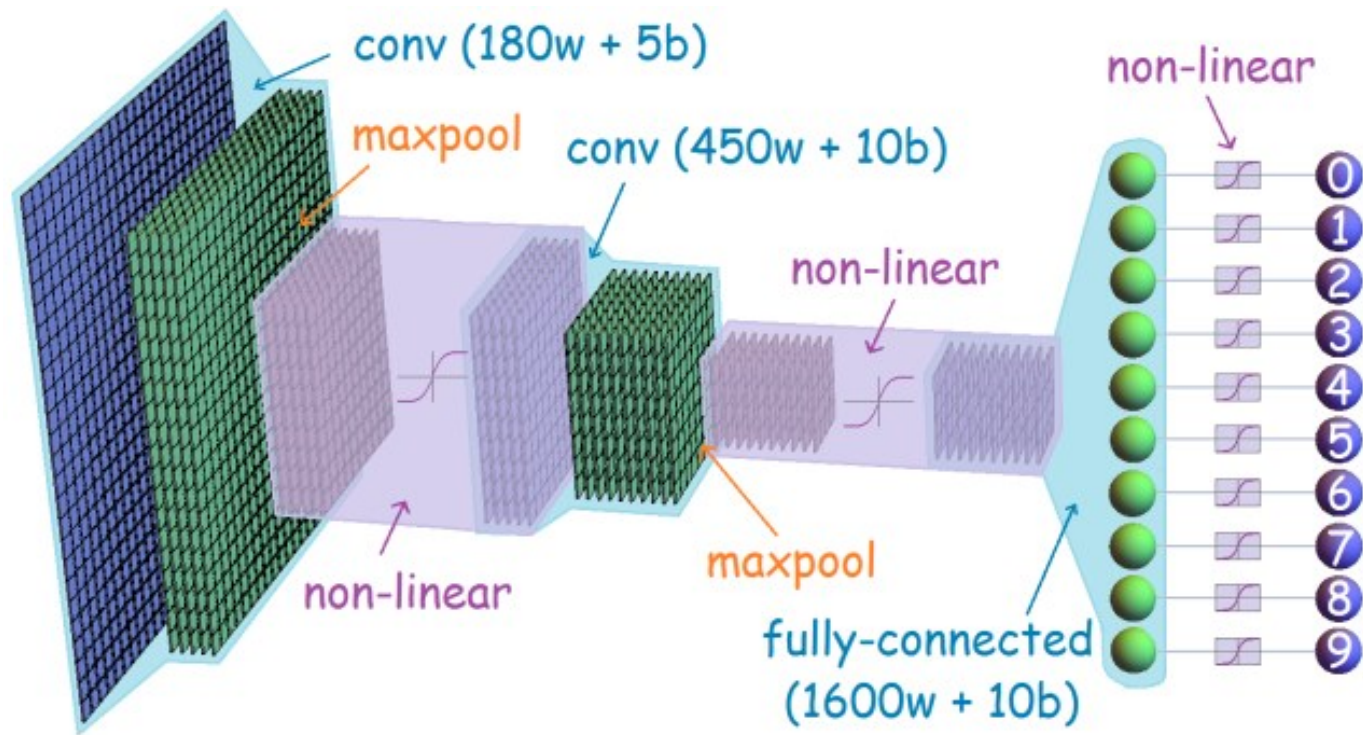
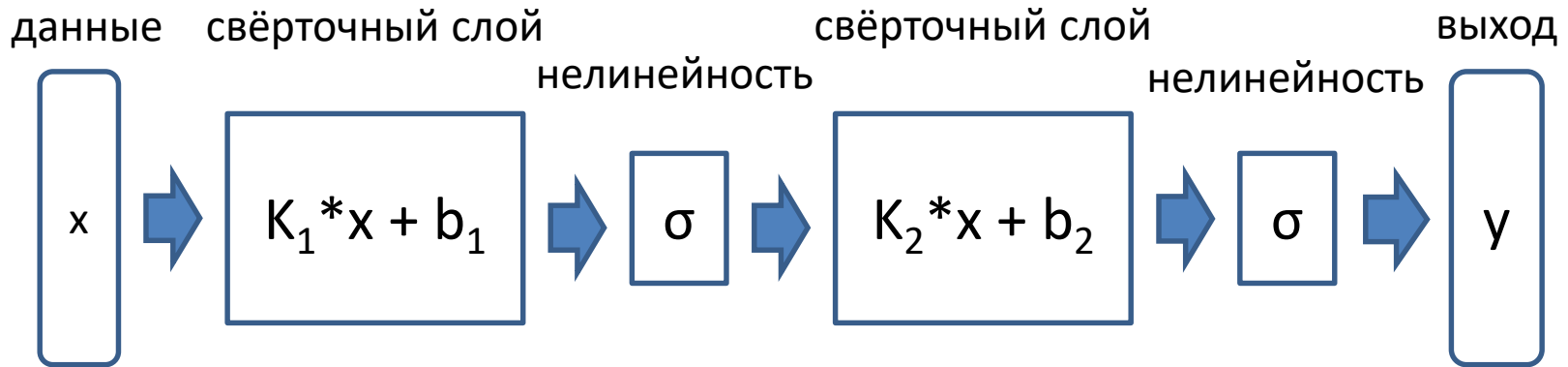


image credit: [apgoucher](#)

Свёрточные сети



- Свёртка (кросс-корреляция)

$$(K * x)(i, j, k) = \sum_{u=i-\delta}^{i+\delta} \sum_{v=j-\delta}^{j+\delta} \sum_c x(u, v, c) K(u - i + \delta, v - j + \delta, c, k)$$

- Гораздо меньше параметров (kernel, filter)
- Фильтры не зависят от размеров картинки

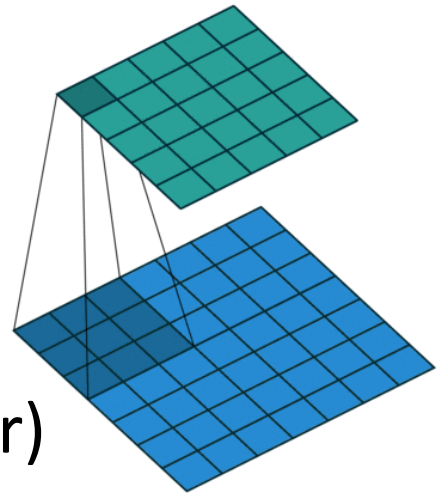
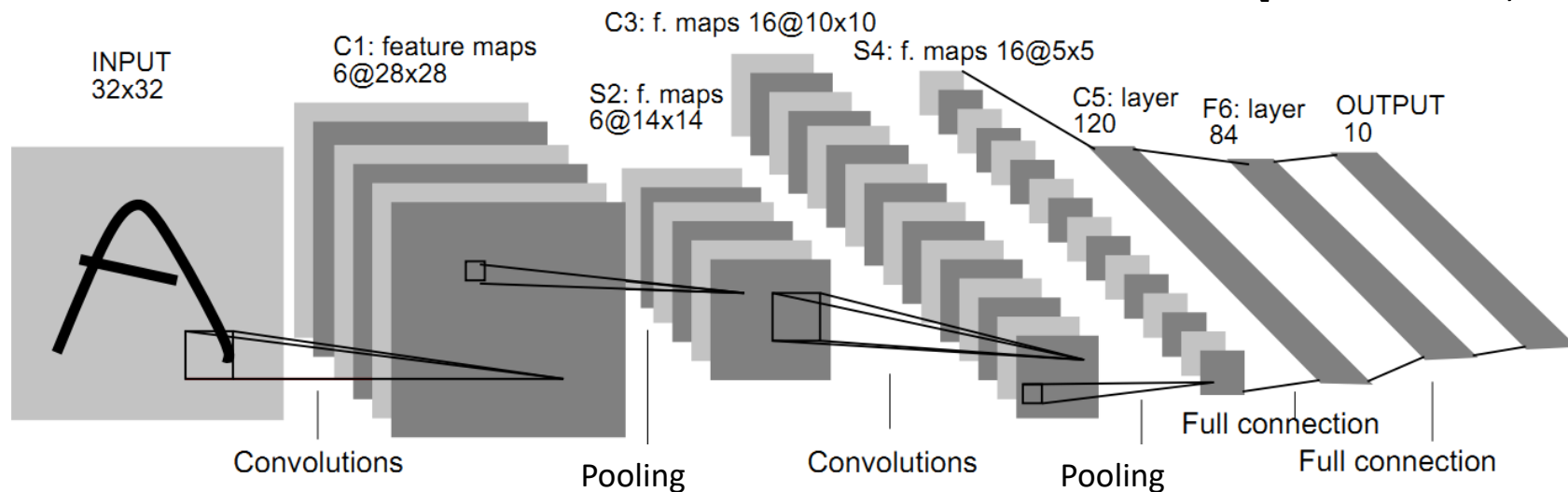


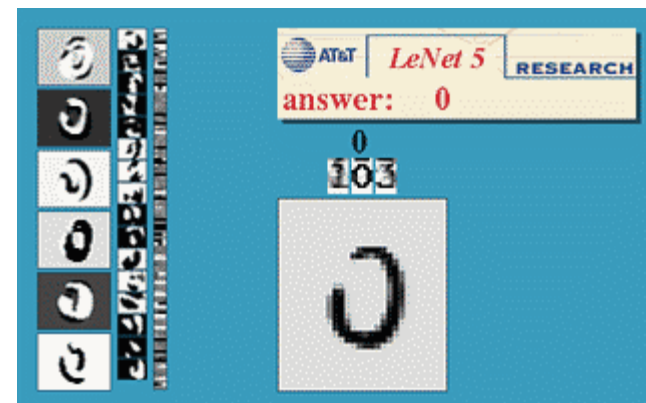
diagram: theano tutorials

Свёрточные сети: LeNet

[LeCun et al., 1998]



- Линейные операции – свёртки
- Пулинг: усреднение
- Нелинейность: сигмоида
- Есть полно-связные слои
- Успешны на MNIST



Свёртка (convolution)

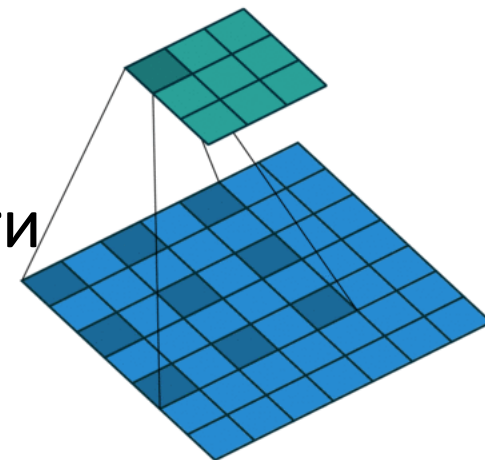
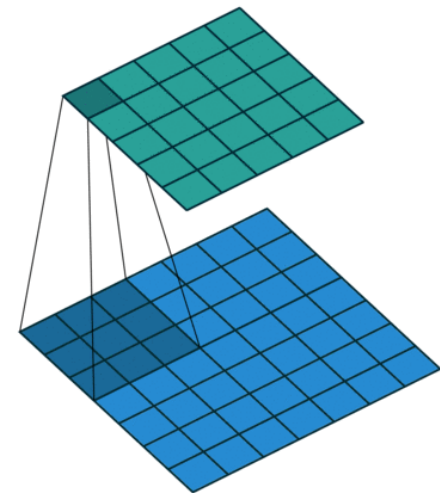
Свёртка (кросс-корреляция)

$$(K * x)(i, j, k) = \sum_{u=i-\delta}^{i+\delta} \sum_{v=j-\delta}^{j+\delta} \sum_c x(u, v, c) K(u - i + \delta, v - j + \delta, c, k)$$

Параметры (torch.nn.Conv2d):

- Количество каналов на входе и выходе
- Размеры ядра
- Смещение (stride) – большие значения понижают разрешение
- Padding (нет, нули, зеркальный)
- Dilation – увеличить область зависимости
- Размер выхода сети:

$$L_{out} = \text{floor}((L_{in} + 2pad - dil(kernel - 1) - 1) / stride + 1)$$



Реализация свёртки

Свёртка (кросс-корреляция)

$$(K * x)(i, j, k) = \sum_{u=i-\delta}^{i+\delta} \sum_{v=j-\delta}^{j+\delta} \sum_c x(u, v, c) K(u - i + \delta, v - j + \delta, c, k)$$

Свёртка – линейная операция

Основная идея – использовать матричное умножение:

$$\begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \end{pmatrix} * \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix} \Rightarrow \begin{pmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} k_1x_1 + k_2x_2 + k_3x_4 + k_4x_5 \\ k_1x_2 + k_2x_3 + k_3x_5 + k_4x_6 \\ k_1x_4 + k_2x_5 + k_3x_7 + k_4x_8 \\ k_1x_5 + k_2x_6 + k_3x_8 + k_4x_9 \end{pmatrix}$$

- Очень эффективные реализации: NVIDIA cuDNN, Intel MKL DNN
- Специализация: метод Винограда для свертки 3x3, преобразования Фурье для больших свёрток

Дифференцирование свёртки

Свёртка (кросс-корреляция)

$$(K * x)(i, j, k) = \sum_{u=i-\delta}^{i+\delta} \sum_{v=j-\delta}^{j+\delta} \sum_c x(u, v, c) K(u - i + \delta, v - j + \delta, c, k)$$

Свёртка – линейная операция

Производная – тоже матричное умножение

$$y = \begin{pmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{pmatrix} x$$



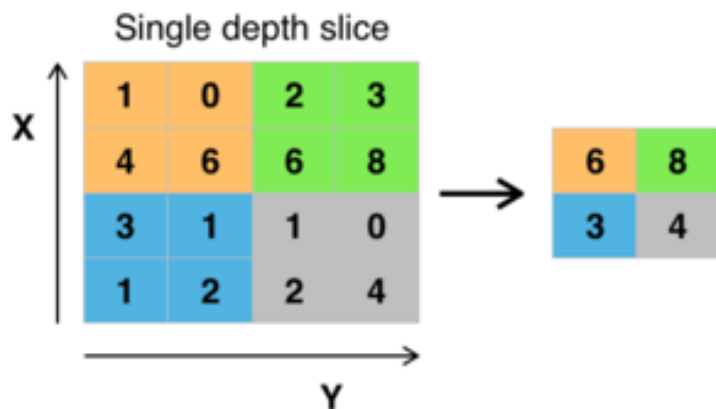
$$\frac{d\mathcal{L}}{dx} = \begin{pmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{pmatrix}^T \frac{d\mathcal{L}}{dy}$$

Операция upconvolution (conv-transpose) позволяет увеличить пространственное разрешение

Пулинг

Пулинг – агрегация признаков (max, sum)

Пулинг слой агрегирует соседние активации (пространственно или из разных фильтров)

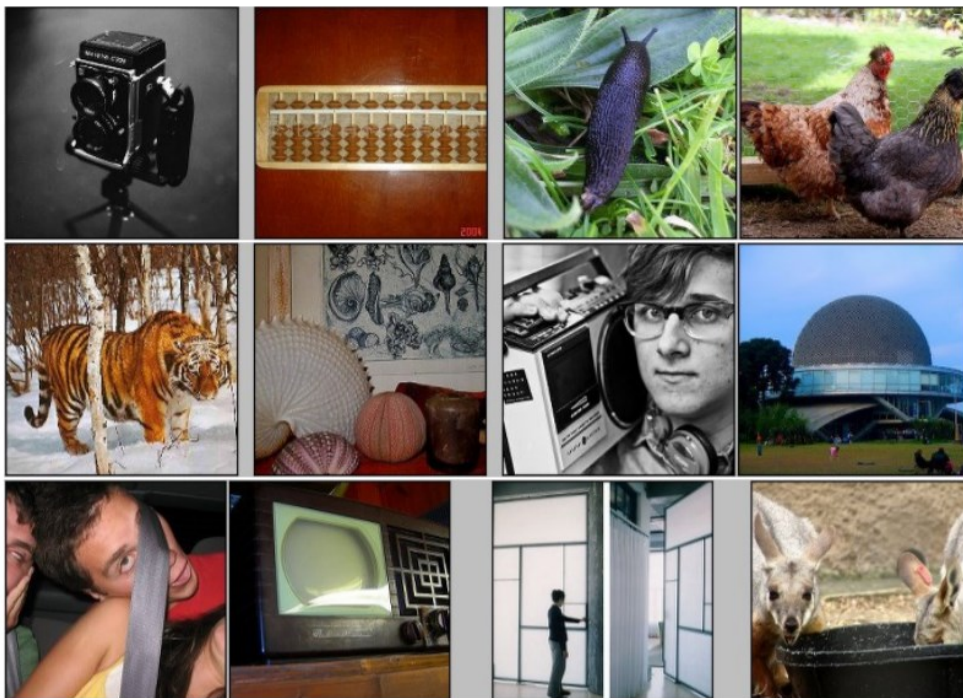


Max-pooling обеспечивает инвариантность к небольшим сдвигам (иногда полезно, иногда нет)

Дифференцирование – вернуть градиент в позиции максимумов

ImageNet challenge

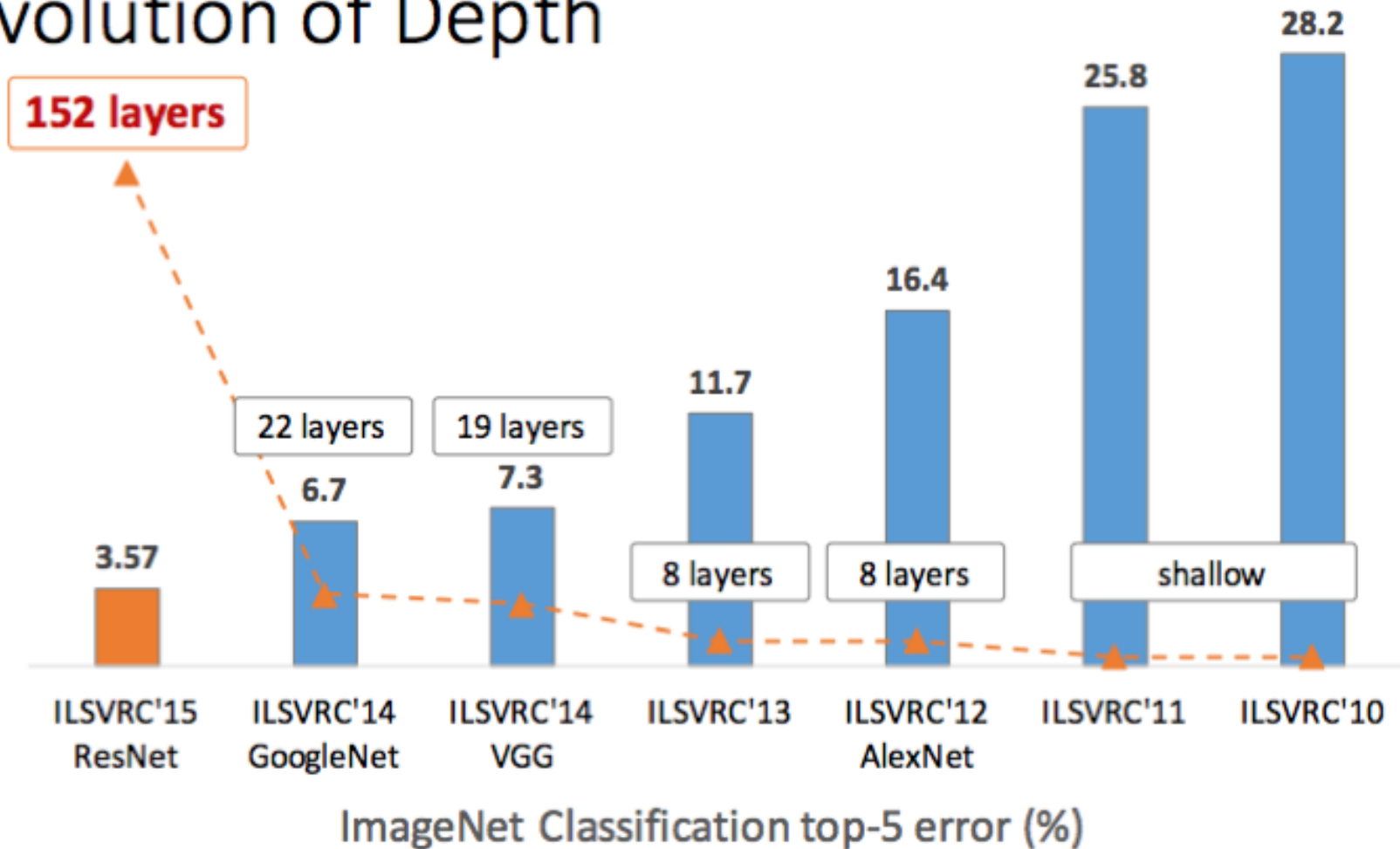
IM  GENET



- Классификация изображений
- 1.2М изображений
- 1000 классов
- Данные из интернета
- Аннотация при помощи Amazon MTurk

ImageNet challenge

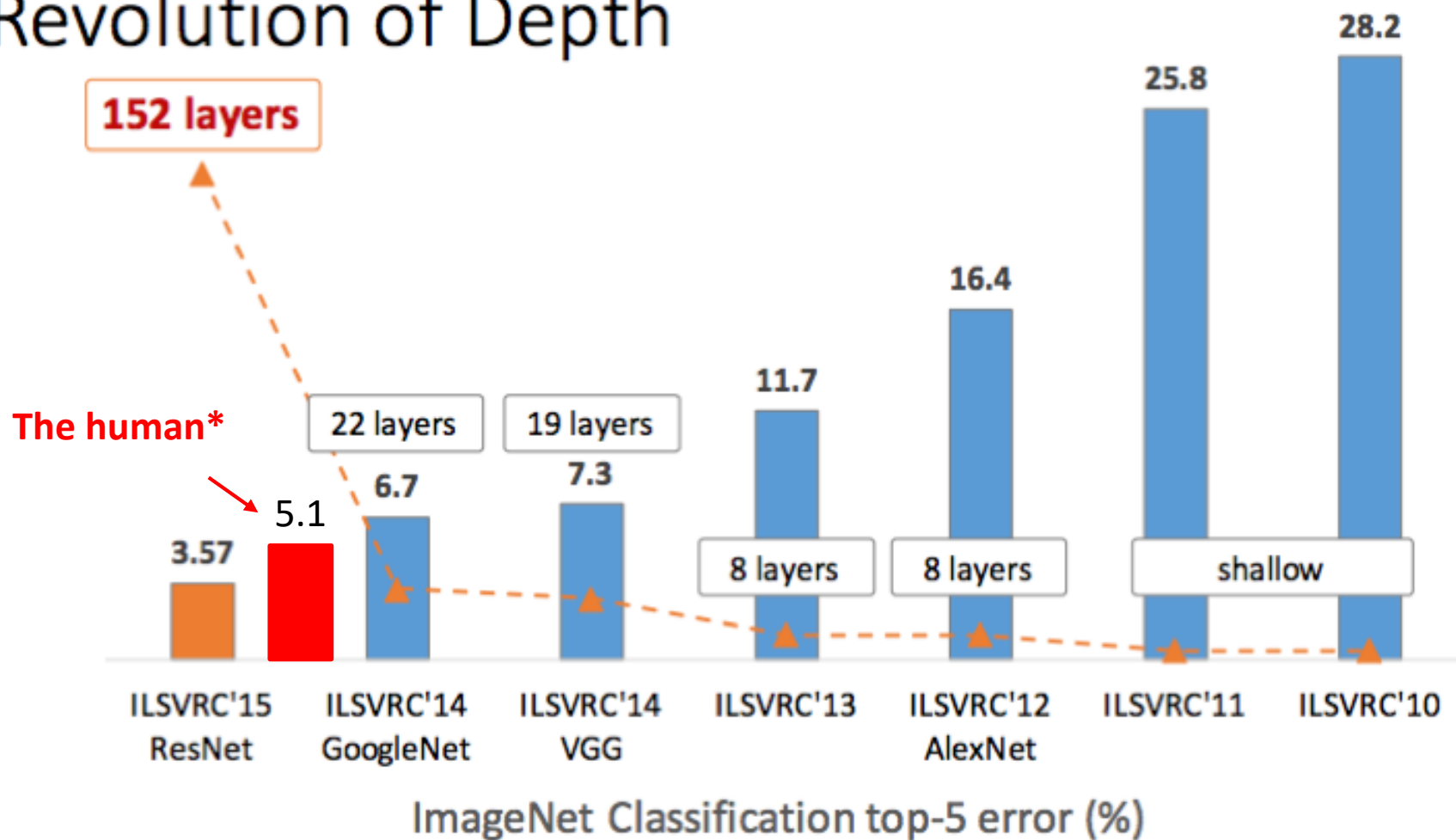
Revolution of Depth



plot credit: Kaiming He

ImageNet challenge

Revolution of Depth

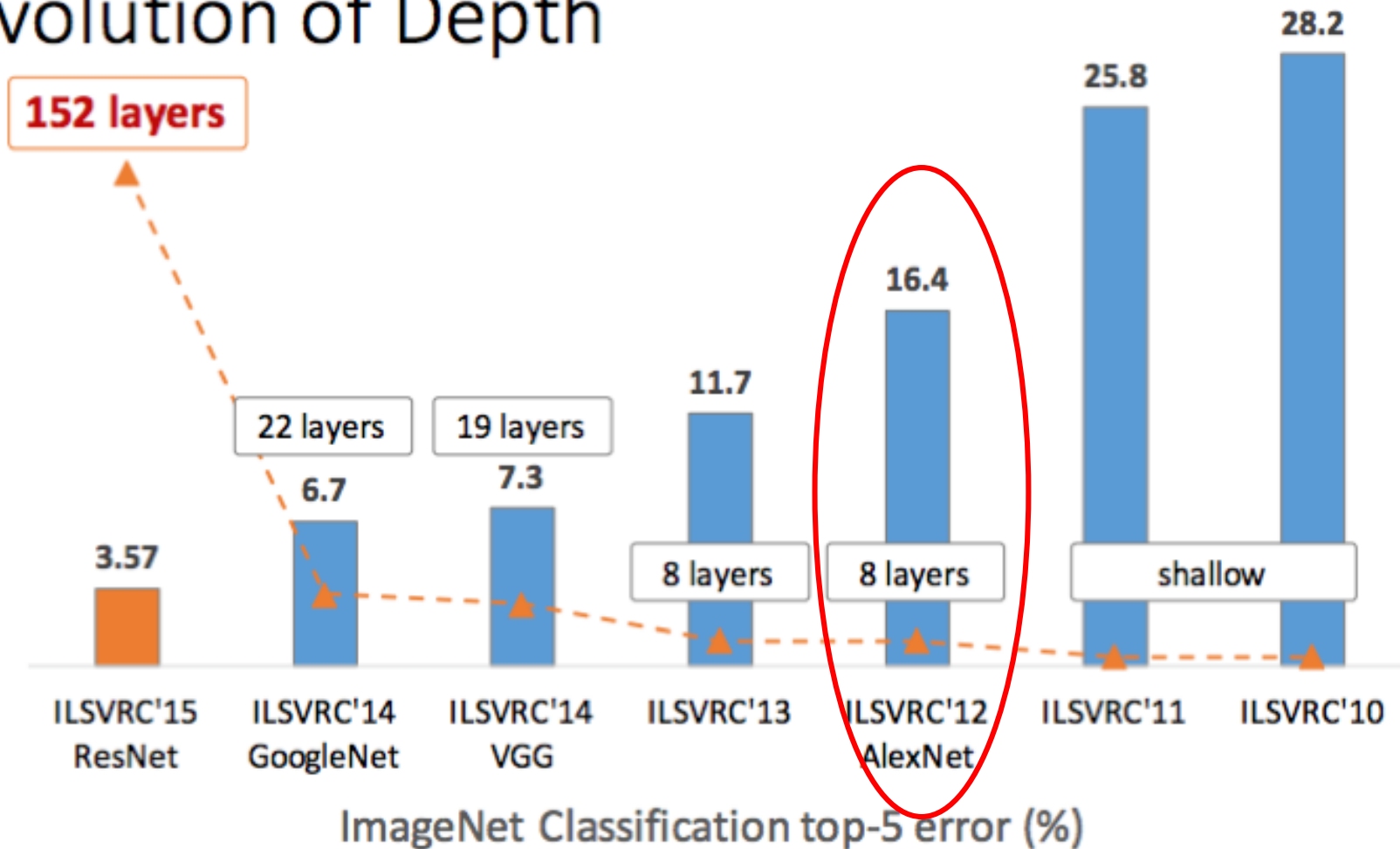


* - by Andrej Karpathy (no big claims)

plot credit: Kaiming He

ImageNet challenge

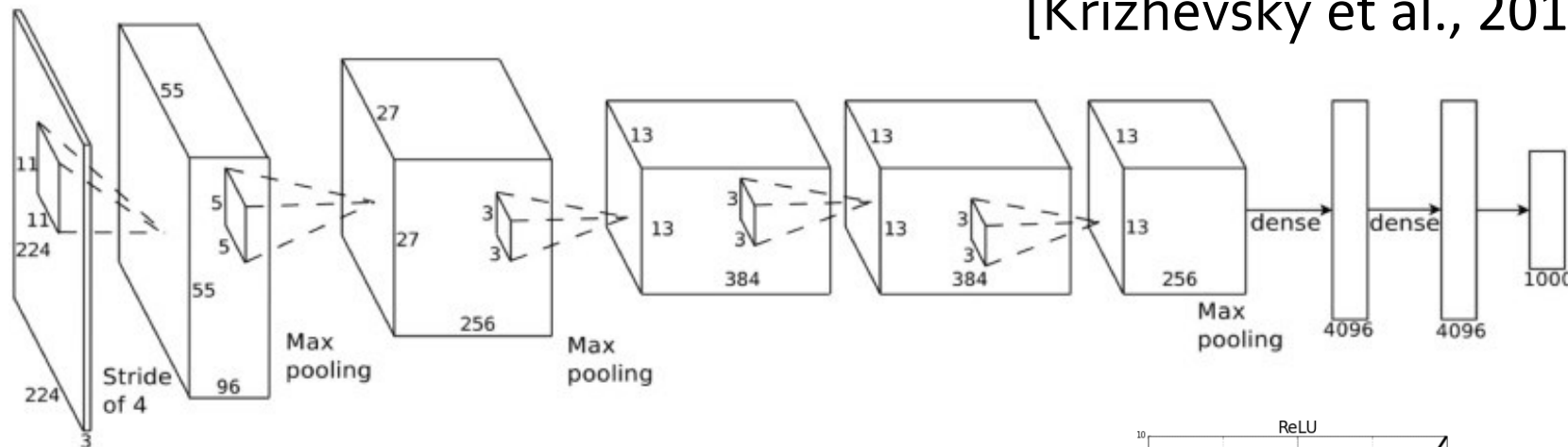
Revolution of Depth



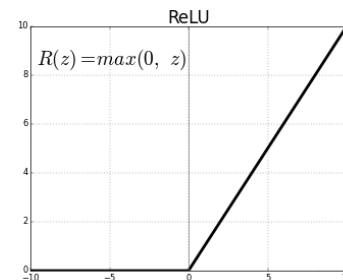
plot credit: Kaiming He

Свёрточные сети: AlexNet

[Krizhevsky et al., 2012]

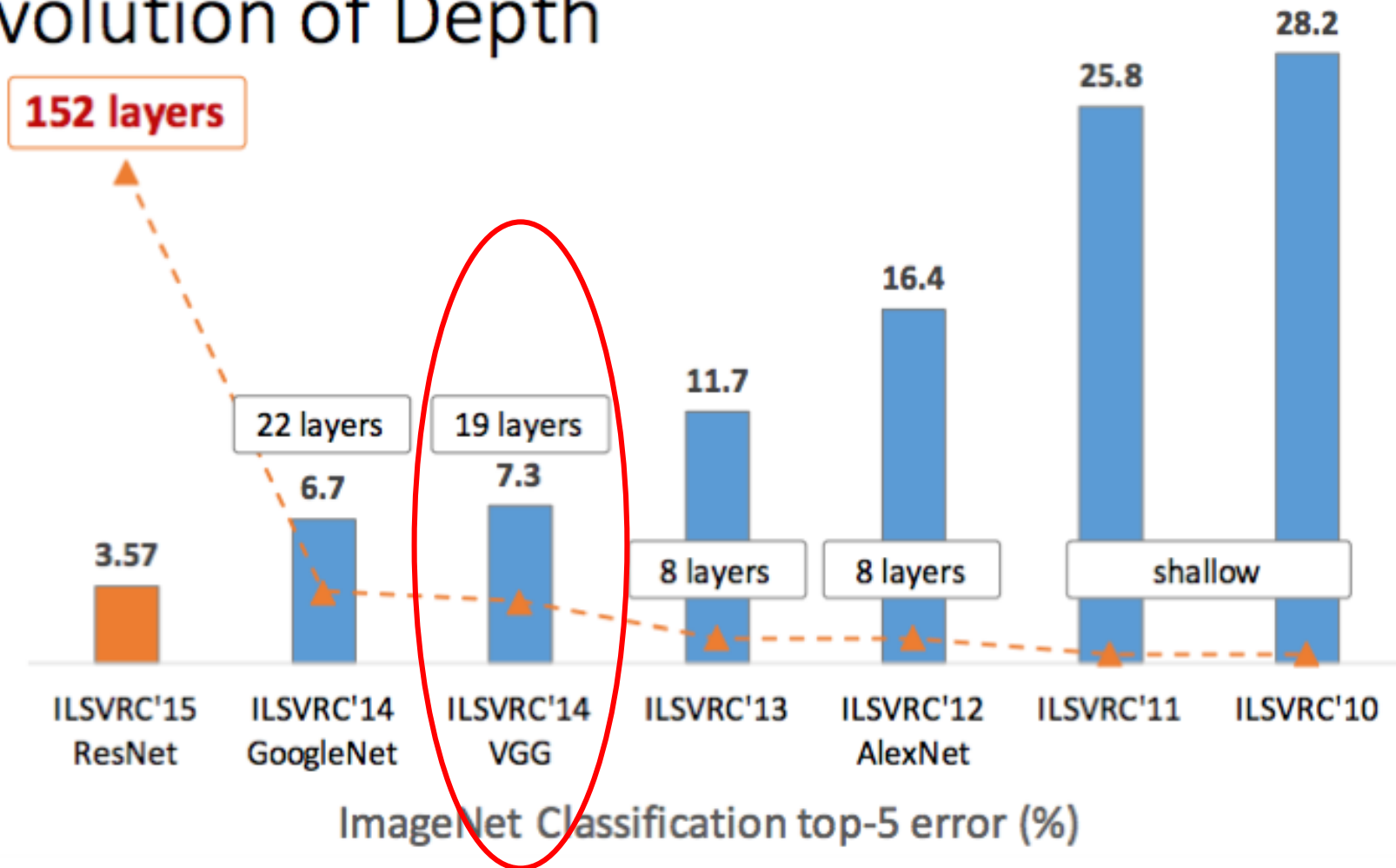


- Max-pooling
- Нелинейность: ReLu
- Полно-связные слои
- Больше данных и параметров (60M)
- + Data augmentation (flips and random samples)
- + Dropout regularization
- + GPUs (50x speed up)
- + 1 week of training on 2 GPUs



ImageNet challenge

Revolution of Depth

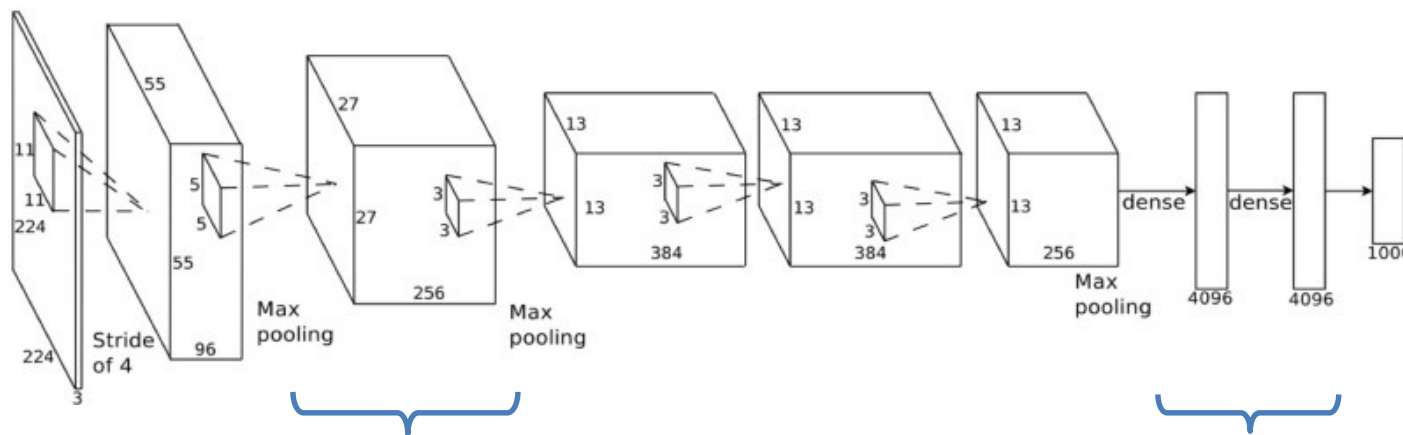


plot credit: Kaiming He

Архитектуры: VGG (2014)

[Simonyan & Zisserman, 2014]

- Каскад свёрток 3x3 вместо больших свёрток (меньше параметров и быстрее, чем 7x7)
- 140M параметров (у AlexNet 60M)
- Более сбалансированные вычисления



Большинство
вычислений

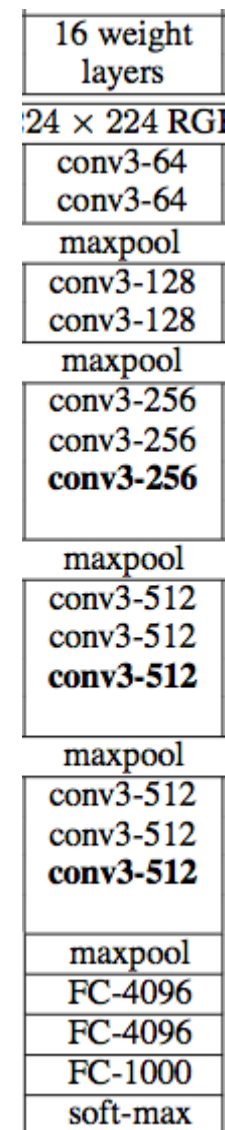
Большинство
параметров

16 weight layers
24 × 224 RGI
conv3-64 conv3-64
maxpool
conv3-128 conv3-128
maxpool
conv3-256 conv3-256 conv3-256
maxpool
conv3-512 conv3-512 conv3-512
maxpool
conv3-512 conv3-512 conv3-512
maxpool
FC-4096
FC-4096
FC-1000
soft-max

Архитектуры: VGG (2014)

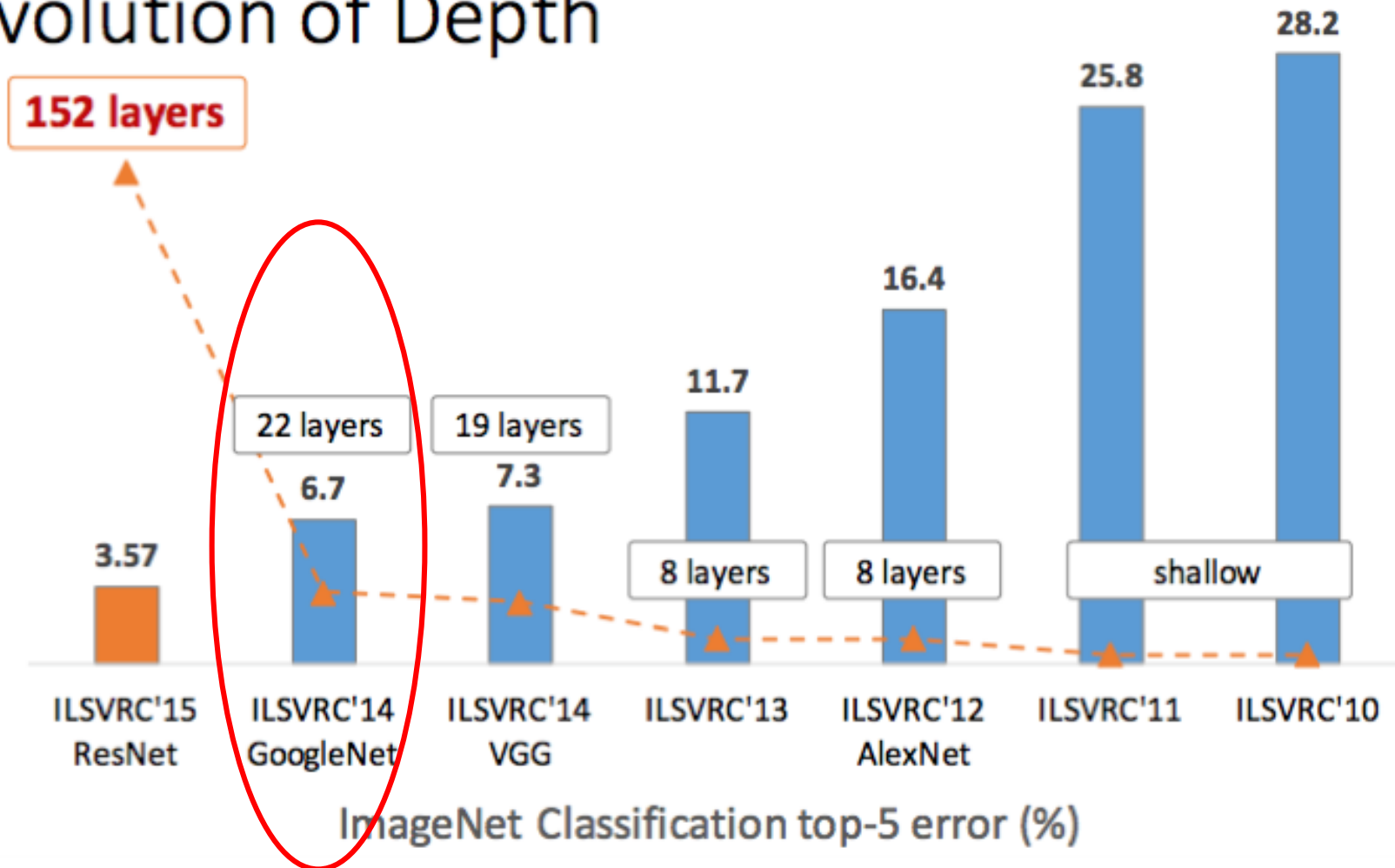
[Simonyan & Zisserman, 2014]

- Каскад свёрток 3x3 вместо больших свёрток (меньше параметров и быстрее, чем 7x7)
- 140M параметров (у AlexNet 60M)
- Более сбалансированные вычисления
- Не обучается целиком (затухает градиент)
- Несколько стадий обучения разной глубины
- Обучение 4 Titan Black GPUs 2-3 недели



ImageNet challenge

Revolution of Depth

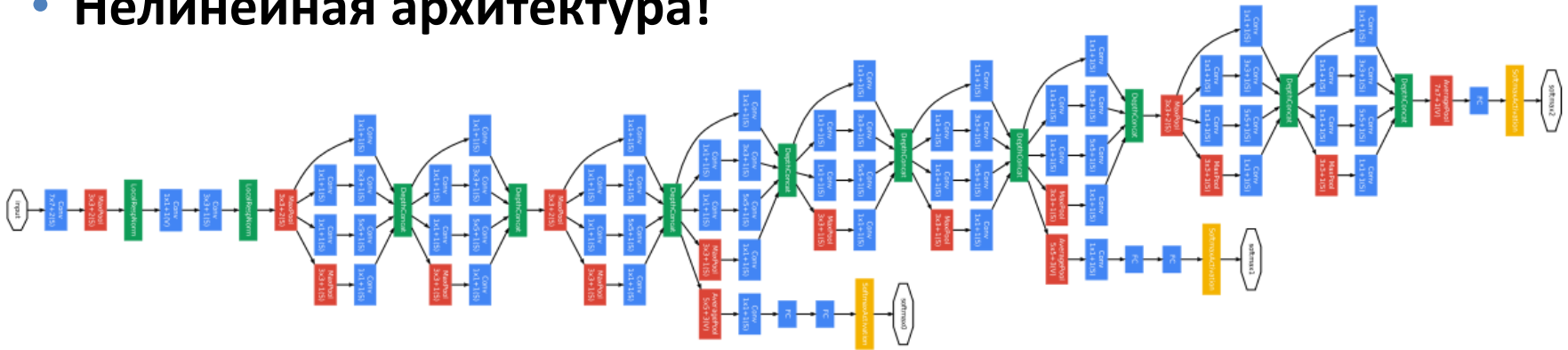


plot credit: Kaiming He

Архитектуры: GoogleNet (2014)

[Szegedy et al., 2015]

- **Нелинейная архитектура!**

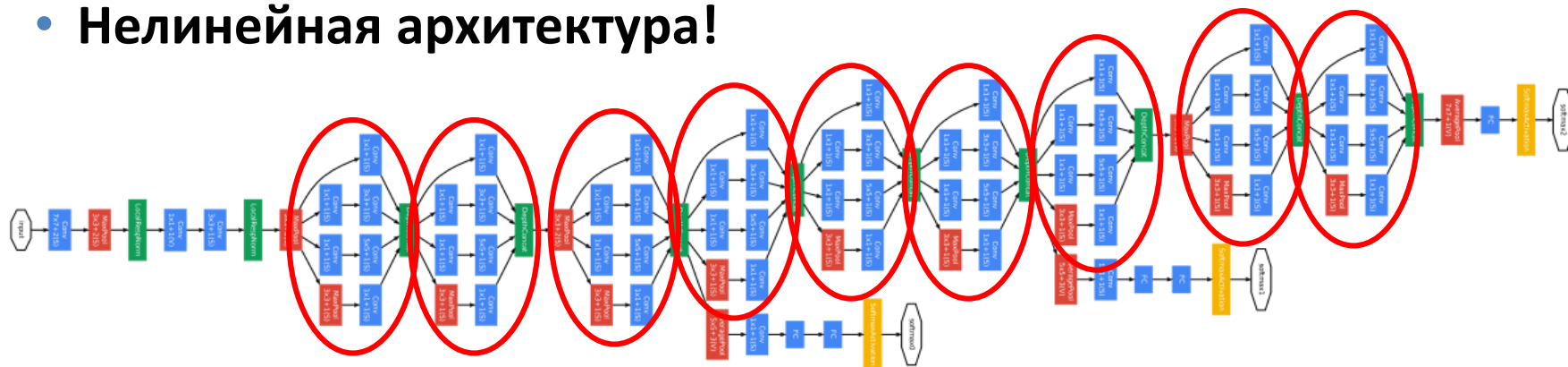


- Макс. глубина: 22 слоя с параметрами
- Нет полносвязных слоев
- 12x меньше параметров (чем в AlexNet)

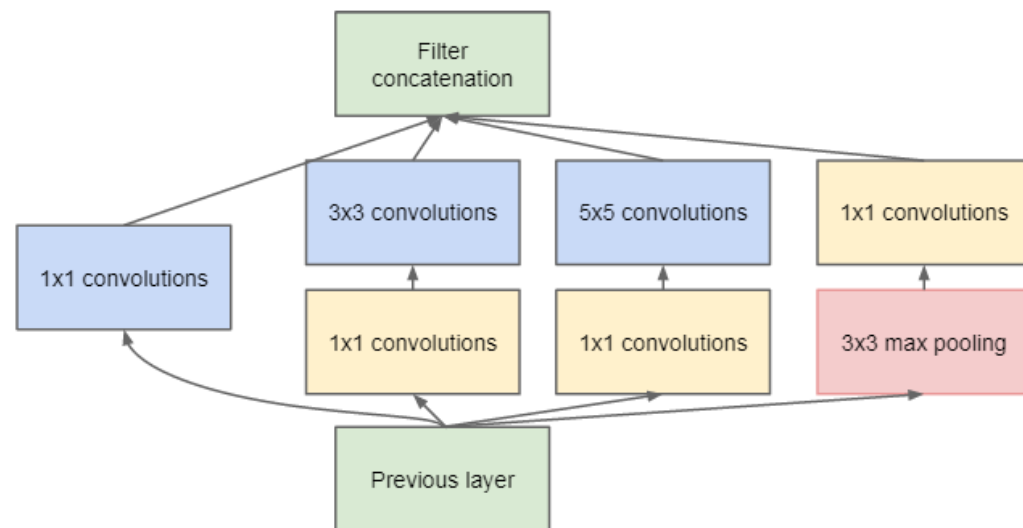
Архитектуры: GoogleNet (2014)

[Szegedy et al., 2015]

- **Нелинейная архитектура!**



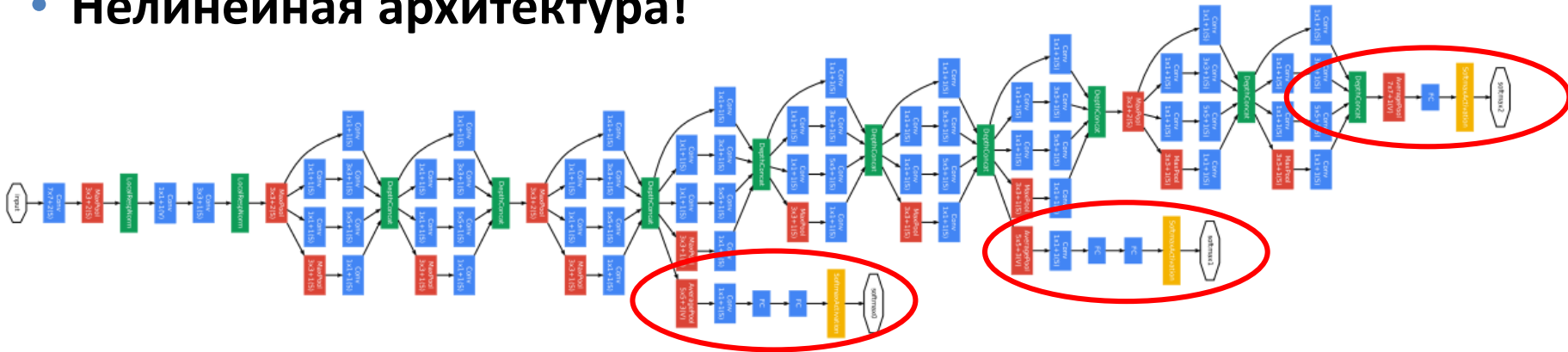
- Основной блок – Inception module (9 штук)
- 1x1 свёртки – уменьшение размерности



Архитектуры: GoogleNet (2014)

[Szegedy et al., 2015]

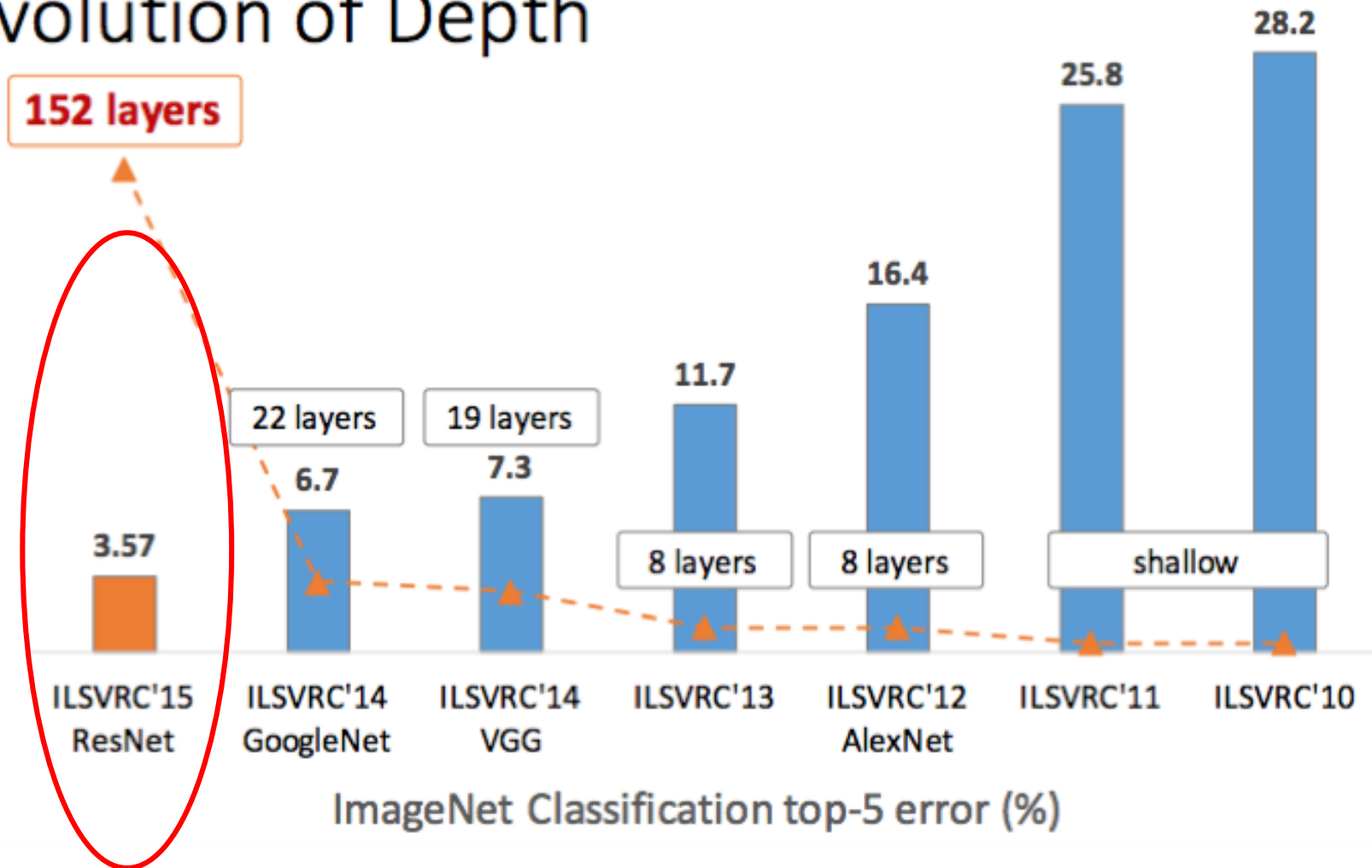
- **Нелинейная архитектура!**



- Очень глубокая сеть, целиком не обучается
- Один и тот же блок с функций потерь в нескольких местах
- «Проталкивает» градиент внутрь сети
- Обучалось на облаке CPU
- Добавился BatchNorm, Residual blocks и т.д. (Inception-v4)

ImageNet challenge

Revolution of Depth



plot credit: Kaiming He

Архитектуры: ResNet (2015)

[He et al., 2015]

- **Ultra deep! 100+ слоев**

AlexNet, 8 layers
(ILSVRC 2012)



ResNet, **152 layers**
(ILSVRC 2015)

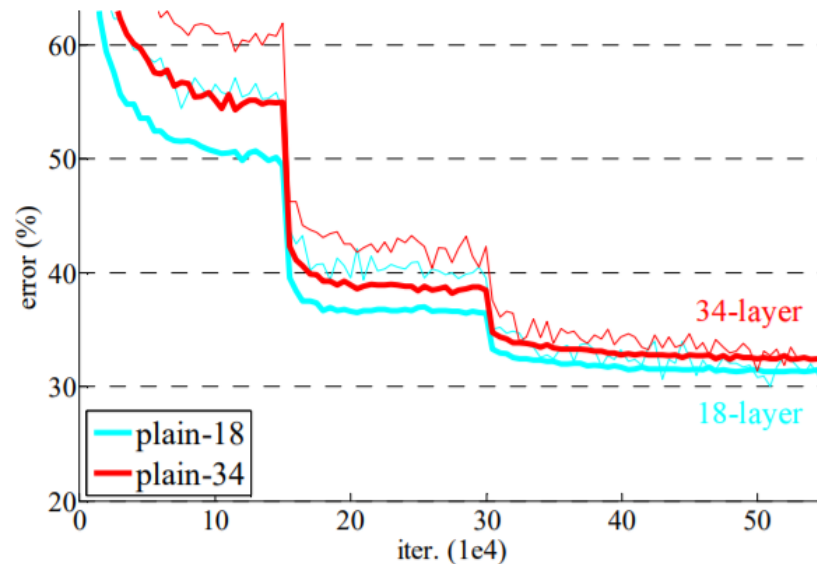
VGG, 19 layers
(ILSVRC 2014)



Архитектуры: ResNet (2015)

[He et al., 2015]

- **Ultra deep! 100+ слоев**
- Просто добавление слоев не работает



Архитектуры: ResNet (2015)

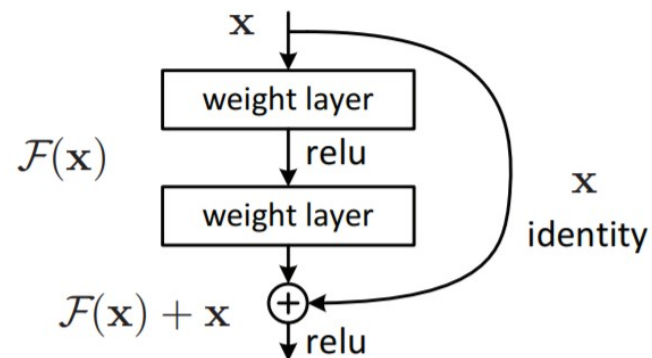
[He et al., 2015]

- **Ultra deep! 100+ слоев**
- Просто добавление слоев не работает
- Основная идея – остаточный блок

добавление тождественной связи

- Обычно: $y := f(x) \Rightarrow \frac{d\ell}{dx} := f'(x) \frac{d\ell}{dy}$
- Skip: $y := f(x) + x \Rightarrow \frac{d\ell}{dx} := f'(x) \frac{d\ell}{dy} + \frac{d\ell}{dy}$

- Связи “перепрыгивают” слои
- Пропускает градиент вглубь



Глубина не определена!

- В ResNet есть как короткие так и длинные пути
- DenseNet [Huang et al., 2016]

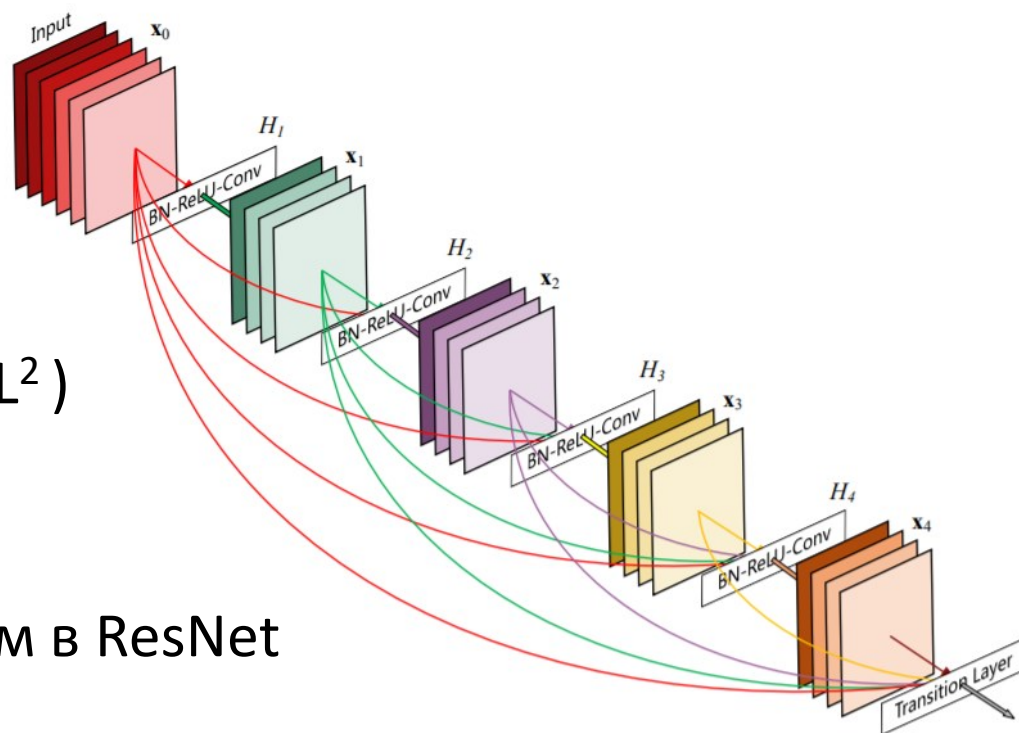
- Все слои связаны со всеми

- Число параметров – $O(L^2)$

L – число слоев

- Параметров меньше чем в ResNet

Очень узкие слои!



Рекуррентные сети (RNN, LSTM, GRU)

RNN – модель для последовательностей

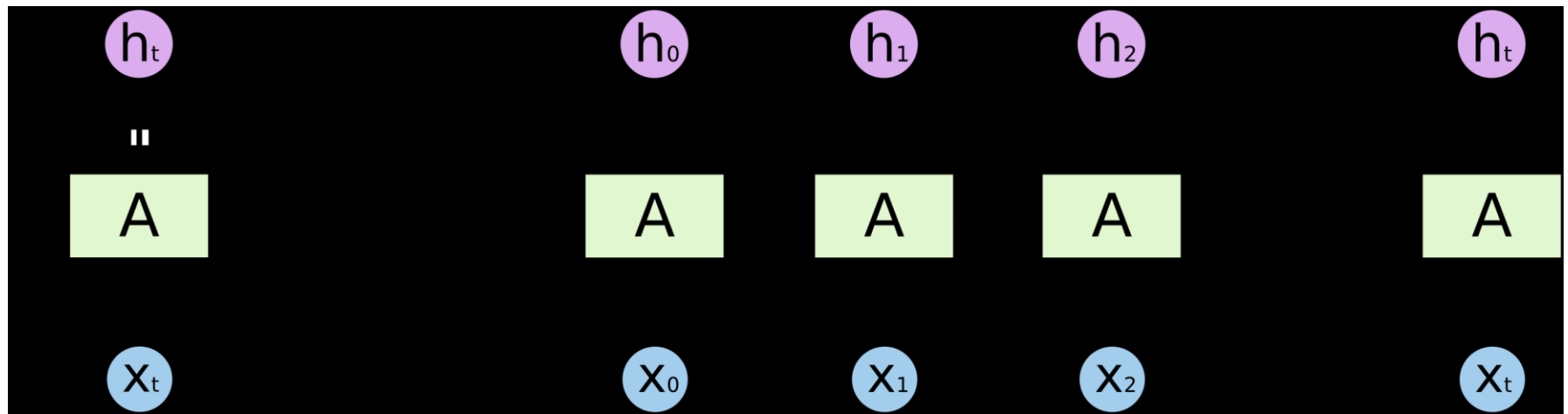
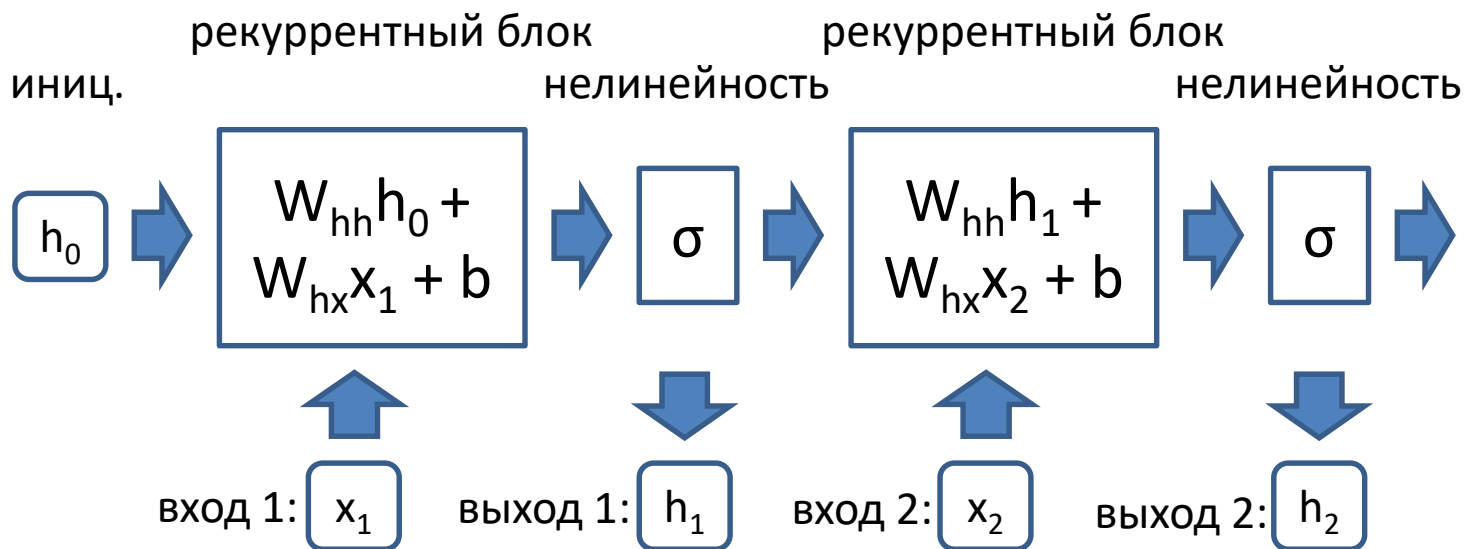


image credit: [Christopher Olah](#)

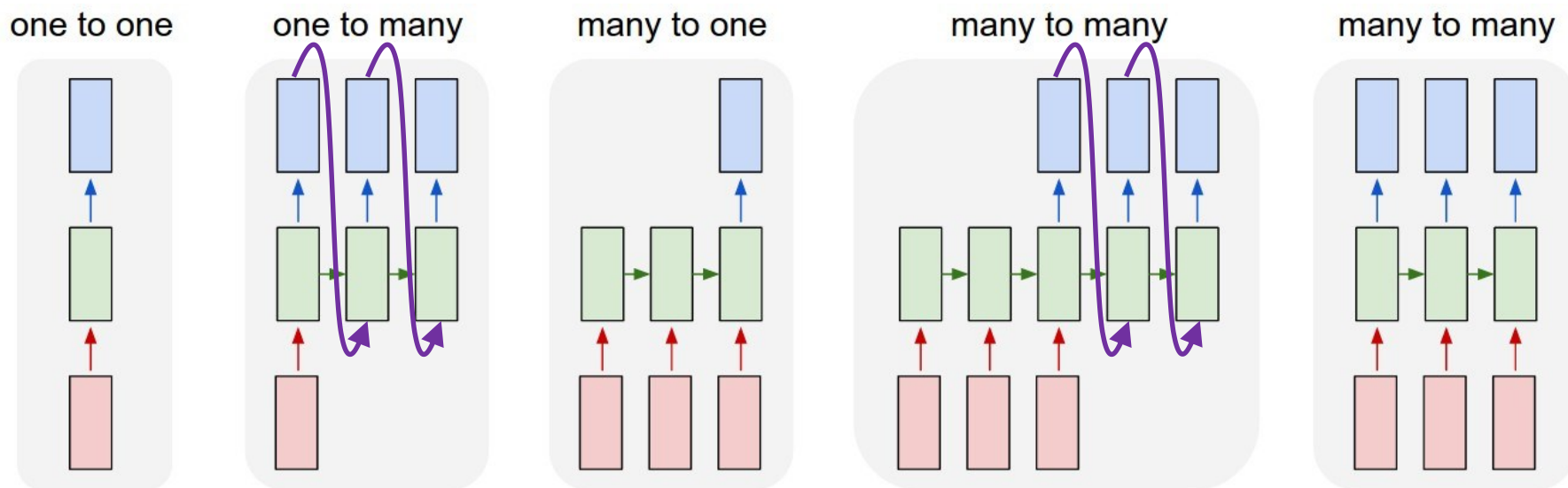
Базовый блок RNN



- Основной блок – линейный слой + нелинейность (tanh)
- Количество параметров не зависит от длины

Модели последовательностей

- RNN можно по разному собирать из блоков
- Варианты с авторегрессией (→)
- Глубокие RNN, двунаправленные RNN



Входы, память, выходы

image credit: [Andrej Karpathy](#)

Проблемы RNN: взрыв/затухание градиента

- Источник проблемы

$$\begin{aligned}\frac{d\ell(h_3)}{dh_0} &= J_{h_2 h_0}^T W_{hh}^T \sigma'(h_3) \frac{d\ell}{dh_3} \\ &= J_{h_1 h_0}^T W_{hh}^T \sigma'(h_2) W_{hh}^T \sigma'(h_3) \frac{d\ell}{dh_3} \\ &= W_{hh}^T \sigma'(h_1) W_{hh}^T \sigma'(h_2) W_{hh}^T \sigma'(h_3) \frac{d\ell}{dh_3}\end{aligned}$$

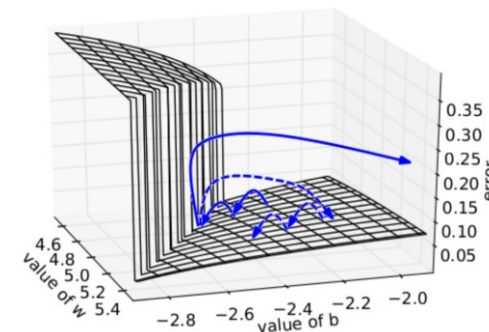


image credit: [Pascanu et al., 2012]

- Взрывы => нестабильное обучение
- Затухание => длинные зависимости не обучаются

- Взрывы – gradient clipping [Pascanu et al., 2012]

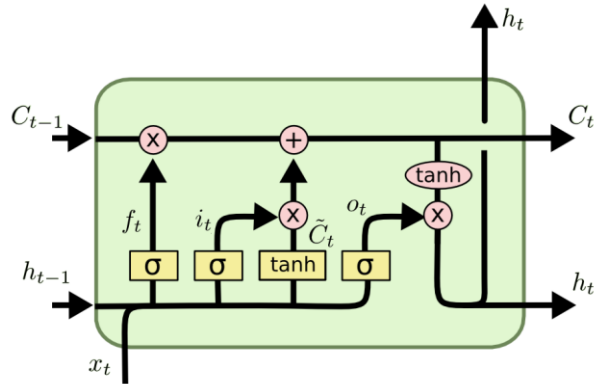
Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

```
 $\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$   
if  $\|\hat{\mathbf{g}}\| \geq threshold$  then  
     $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$   
end if
```

- Затухание градиента – специальные ячейки (LSTM, GRU)

Варианты базового блока

Long Short Term Memory (LSTM) [Hochreiter&Schmidhuber, 1997]



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

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

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

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

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

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

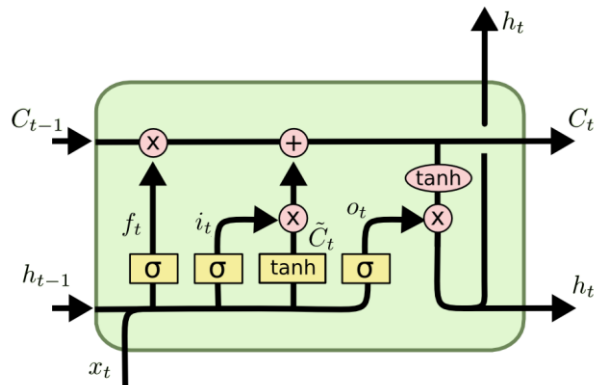
Основные компоненты:

- “Ворота” (gates) – умножение на $\sigma(\text{data}) \in [0, 1]$ (открыты – закрыты)
- Память C_t при открытом f_t и закрытом i_t проходит насквозь
- Линейные слои и нелинейности

image credit: [Christopher Olah](#)

Варианты базового блока

Long Short Term Memory (LSTM) [Hochreiter&Schmidhuber, 1997]



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

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

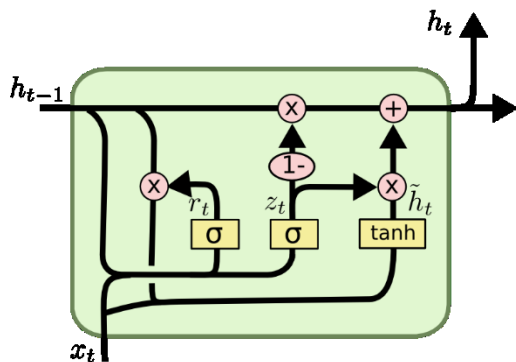
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

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

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

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

Gated Recurrent Unit (GRU) [Cho et al., 2014]



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- В cuDNN другой GRU!
- Гиперпараметры важны!
- Есть и другие варианты ячеек

image credit: [Christopher Olah](#)

Заключение

- Свёрточные сети – учитывают пространственную связность (2D, 1D, 3D, 4D)
 - Свёртка – нелинейность – пулинг
 - Для изображений – ResNet, DenseNet
 - Сложно – обучать с нуля; дообучать сильно проще
- Рекуррентные сети – для последовательностей
 - Многократное применение одного слоя
 - Затухающие и взрывающиеся градиенты
 - Clipping градиентов
 - LSTM, GRU, etc.