

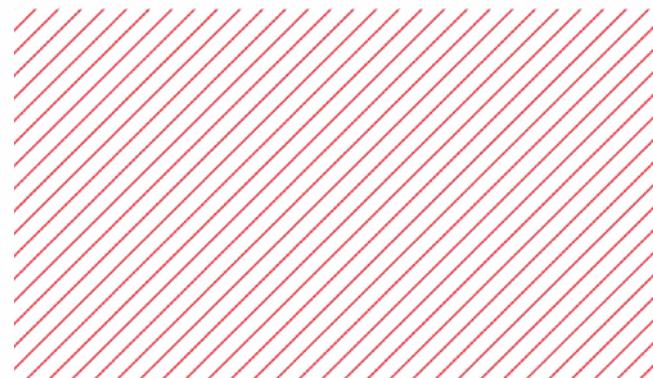
академия
больших
данных



Современные архитектуры CNN

Даниил Лысухин

Команда компьютерного зрения mail.ru



Recap

Recap: свертка

Входной сигнал X_{ij}

X00	X01	X02	X03
X10	X11	X12	X13
X20	X21	X22	X23
X30	X31	X32	X33

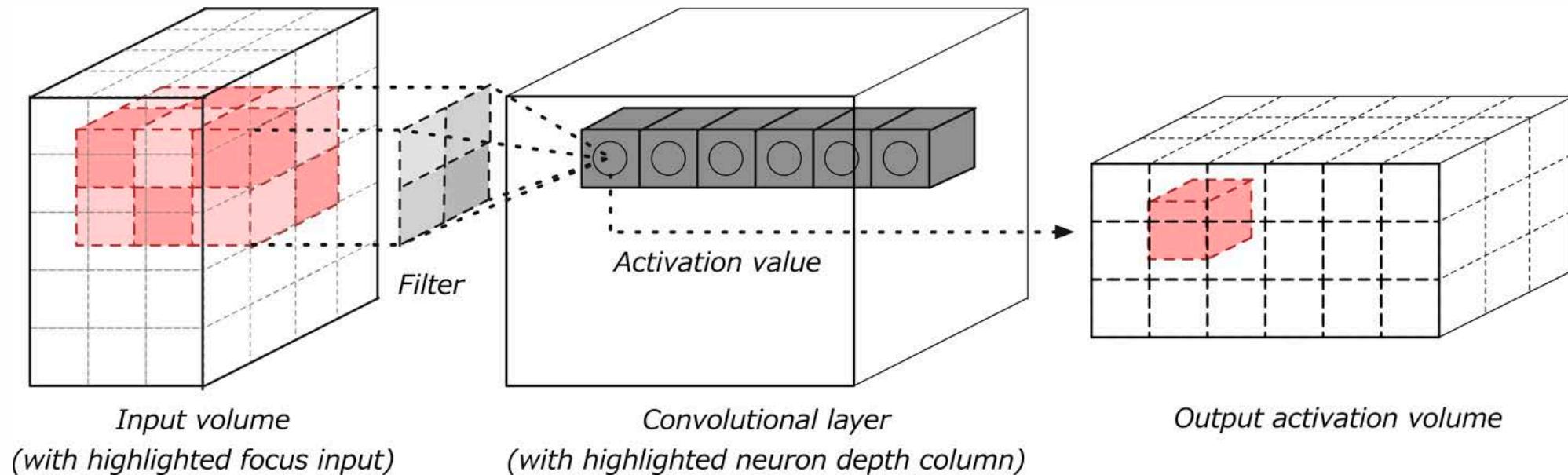
$$Y_{ij} = (X * W)_{ij} = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} X_{i+u, j+v} W_{uv}$$

Результат свертки Y_{ij}

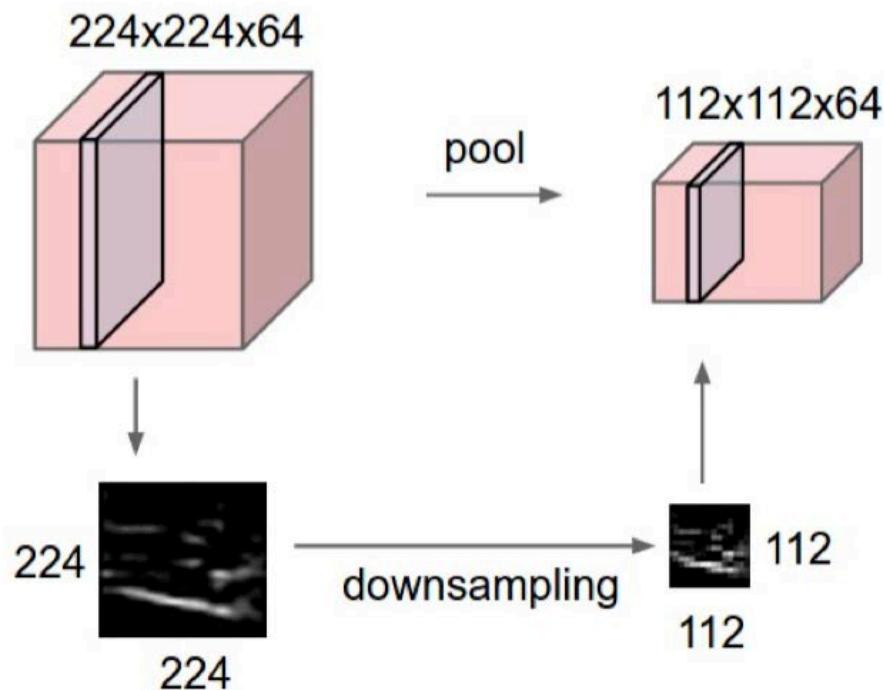
Y00	Y01
Y10	Y11

Ядро свертки W
Размер **3x3**
Веса W_{ij}

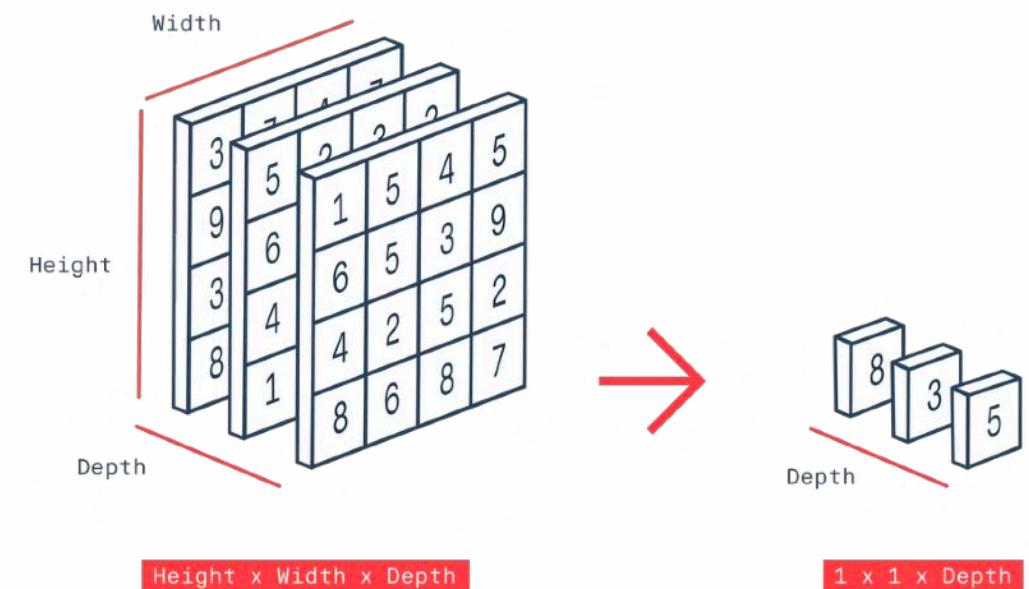
Recap: сверточный слой



Recap: pooling



“Обычный” pooling



Global pooling

LeNet-5 (1998)

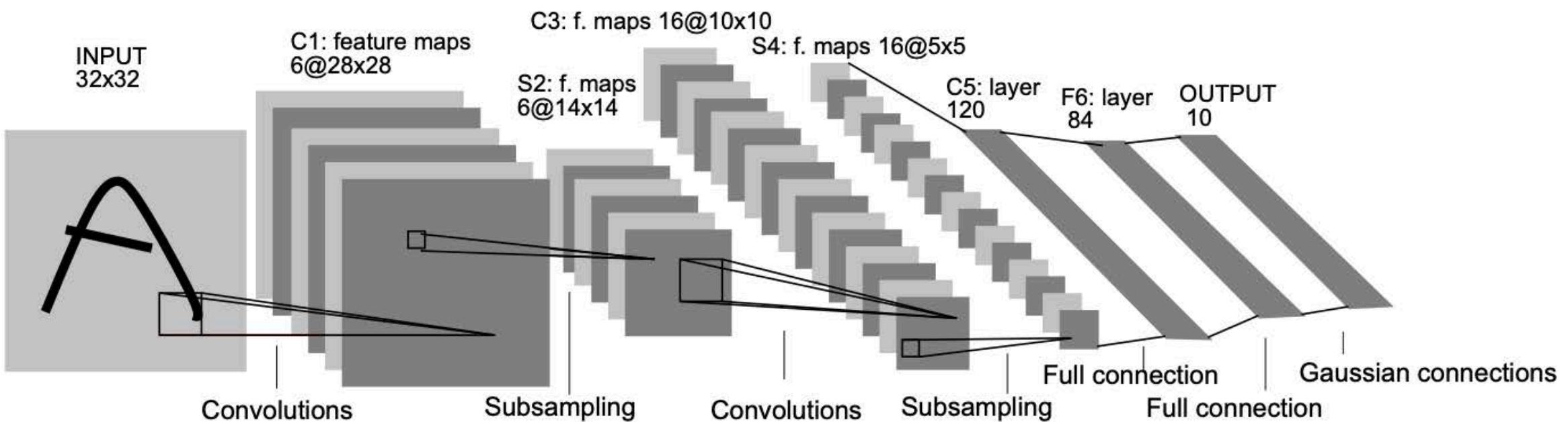


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



ImageNet



ImageNet

- База изображений на основе иерархии WordNet
 - > 20k категорий
 - > 15M примеров
 - Разметка = 1 категория + bounding box
- Есть проблемы
 - 1 категория для фото независимо от числа объектов
 - Разнообразие классов (зачем столько пород собак?)

ImageNet



grille



mushroom



cherry



Madagascar cat



mite



container ship



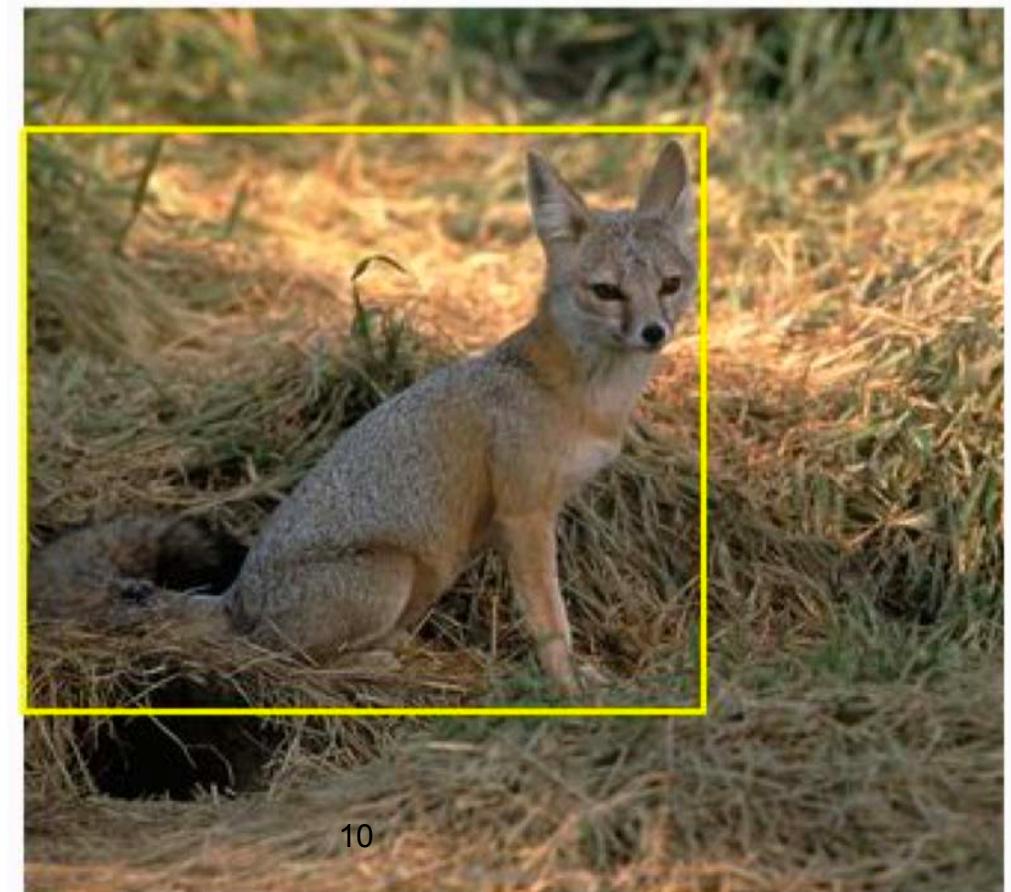
motor scooter

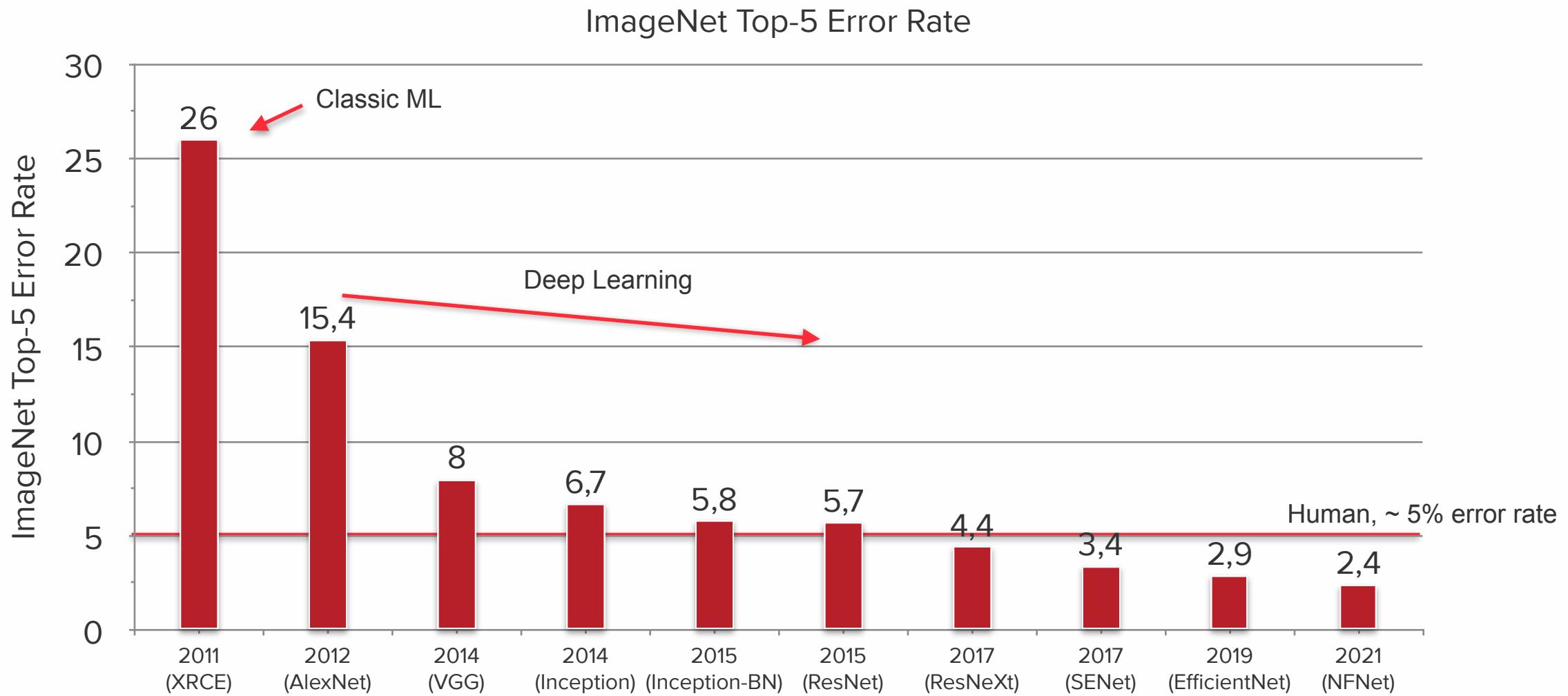


leopard

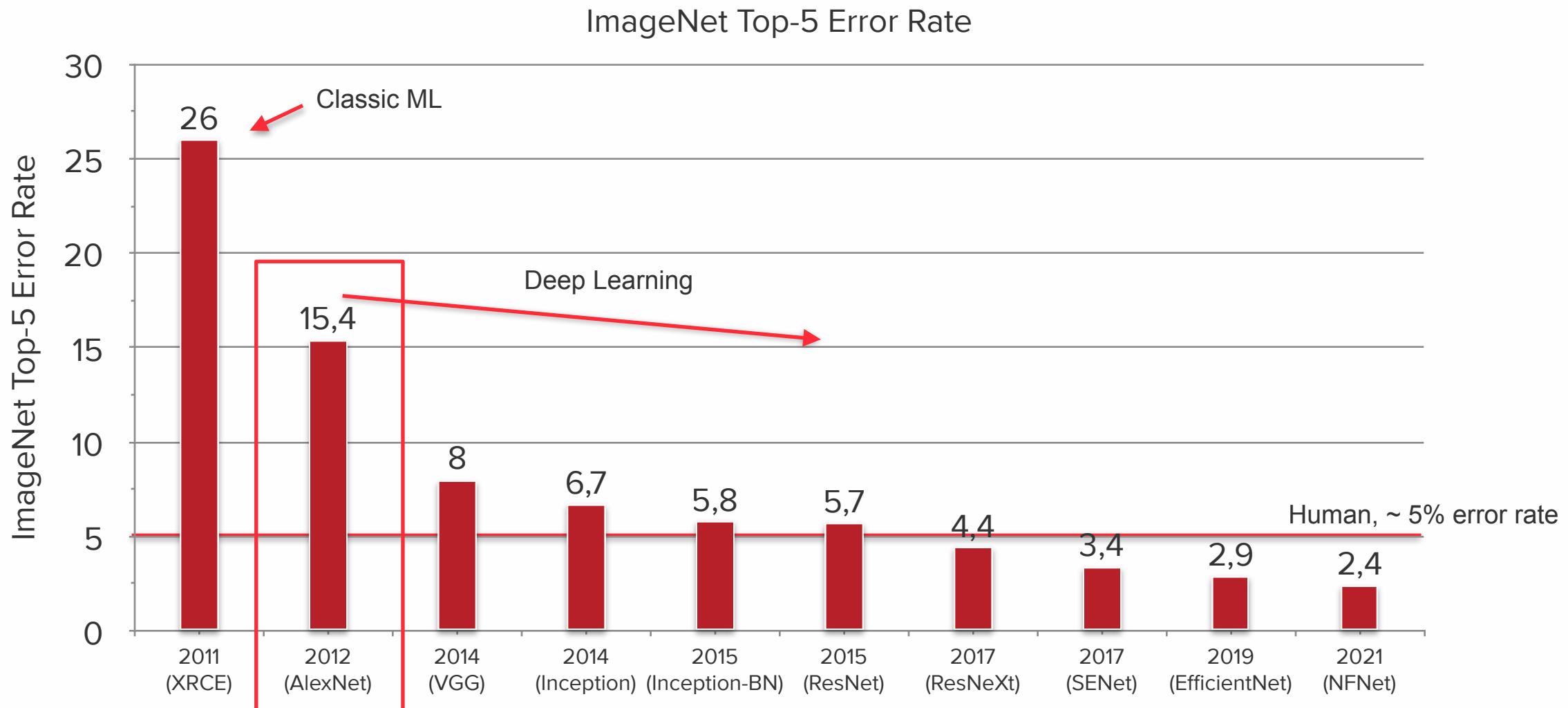
ILSVRC

- ImageNet Large-scale Visual Recognition Challenge
- Соревнование на данных ImageNet
- Подзадачи:
 - Классификация (1000 классов)
 - Локализация





Эволюция DL

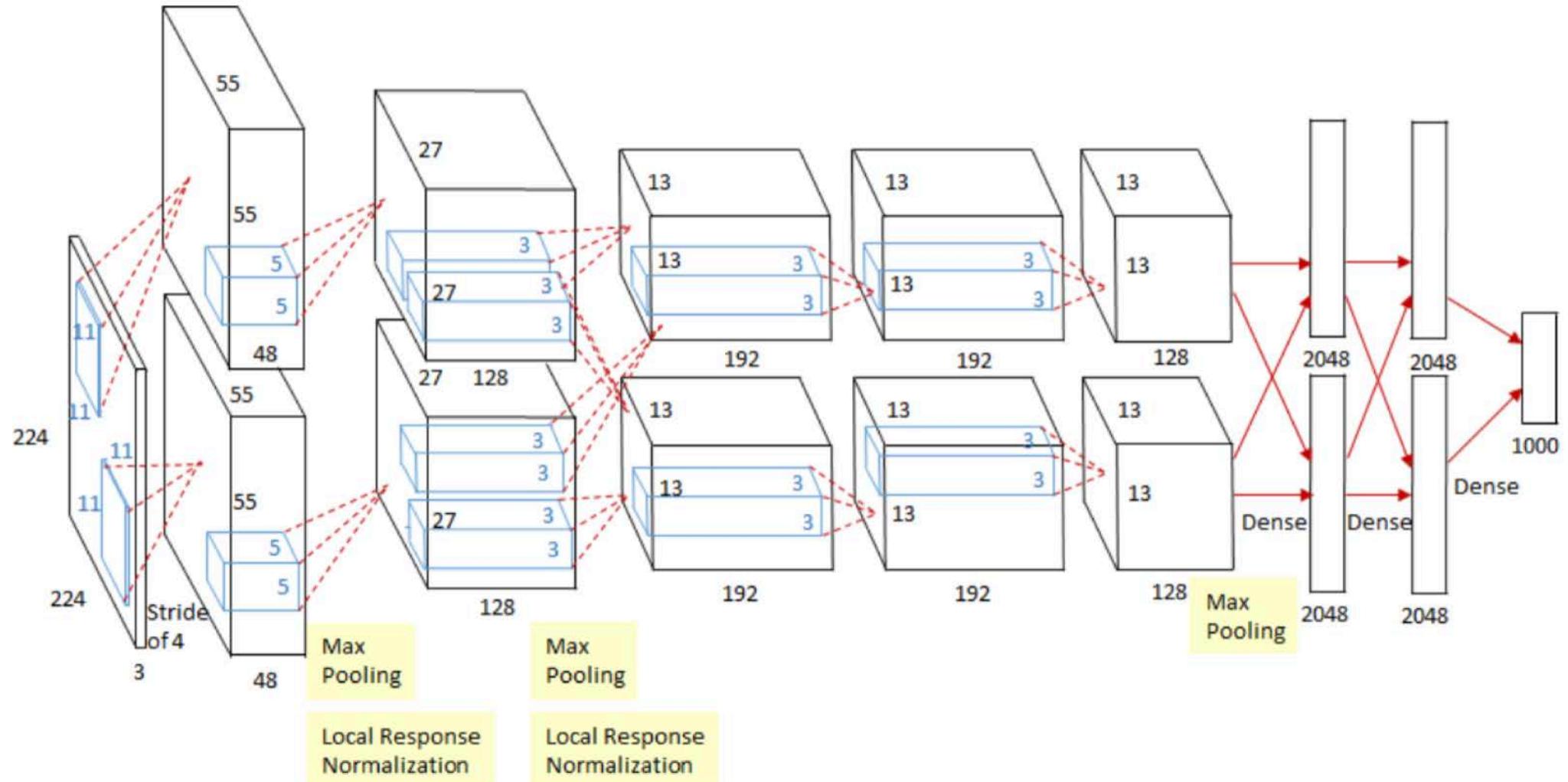




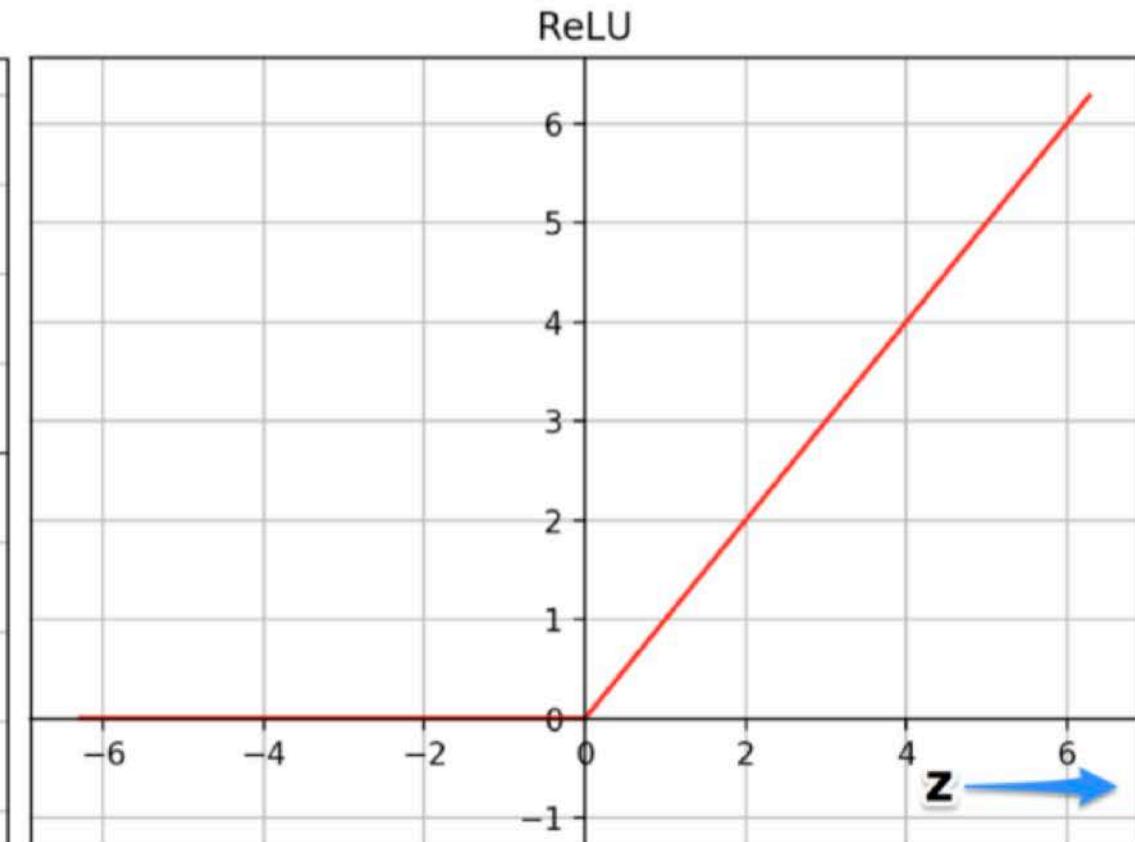
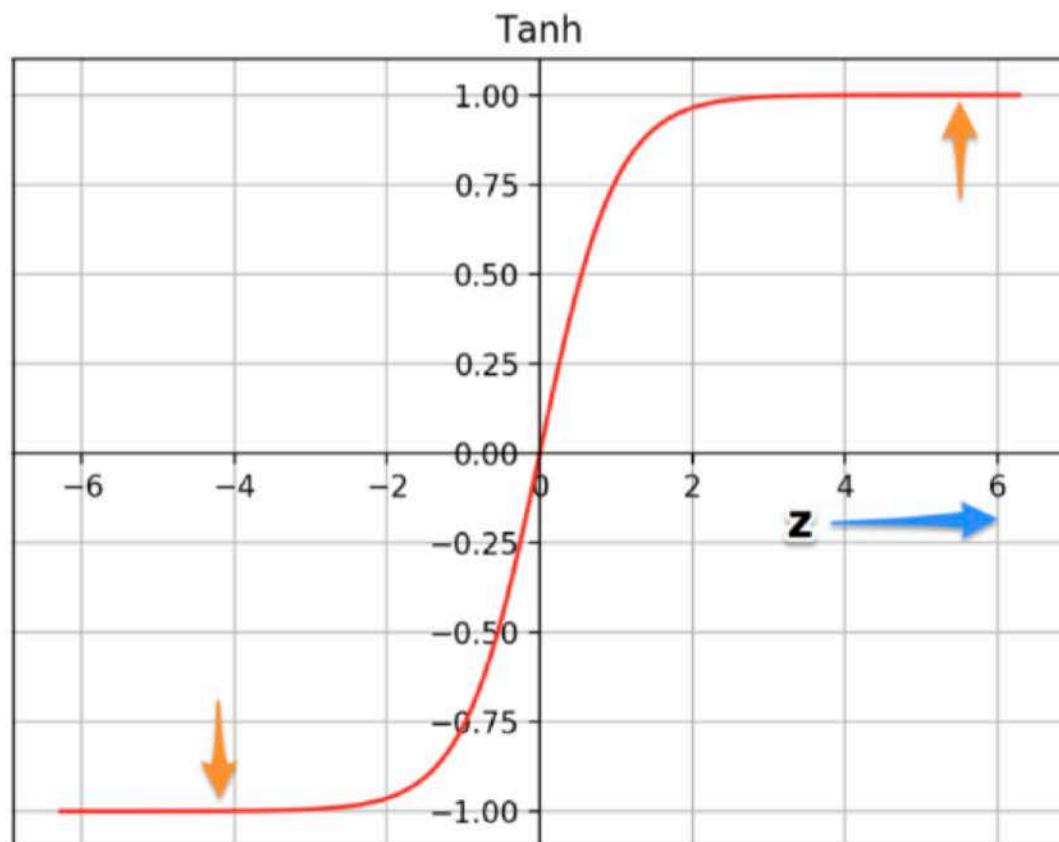
AlexNet (2012)

- [ImageNet Classification with Deep Convolutional Neural Networks](#)
- Первый победитель ILSVRC на основе нейросетей
- **60M** параметров
- 2 ветви на **2 GPU**
- Свертки **от 3x3 до 11x11**
- ReLU, Dropout (в FC)
- **Test-time Augmentation**

AlexNet (2012)



AlexNet (2012)



AlexNet (2012)

- Аугментации на обучении
 - Случайные кропы 227x227 (из 256x256)
 - Зеркальное отражение

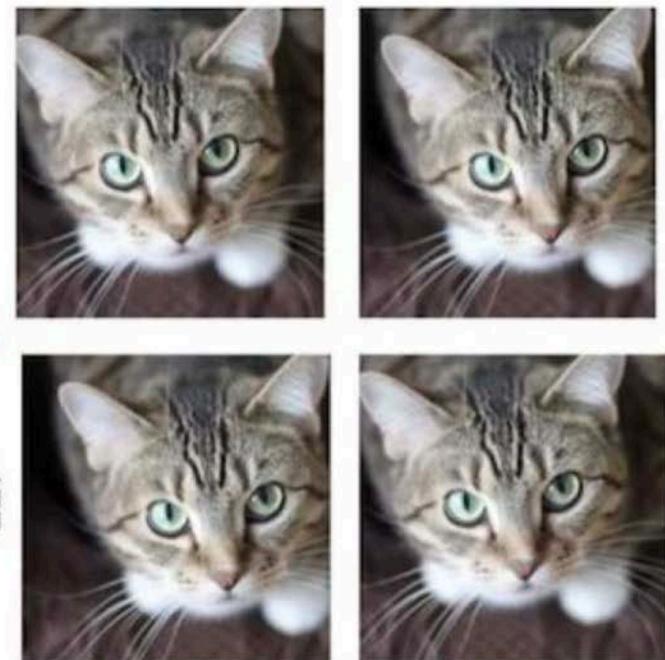


Mirror Image



256

Random Crops



227

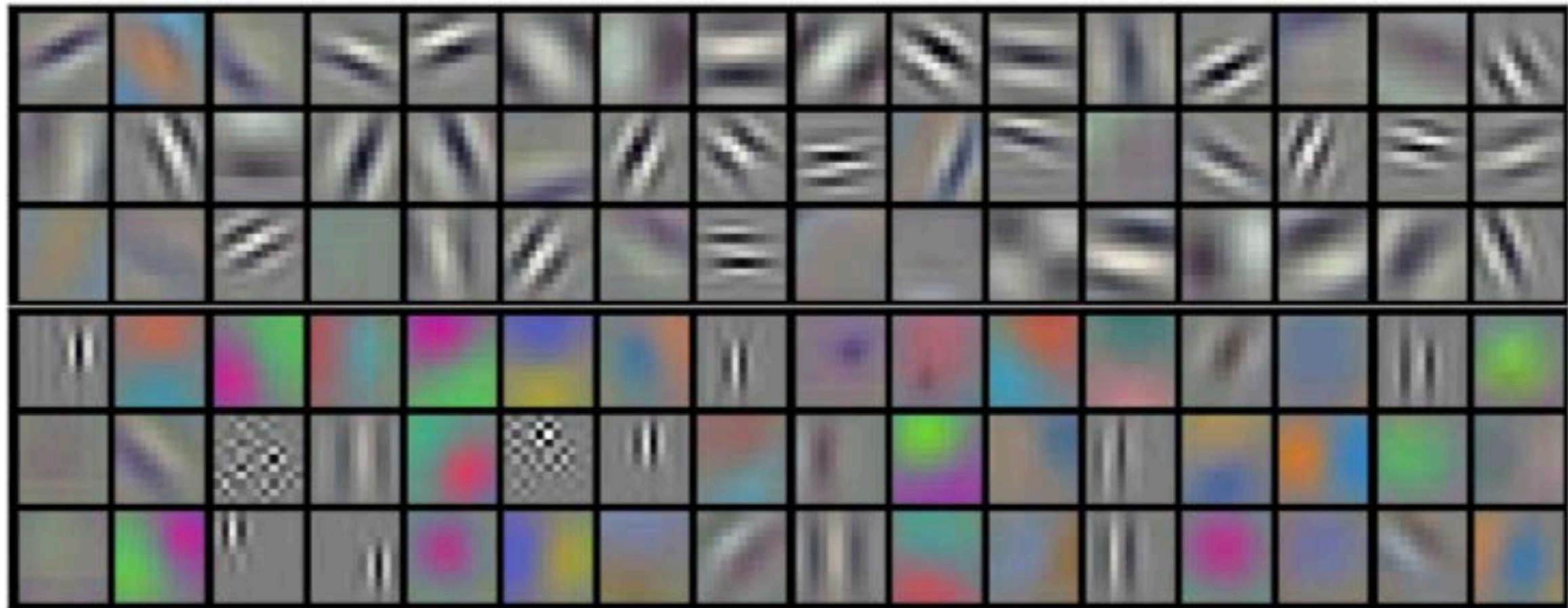
227



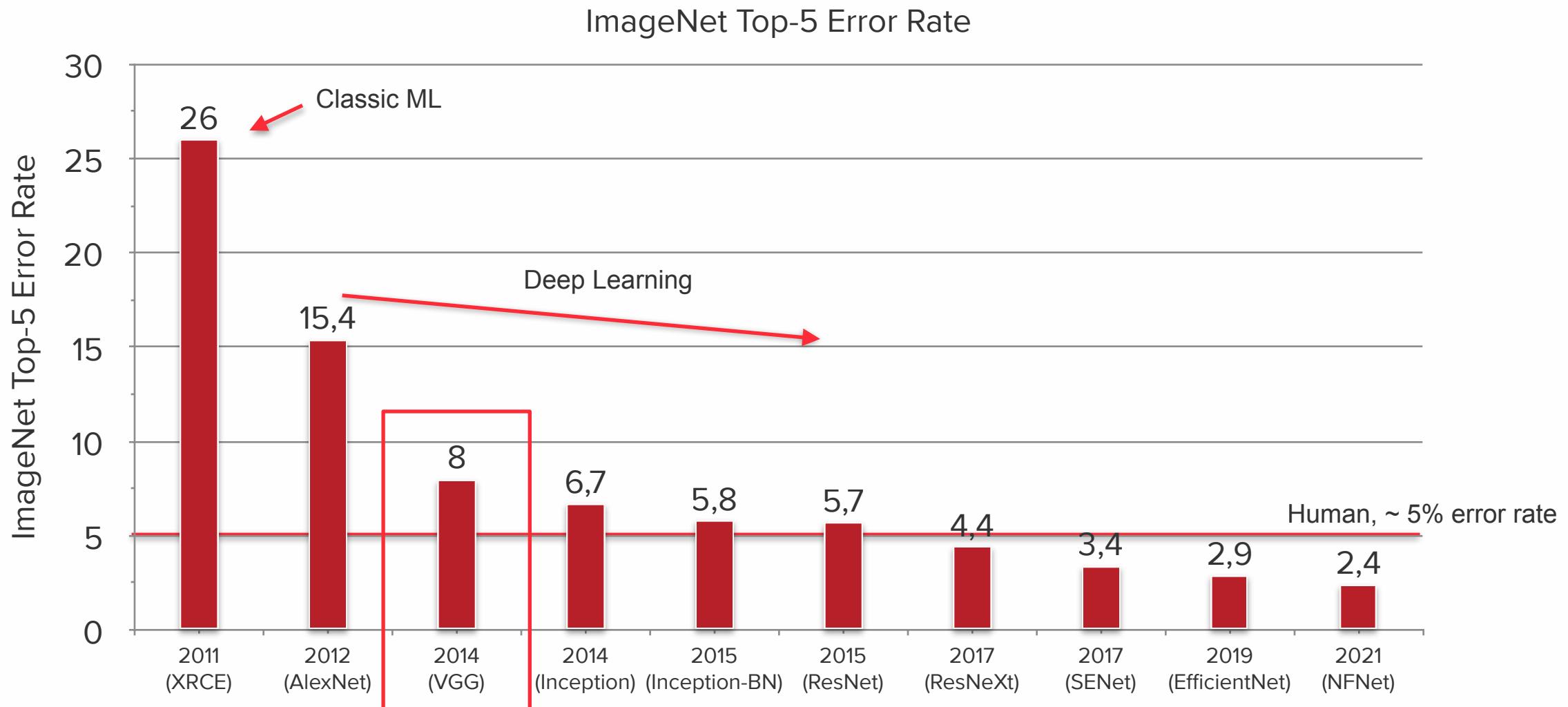
AlexNet (2012)

- Аугментации на инференсе (**Test-time Augmentation, TTA**)
 - Агрегирование предсказаний для 1 изображения по 10 его вариациям
 - 5 кропов (углы + центр)
 - $\times 2$ с учетом отраженных

AlexNet (2012)

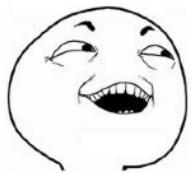


Learnings of First convolution layer on image of size 224X224X3





VGG (2014)



- Very Deep Convolutional Networks for Large-Scale Image Recognition
- Свертки **1x1** и **3x3**
- **138M** параметров у самой большой вариации (VGG-19)

VGG (2014)

- Семейство моделей
- Отличаются глубиной
 - VGG-11 / 13 / 16 / 19

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

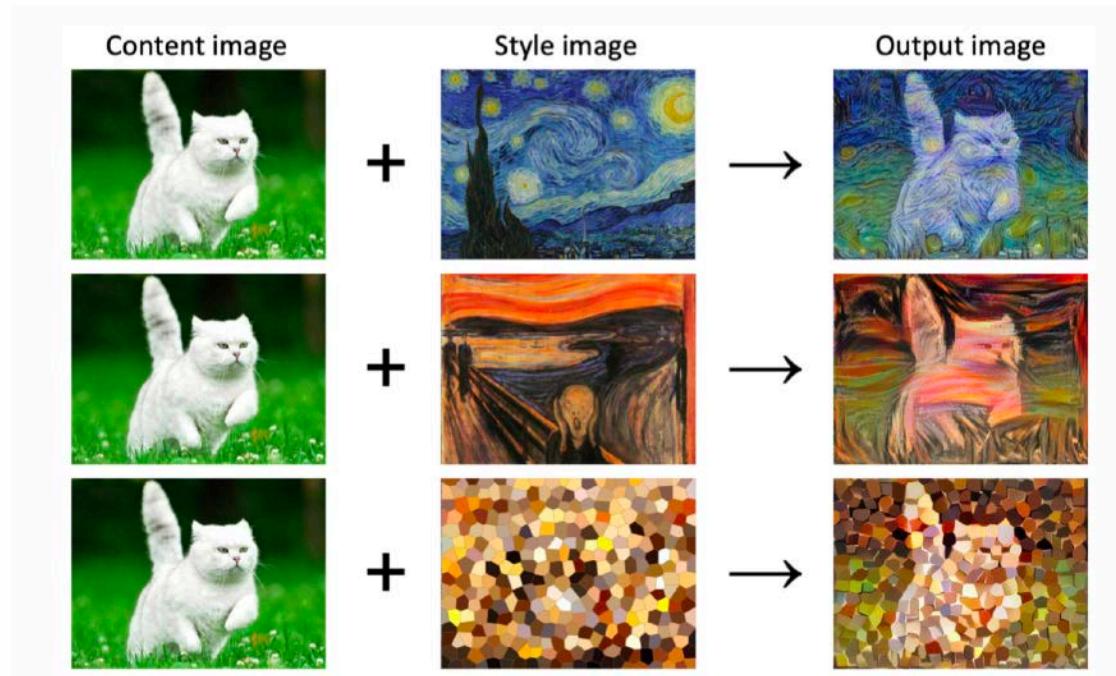
VGG (2014)

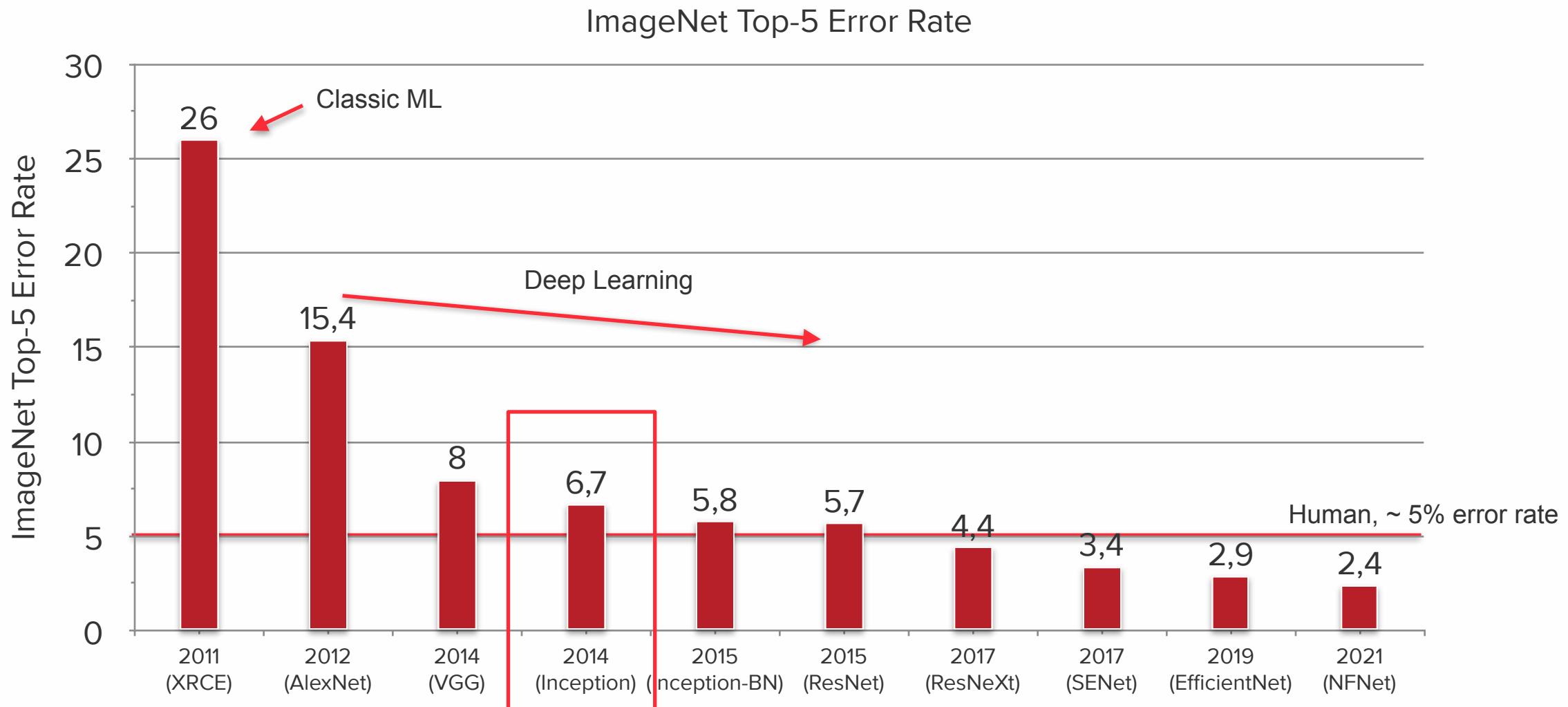
- # весов: $(3 \times 3 \times 512) \times 512 = 2\ 359\ 296$
- # весов: $(7 \times 7 \times 512) \times 4096 = 102\ 760\ 448$

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG (2014)

- Оказалось, что предобученная VGG извлекает “хорошие” признаки, пригодные для переиспользования
- Например, в задаче Style Transfer





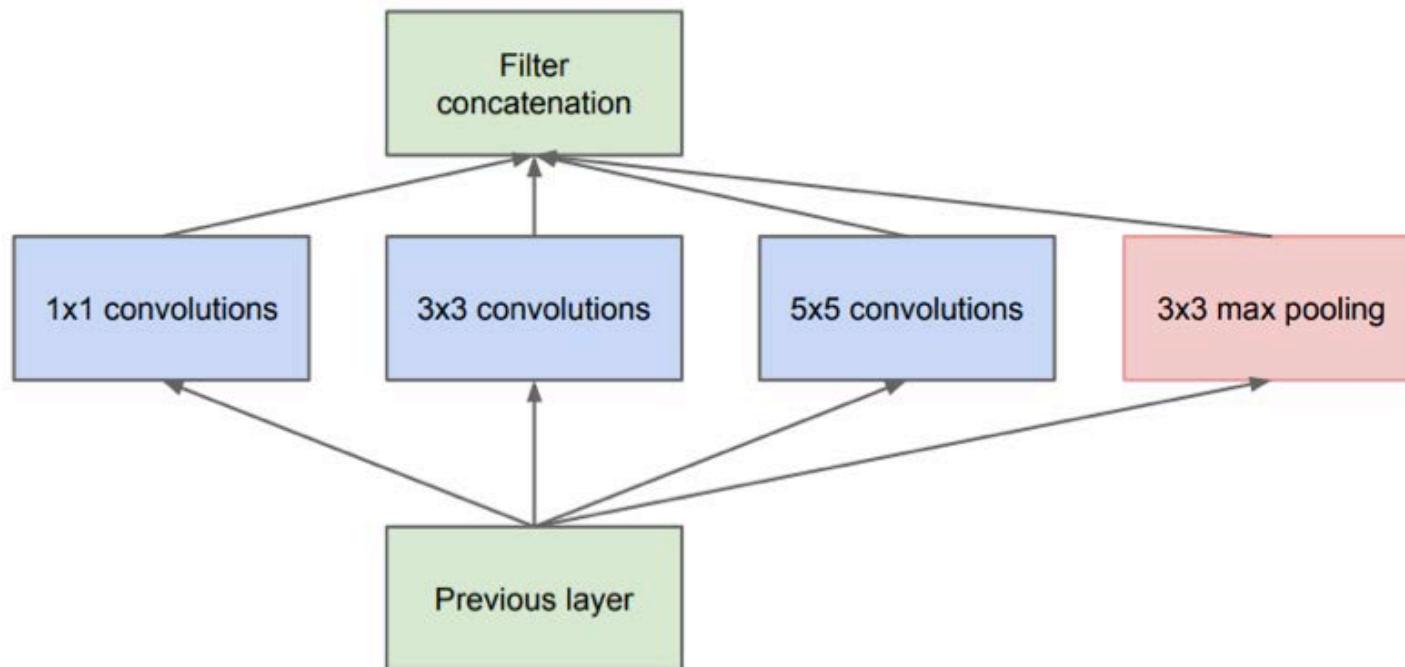


Inception / GoogLeNet (2014)

- Going Deeper with Convolutions
- Одновременное извлечение **признаков разного масштаба**
 - Новая структурная единица - **InceptionBlock**
- **Дополнительные выходы** с функцией потерь
- **Global Average Pooling** перед классификацией
- Мало параметров (**5М**)

Inception / GoogLeNet (2014)

- InceptionBlock (“ванильный” вариант)

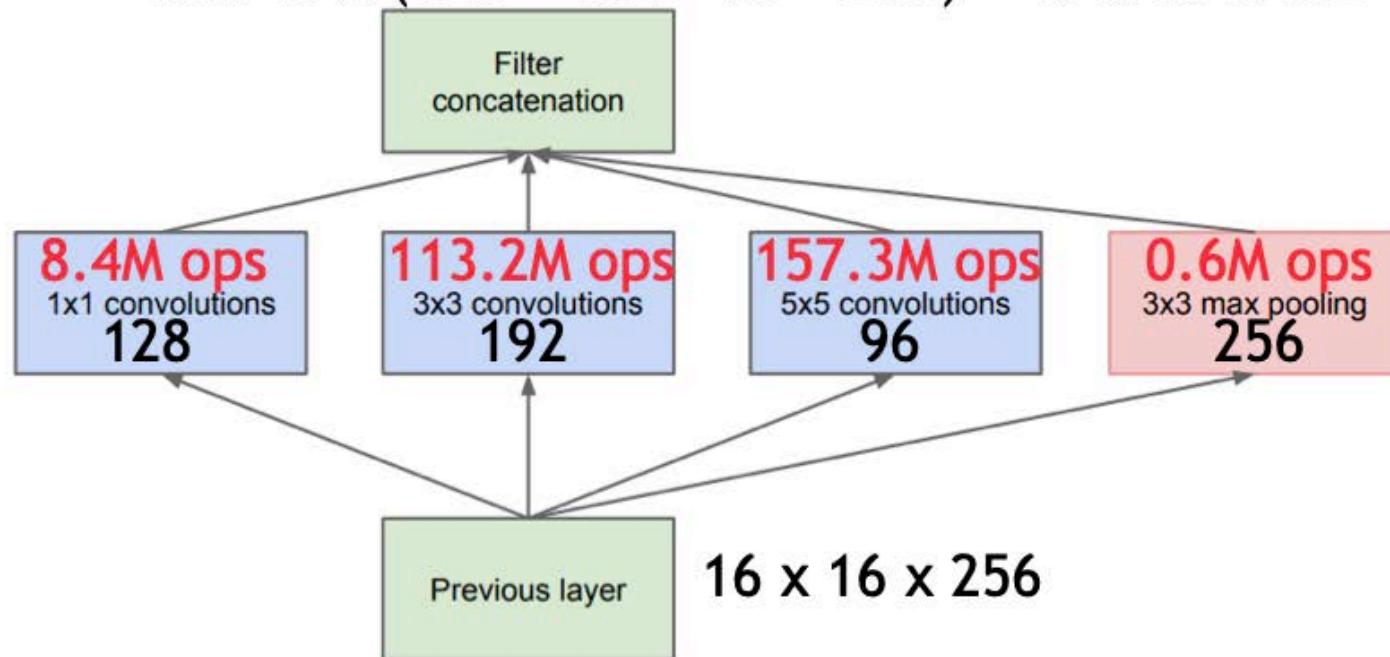


(a) Inception module, naïve version

Inception / GoogLeNet (2014)

- InceptionBlock (“ванильный” вариант)

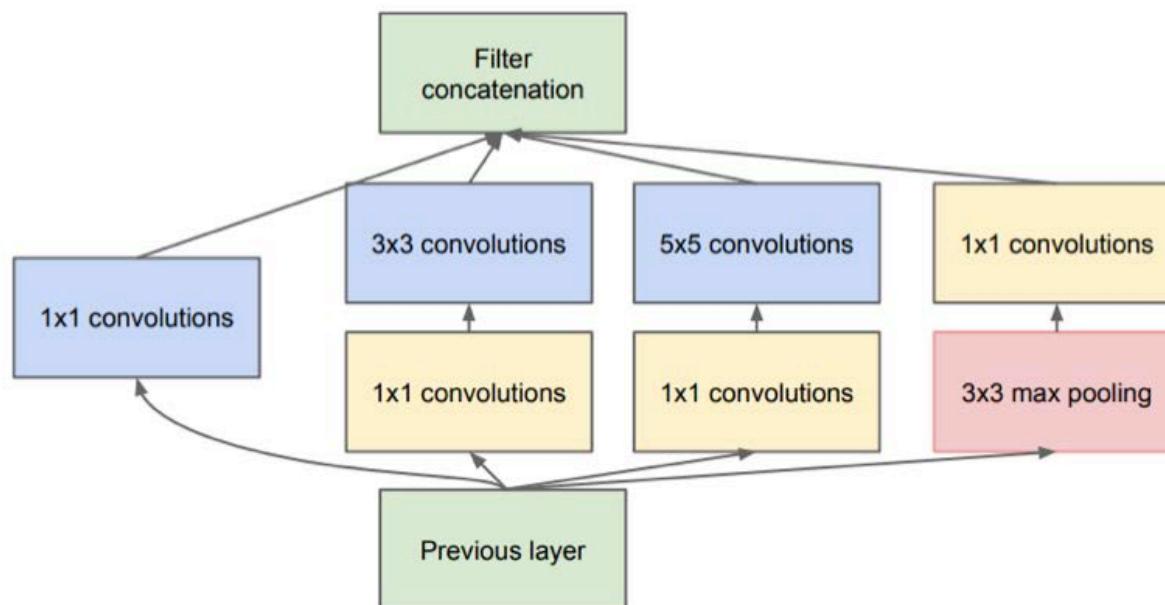
$$16 \times 16 \times (128 + 192 + 96 + 256) = 16 \times 16 \times 672 \quad 280M \text{ ops}$$



(a) Inception module, naïve version

Inception / GoogLeNet (2014)

- InceptionBlock (bottleneck-вариант)

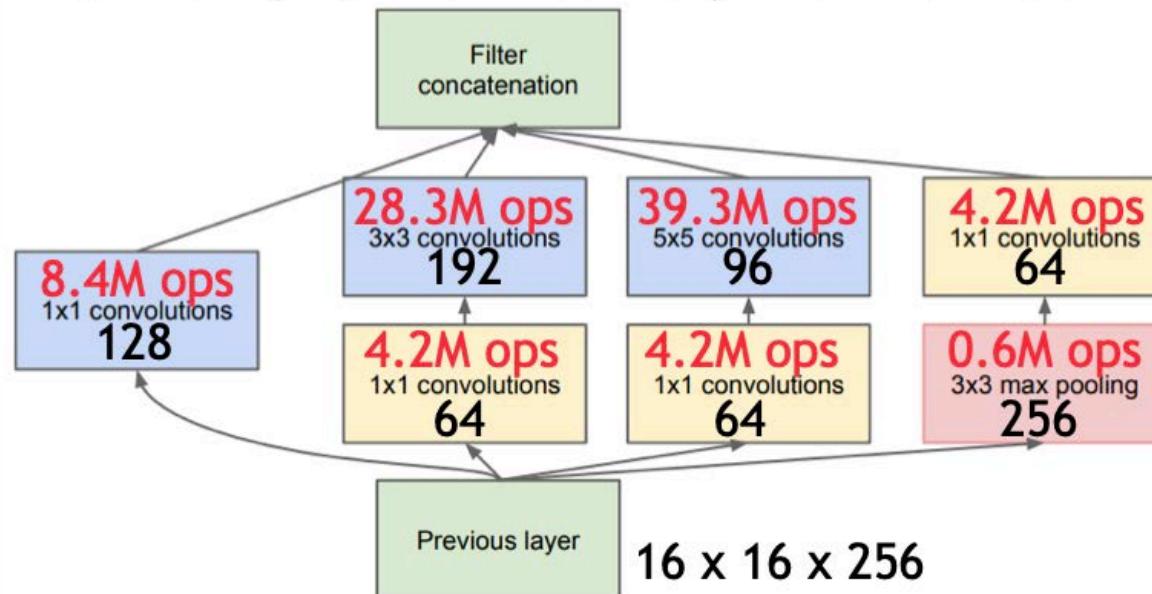


(b) Inception module with dimension reductions

Inception / GoogLeNet (2014)

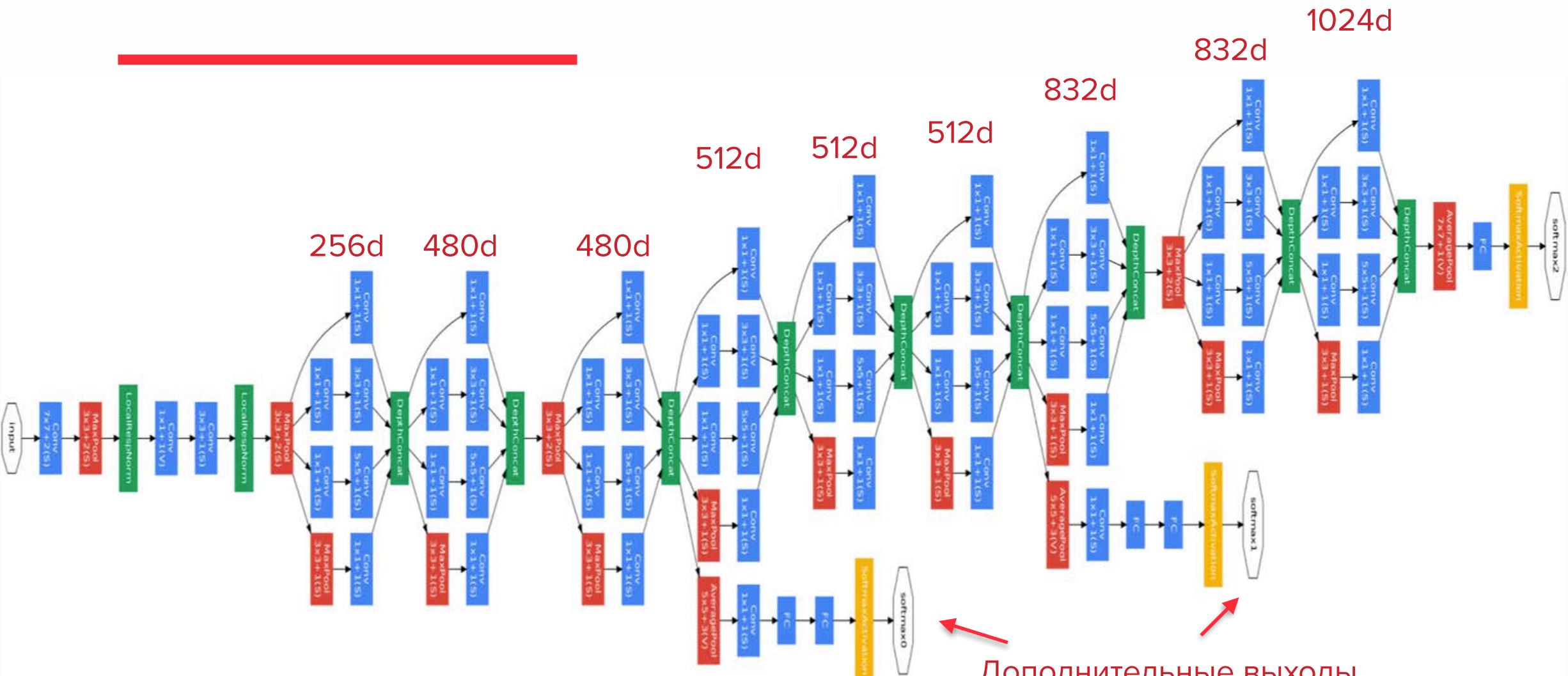
- InceptionBlock (bottleneck-вариант)

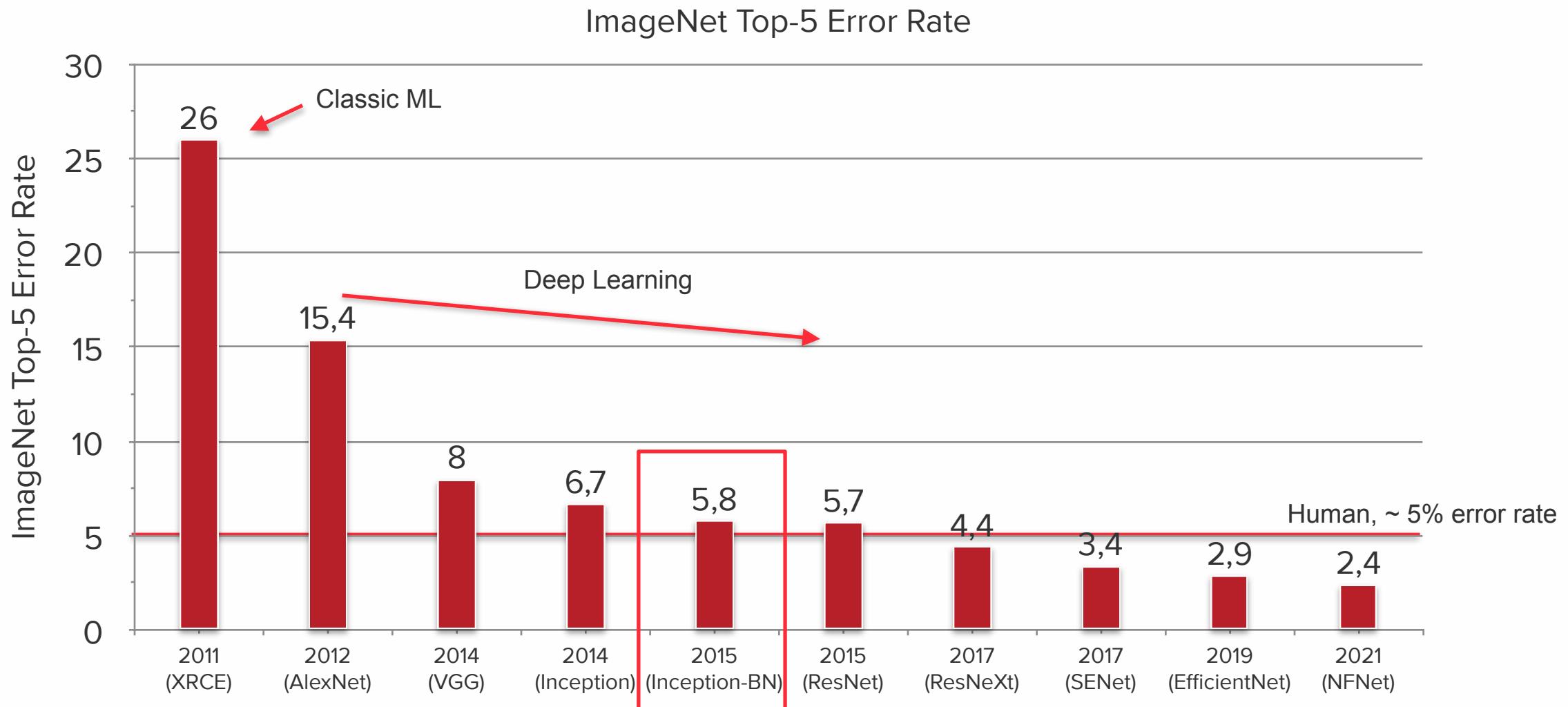
$$16 \times 16 \times (128 + 192 + 96 + 64) = 16 \times 16 \times 480 \quad 90M \text{ ops}$$



(b) Inception module with dimension reductions

Inception / GoogLeNet (2014)







Inception-BN / Inception-V2 (2015)

- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Новый слой нормализации **Batch Normalization**
 - Значительное ускорение сходимости при обучении



BatchNormalization

- Проблема: **internal covariate shift**
 - Обновление параметров слоя ведет к изменению распределения его выходных значений - эффект домино
 - Некоторые функции активации “насыщаются” (sigmoid, tanh) и “хорошо” пропускают градиент только в окрестности 0



BatchNormalization

- Пусть x - вектор активаций батча длины m где-то в сети

BatchNormalization

- Пусть x - вектор активаций батча длины m где-то в сети
- Посчитаем среднее (по батчу): $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$

BatchNormalization

- Пусть x - вектор активаций батча длины m где-то в сети
- Посчитаем среднее (по батчу): $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$
- Посчитаем дисперсию (по батчу): $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$

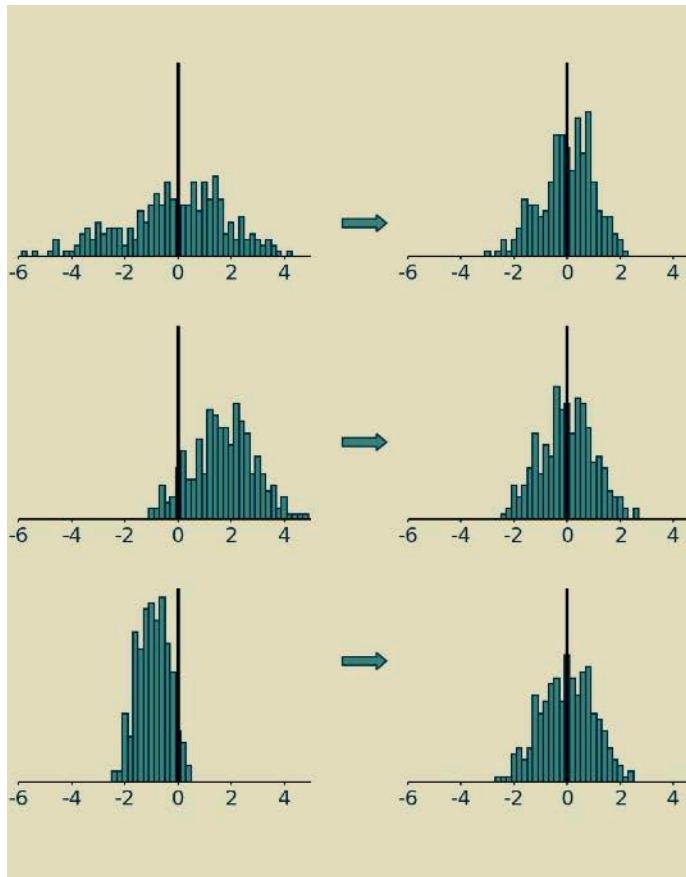
BatchNormalization

- Пусть x - вектор активаций батча длины m где-то в сети
- Посчитаем среднее (по батчу): $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$
- Посчитаем дисперсию (по батчу): $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$
- Отнормируем: $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$

BatchNormalization

- Пусть x - вектор активаций батча длины m где-то в сети
- Посчитаем среднее (по батчу): $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$
- Посчитаем дисперсию (по батчу): $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$
- Отнормируем: $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
- Добавим обучаемое масштабирование: $BN_{\gamma, \beta}(x_i) = \gamma \hat{x}_i + \beta$

BatchNormalization



BatchNormalization

$$BN_{\gamma, \beta}(x_i) = \gamma \hat{x}_i + \beta = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

BatchNormalization

Обучаемые параметры
(могут отсутствовать)

$$BN_{\gamma, \beta}(x_i) = \gamma \hat{x}_i + \beta = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Обучение: вычисляются
по батчу

Инференс: используются
пред-вычисленные по
обучающей выборке

BatchNormalization - backprop?

$$BN_{\gamma, \beta}(x_i) = \gamma \hat{x}_i + \beta = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

- μ_B и σ_B зависят от x_i
- Проверить себя можно, например, [здесь](#)

BatchNormalization

- Ускорение сходимости

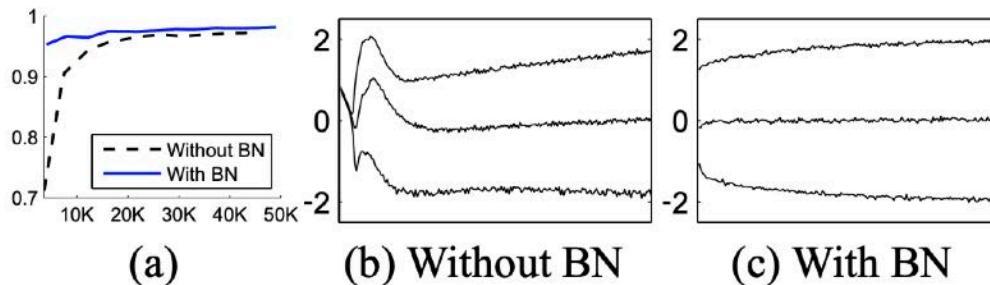


Figure 1: (a) The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy. (b, c) The evolution of input distributions to a typical sigmoid, over the course of training, shown as $\{15, 50, 85\}$ th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.



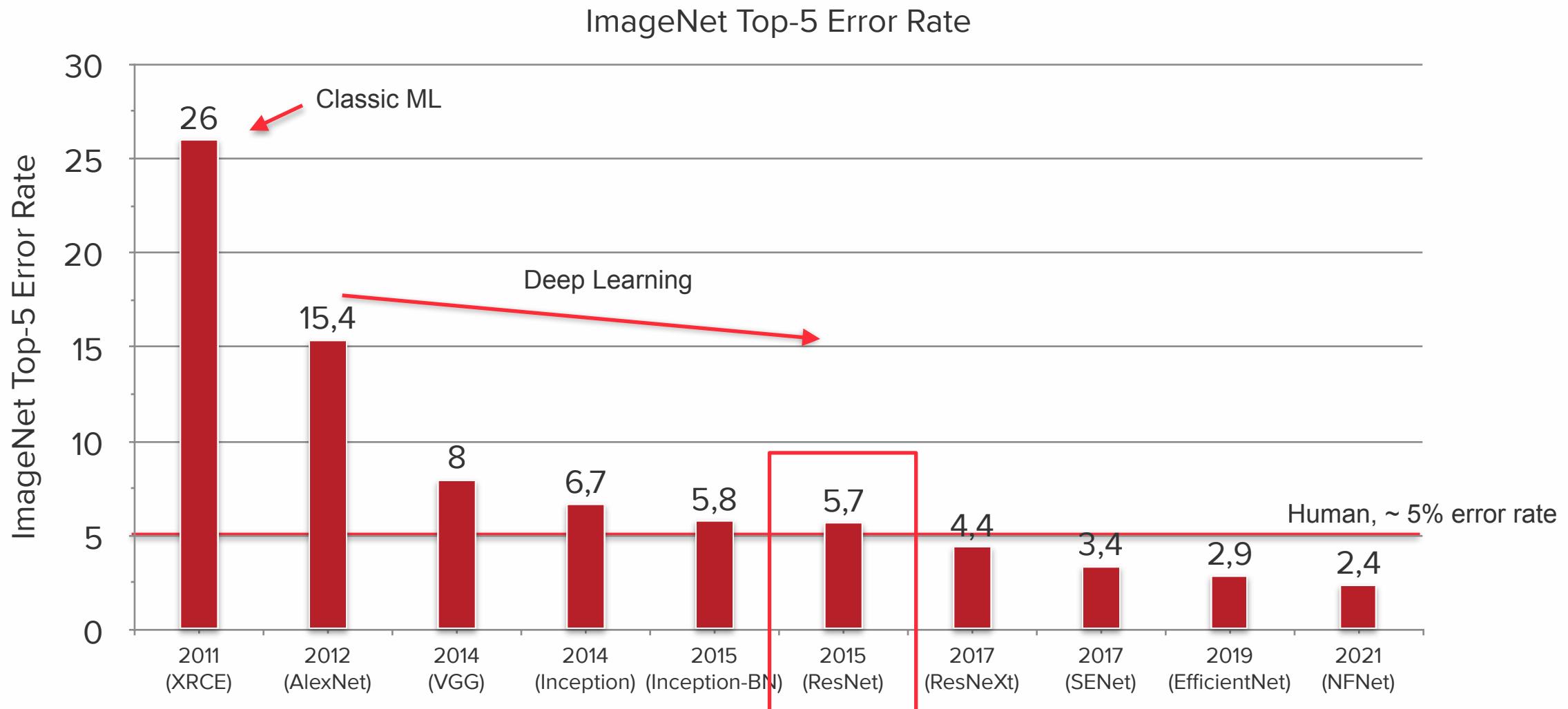
BatchNormalization

- Ускорение сходимости
 - Добавление регуляризующего эффекта
 - Повышение устойчивости к инициализации весов
-
- Требуется достаточно большой батч при обучении
 - Отличается поведение не обучении и инференсе
 - Требует дополнительных ресурсов (память, вычисления)



BatchNormalization

- How Does Batch Normalization Help Optimization?





ResNet (2015)

- Deep Residual Learning for Image Recognition
- Наращивание глубины сети с помощью **Residual Connections**
- До **152** слоев!

ResNet (2015)

- Наблюдение: увеличение глубины сети не обязательно приводит к улучшению качества

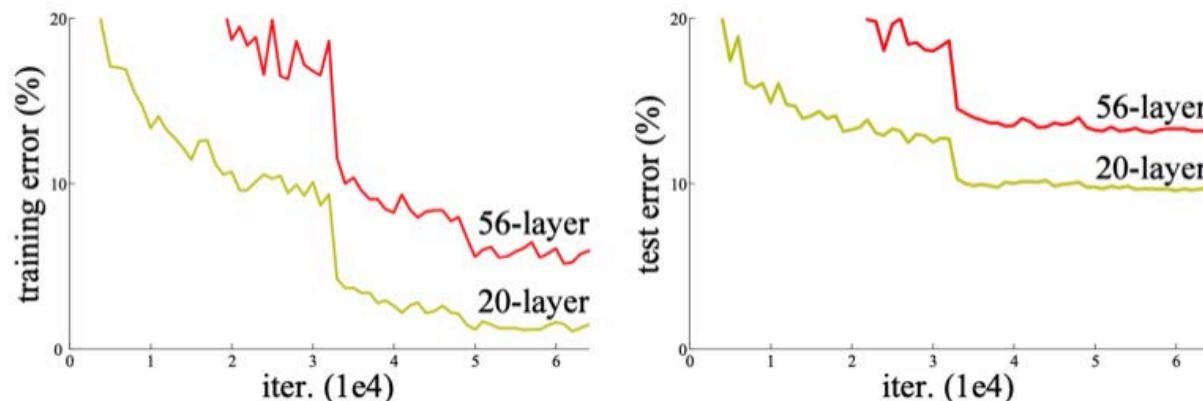


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не улучшают качество

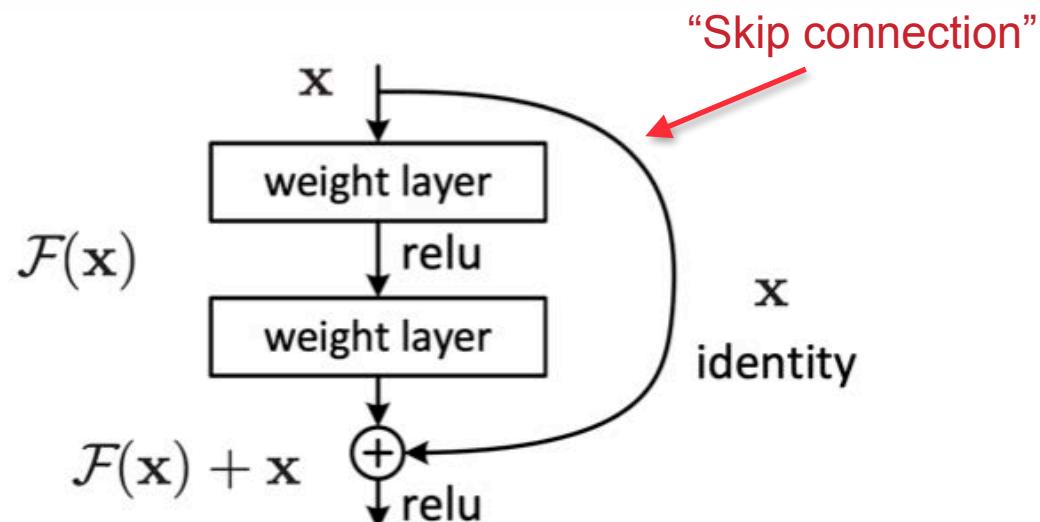


Figure 2. Residual learning: a building block.

ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не улучшают качество

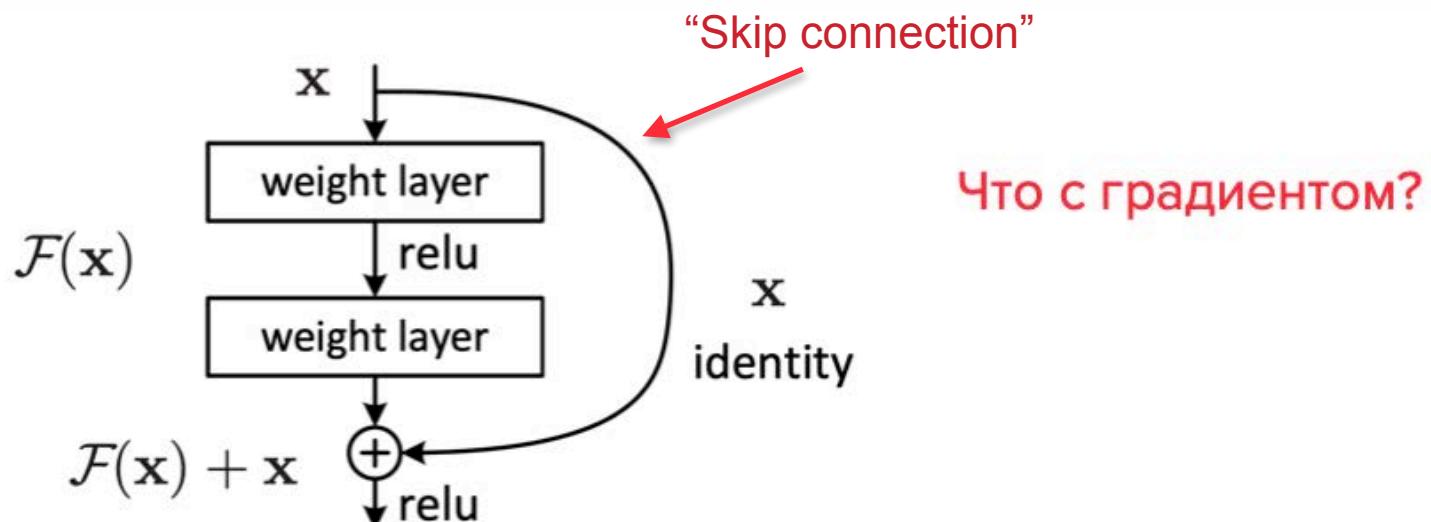


Figure 2. Residual learning: a building block.

ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не улучшают качество

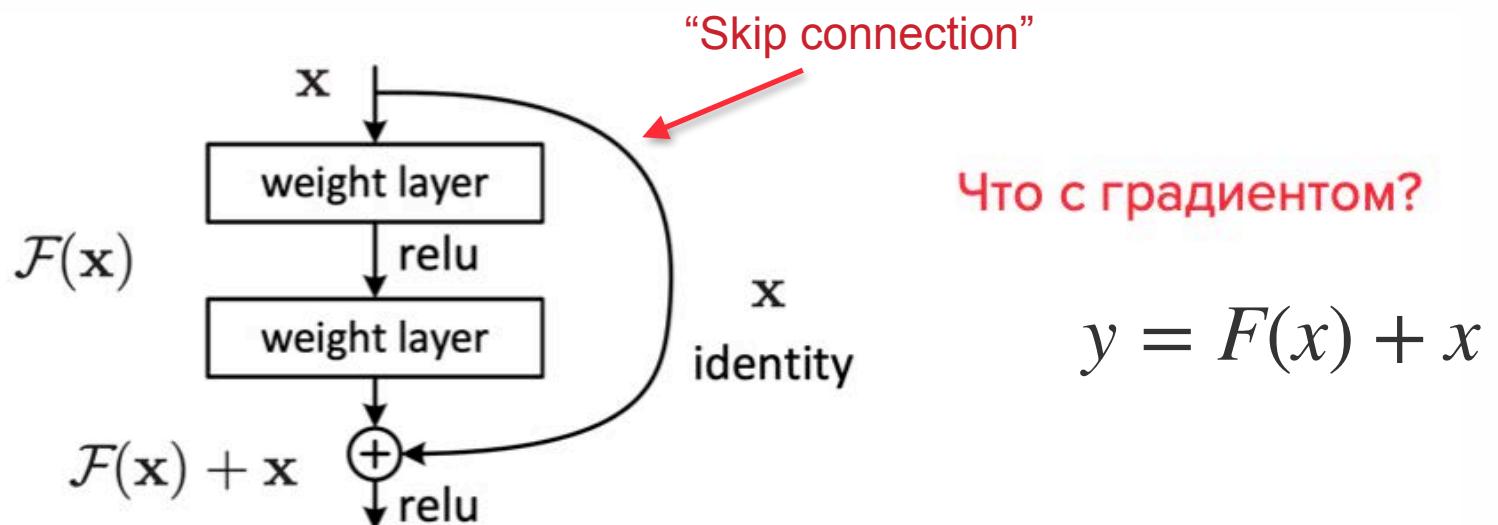


Figure 2. Residual learning: a building block.

ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не улучшают качество

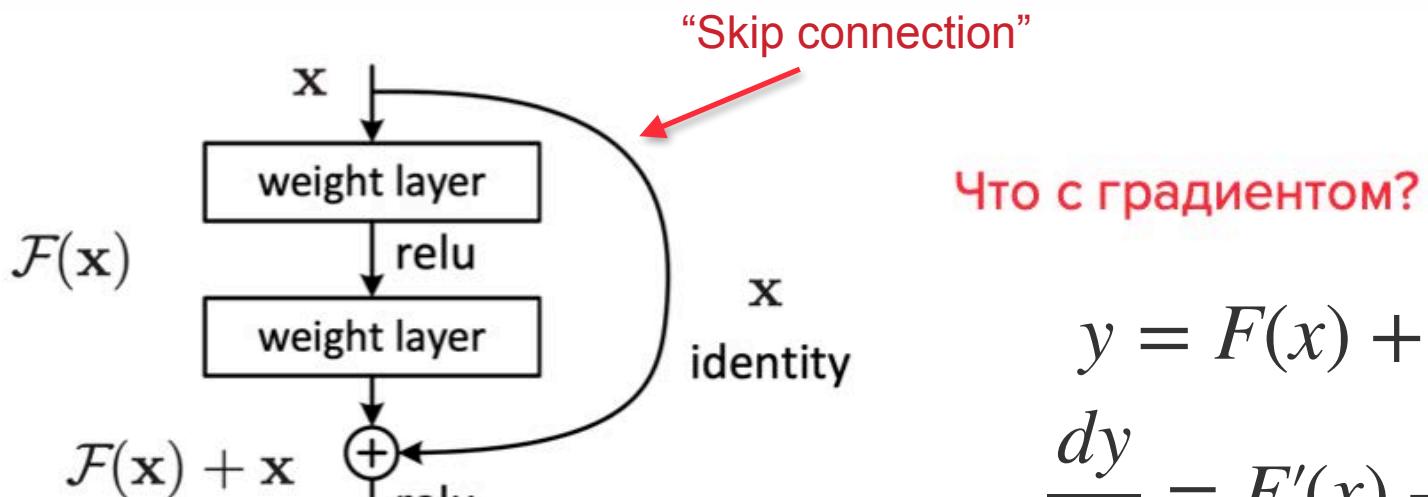


Figure 2. Residual learning: a building block.

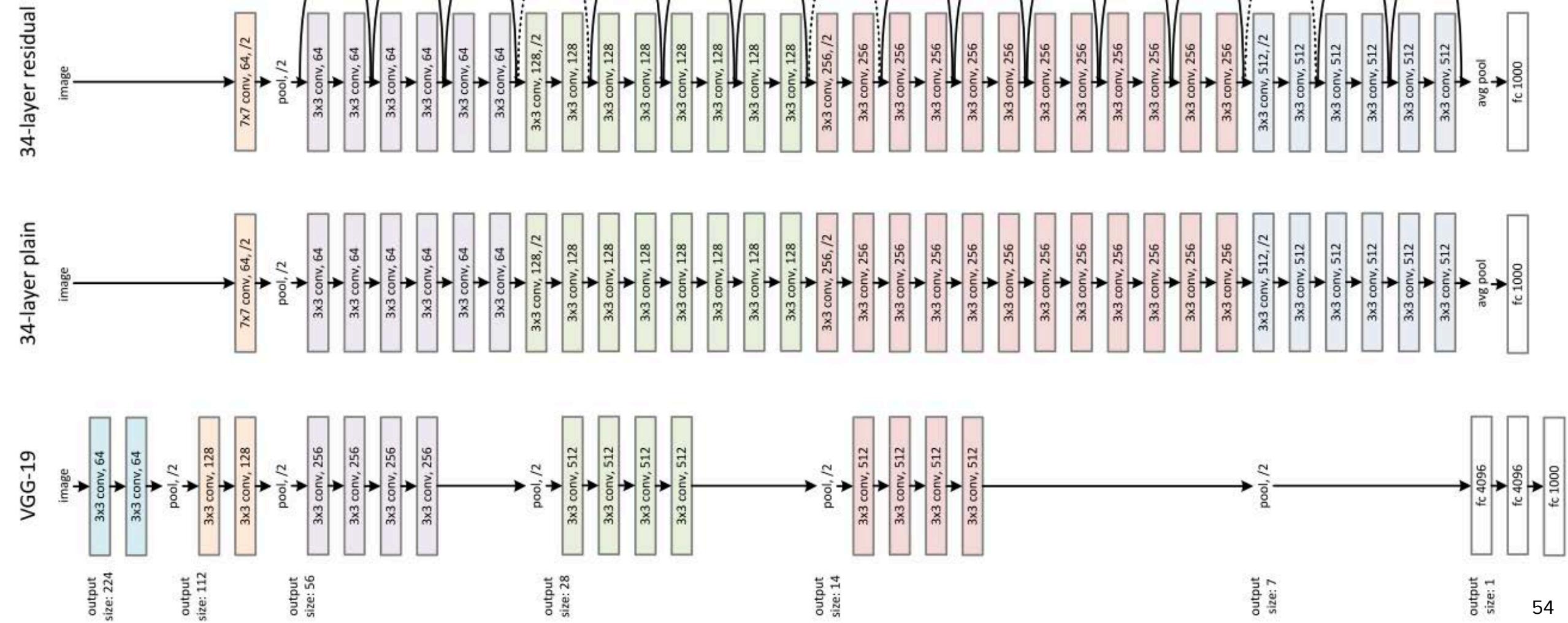
Что с градиентом?

$$y = F(x) + x$$

$$\frac{dy}{dx} = F'(x) + 1$$

ResNet (2015)

Рост ширины блоков
+ пулинг



ResNet (2015)

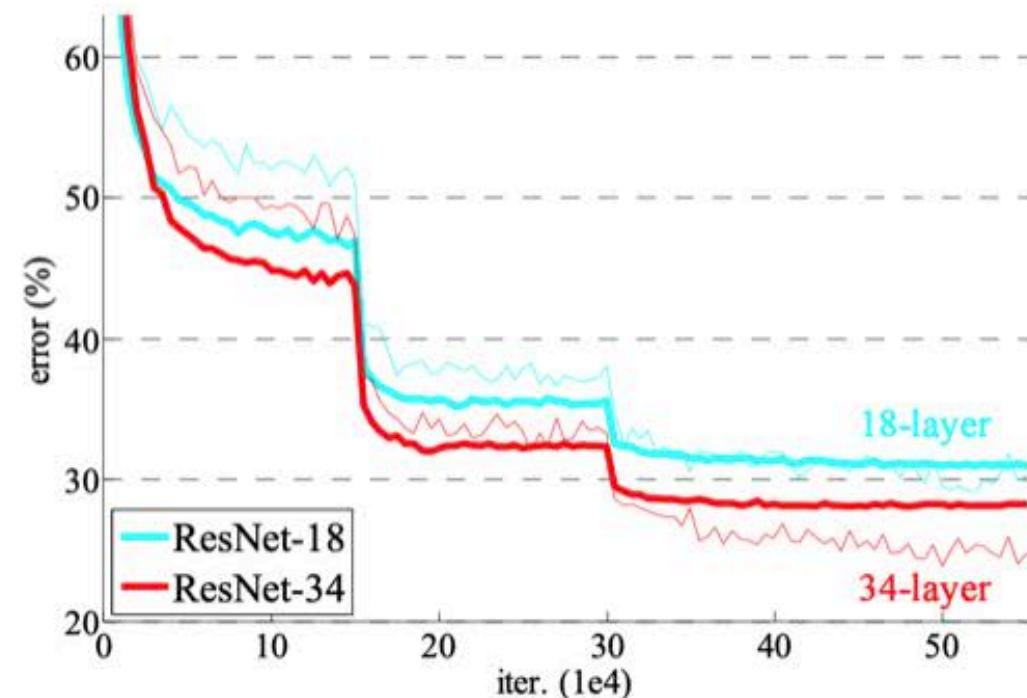
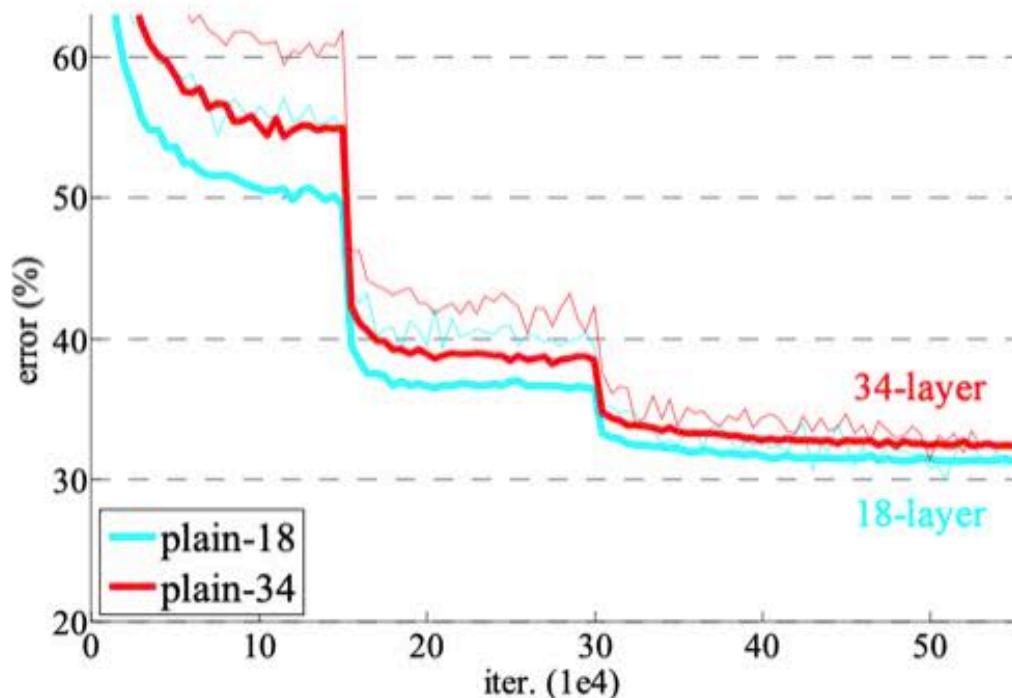


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet (2015)

- 2 типа базовых блоков
 - ResNet-18 / 34: “обычный” блок (слева)
 - ResNet-50 / 101 / 152: bottleneck-блок (справа)

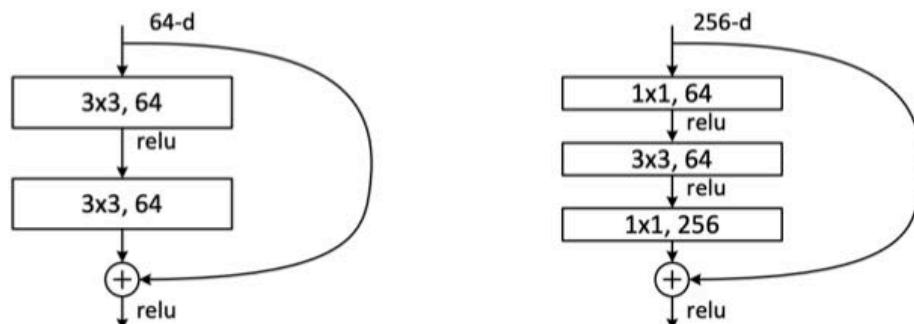
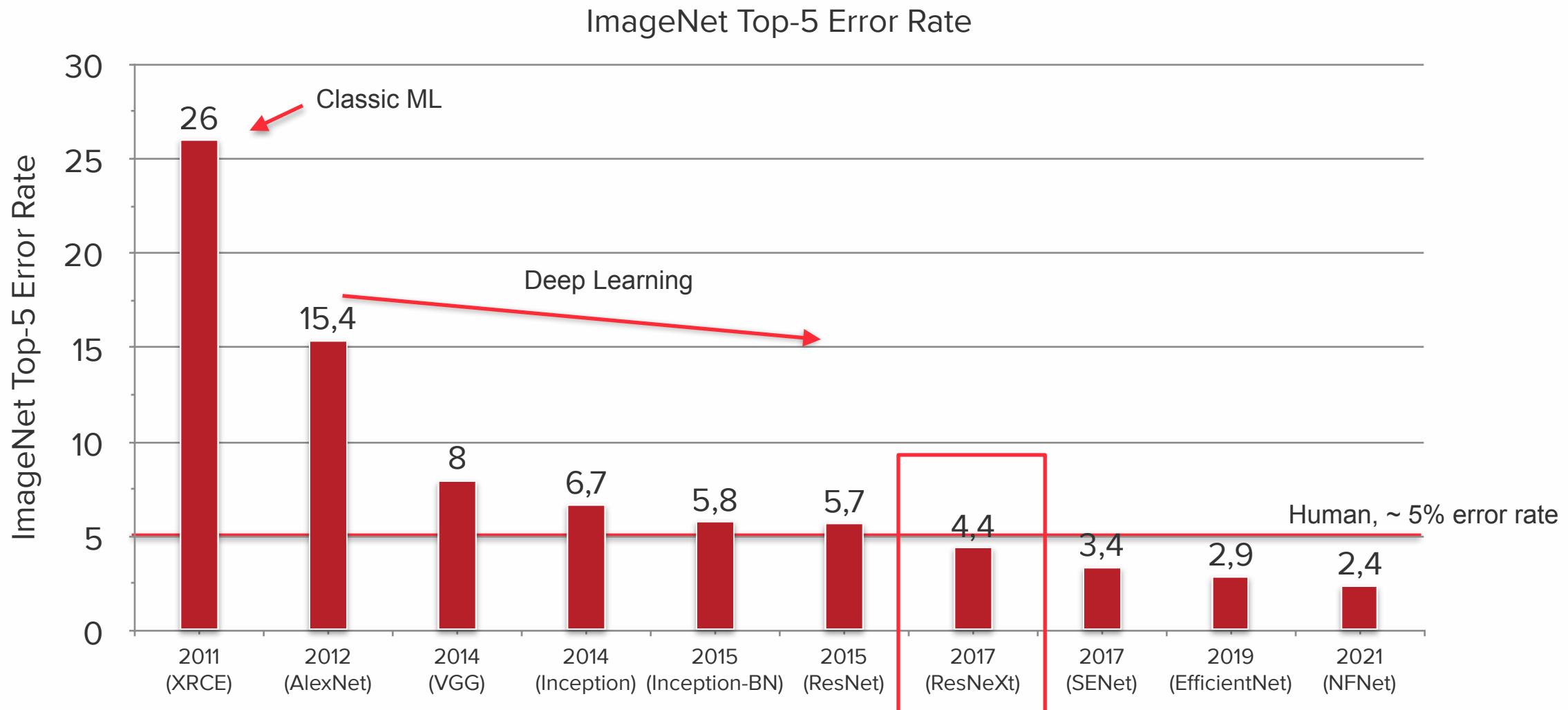


Figure 5. A deeper residual function \mathcal{F} for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.



ResNeXt (2015)

- [Aggregated Residual Transformations for Deep Neural Networks](#)
- Совмещение идей о **параллельных вычислениях** в рамках одного блока (Inception) и **Residual Connections** (ResNet)



Why don't we have both?

ResNeXt (2015)

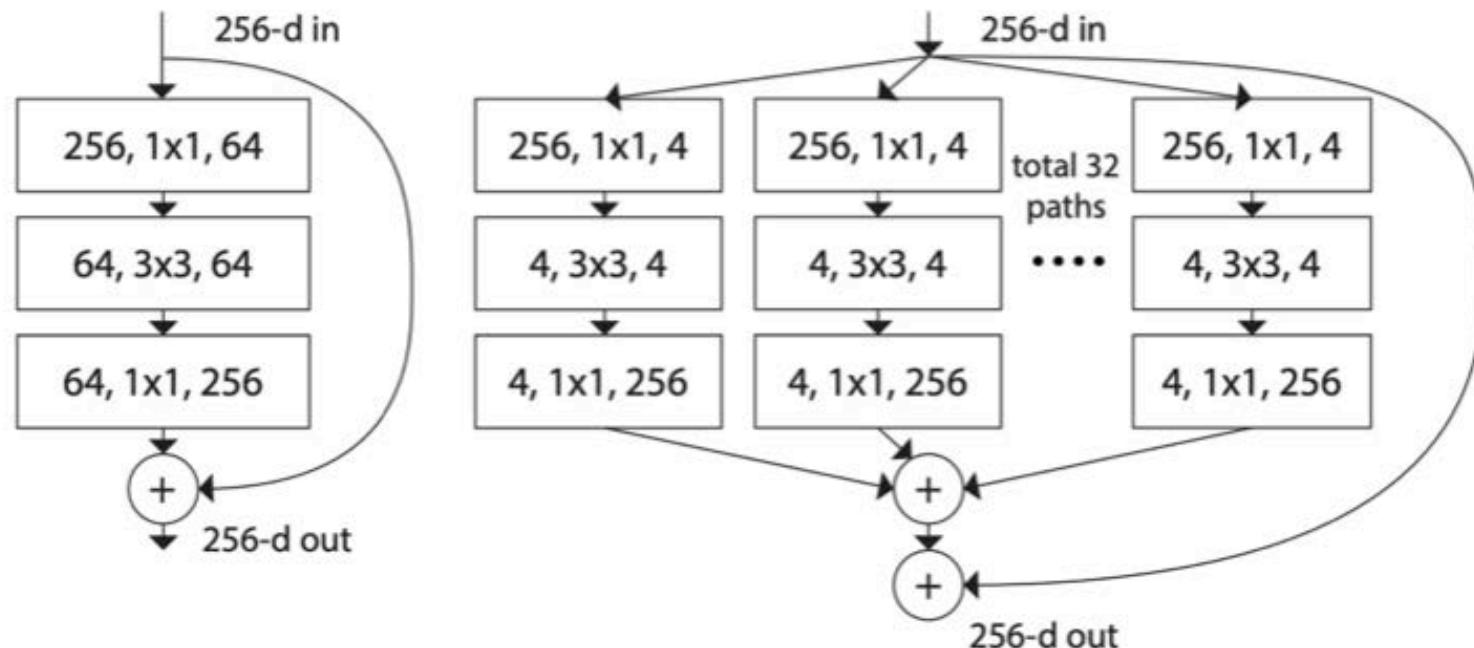
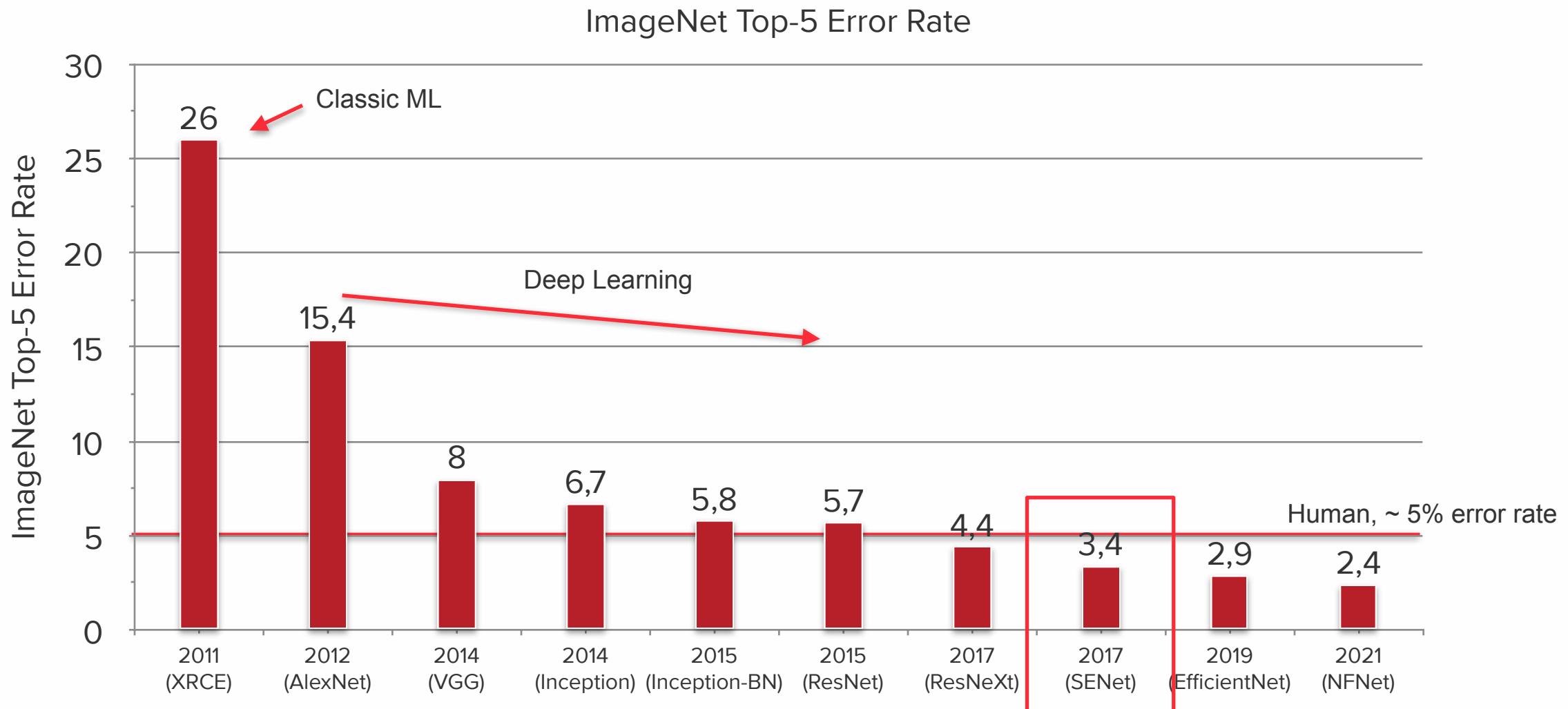


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).





Squeeze-n-Excitation (SENet) (2017)

- Squeeze-and-Excitation Networks
- Идея о перевзвешивании карт активаций
 - Не все признаки одинаково полезны

Squeeze-n-Excitation (SENet) (2017)

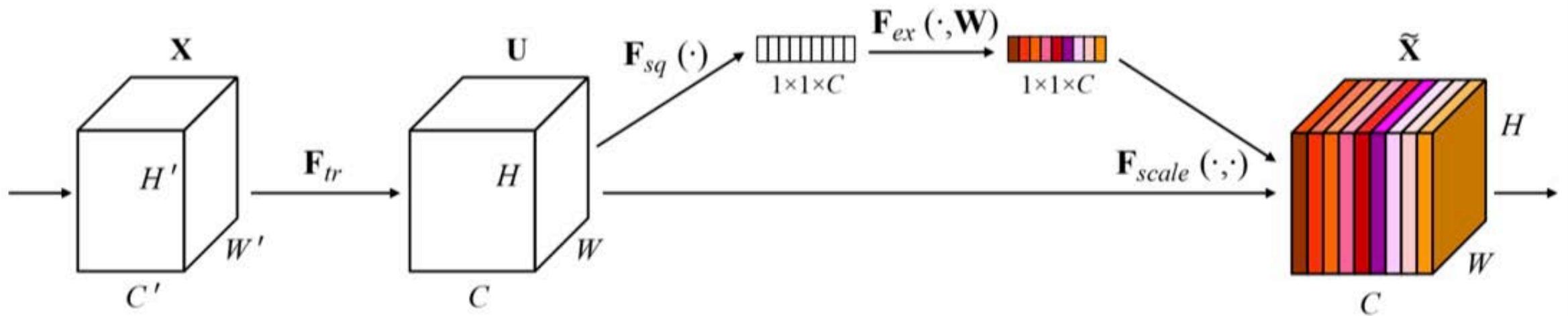
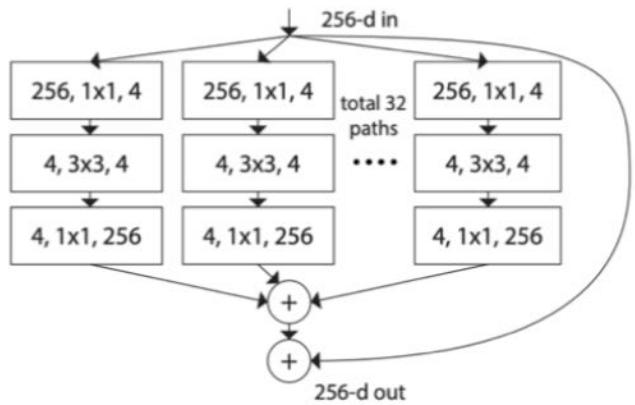


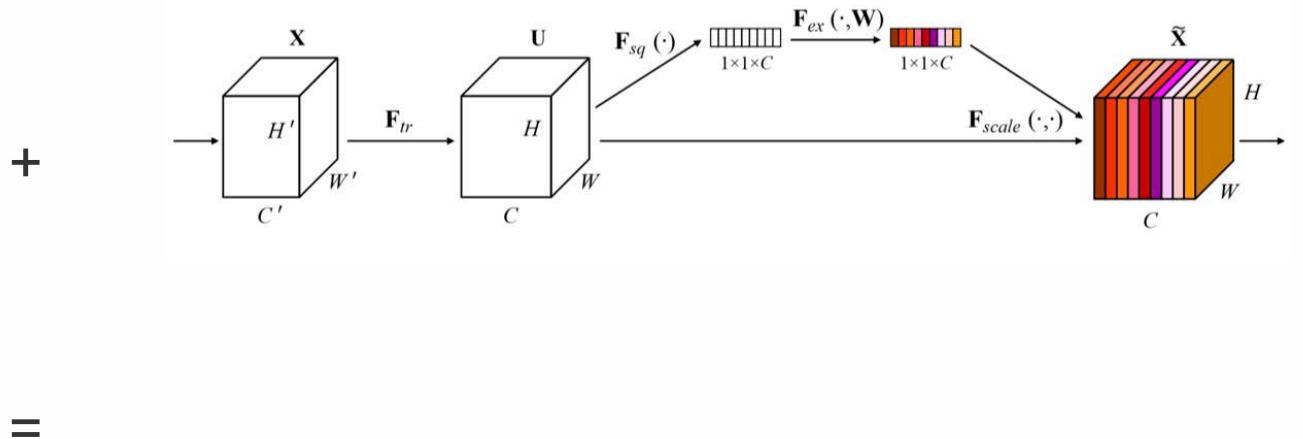
Fig. 1. A Squeeze-and-Excitation block.

SE-ResNeXt-...

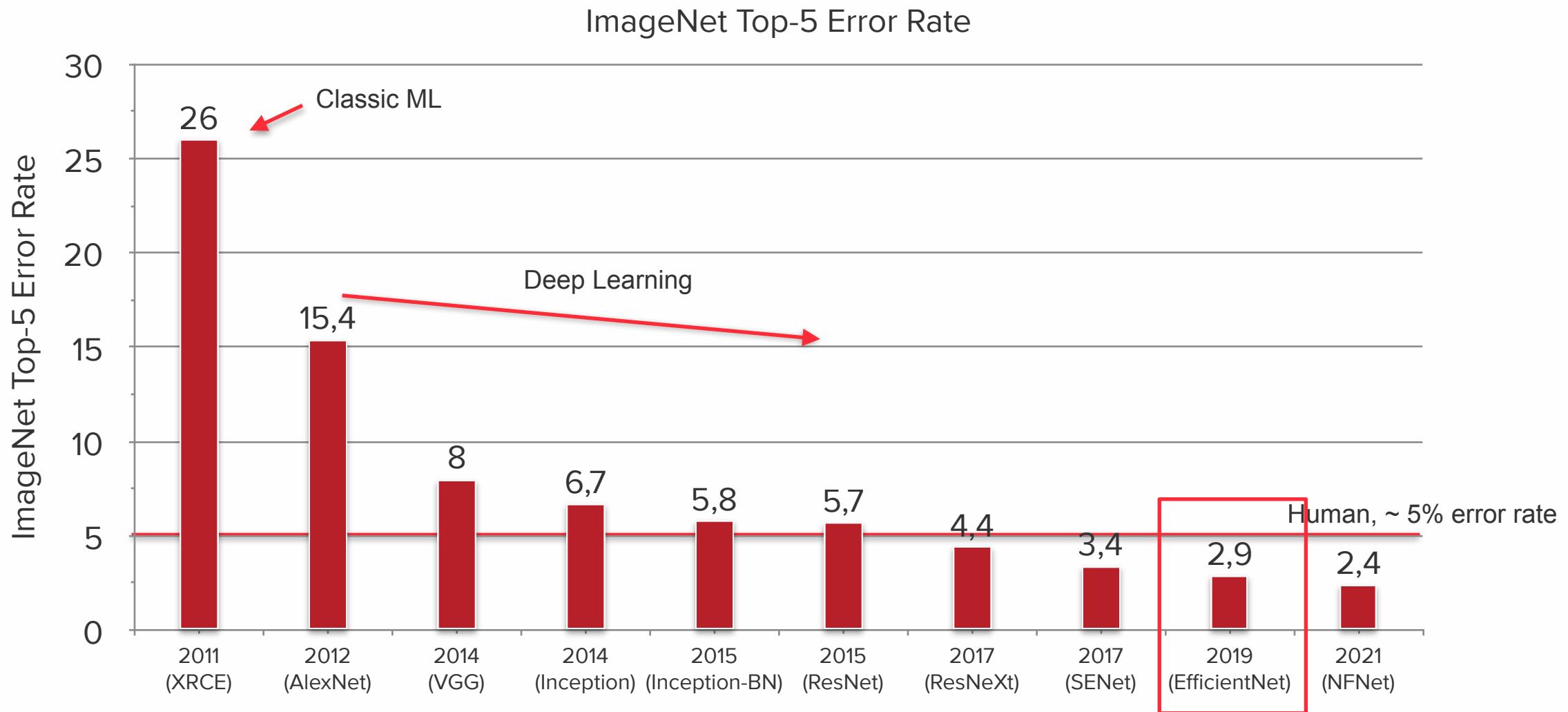
ResNeXt-bottleneck



SE-block



SE-ResNeXt50_32x4d





EfficientNet (2019)

- [EfficientNet: Rethinking Model Scaling for CNNs](#)
- Цель - подобрать лучшую архитектуру при ограниченных ресурсах на вычисления
- Идея: подобрать способ эффективного масштабирования размерностей моделей
 - Размер входного изображения
 - Число фильтров в сверточных слоях (“ширина”)
 - Число сверточных слоев (“глубина”)

EfficientNet (2019)

- Увеличение одной из размерностей не дает “бесконечного улучшения” качества:

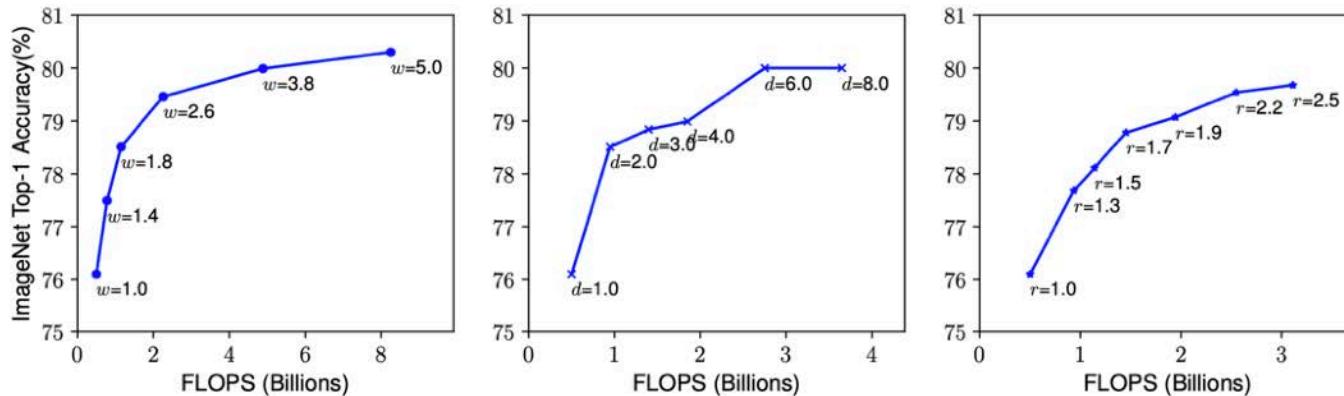


Figure 3. Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients. Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturates after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

- Масштабировать разные размерности модели следует совместно

EfficientNet (2019)

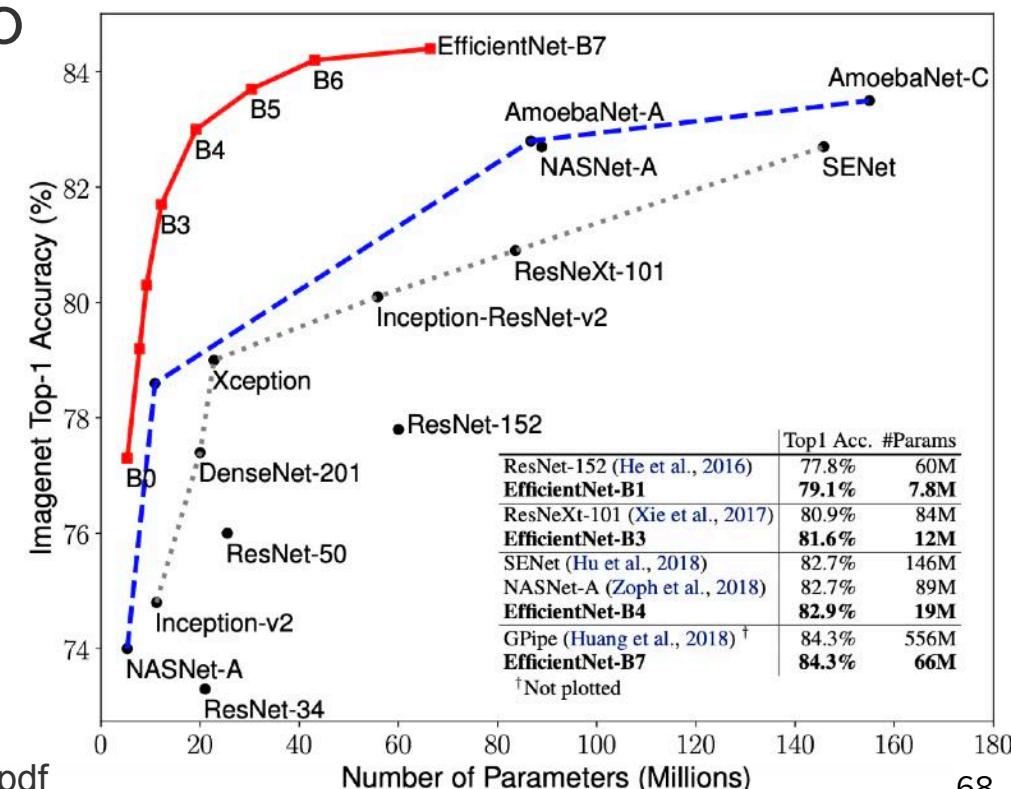
- Глубина влияет на вычислительную сложность \sim линейно, ширина и размер входа \sim квадратично
- При условиях ниже увеличение размеров в ϕ раз увеличит сложность модели в 2^ϕ раз

In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } d &= \alpha^\phi \\ \text{width: } w &= \beta^\phi \\ \text{resolution: } r &= \gamma^\phi \\ \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

EfficientNet (2019)

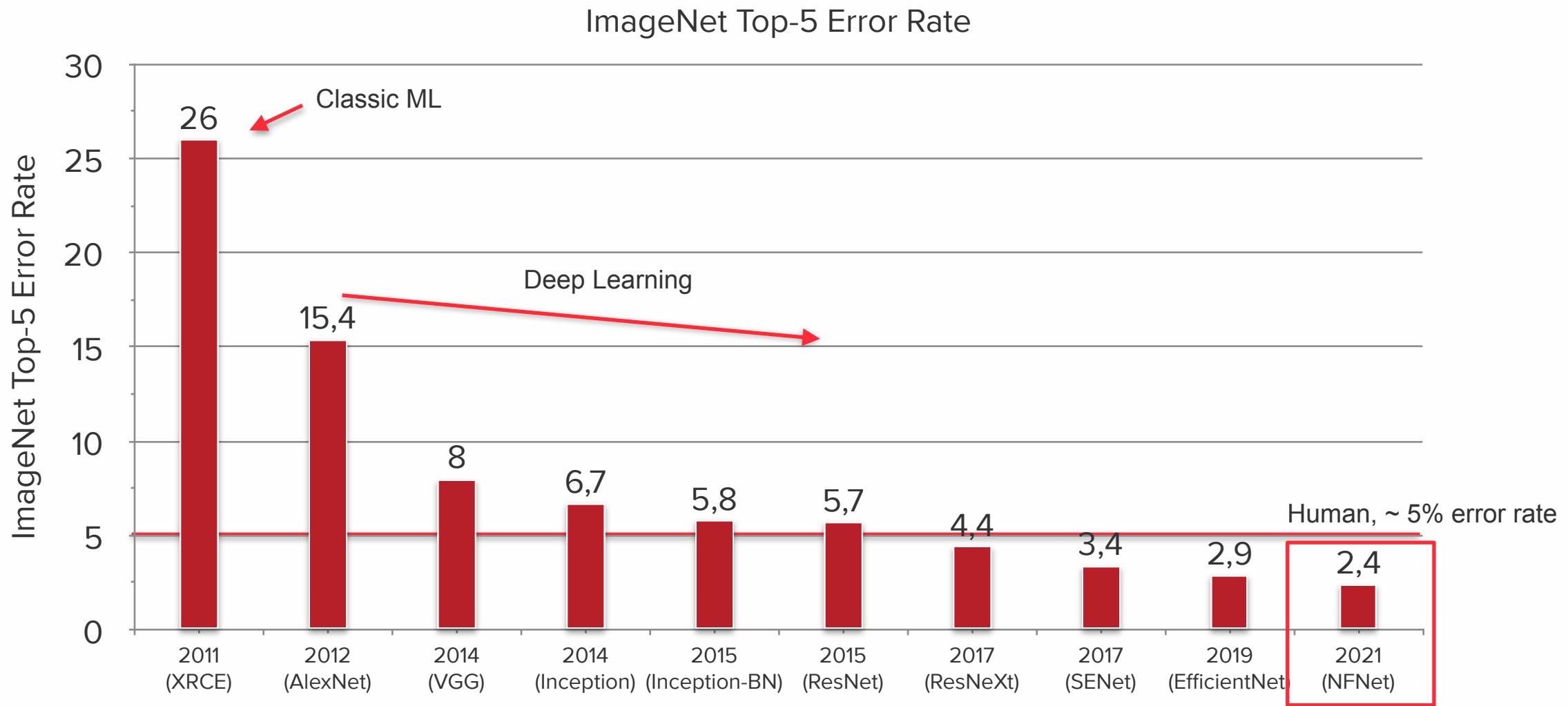
- Подобрали α, β, γ
- Подобрали архитектуру базового блока для $\phi = 1$
- Увеличивали ϕ , получили семейство EfficientNetB0-B7 возрастающей сложности





Neural Architecture Search

- Архитектура базового блока EfficientNet подбиралась с помощью **Neural Architecture Search (NAS)**
- Гиперпараметры в NAS подбираются автоматически на основе целевых метрик
 - Например, с помощью эволюционных алгоритмов
- Что почитать: [публикация](#), [пост](#)





NFNet (2021)

- High-Performance Large-Scale Image Recognition Without Normalization
- Избавились от BatchNorm, сэкономив ресурсы
 - Идея: понять, каковы основные преимущества использования BatchNorm, и сымитировать их, убрав сам BN
 - Ускорили \sim вдвое 1 итерацию обучения
 - Побили EfficientNet с тем же временем обучения
- Тоже использовали Neural Architecture Search

NFNet (2021)

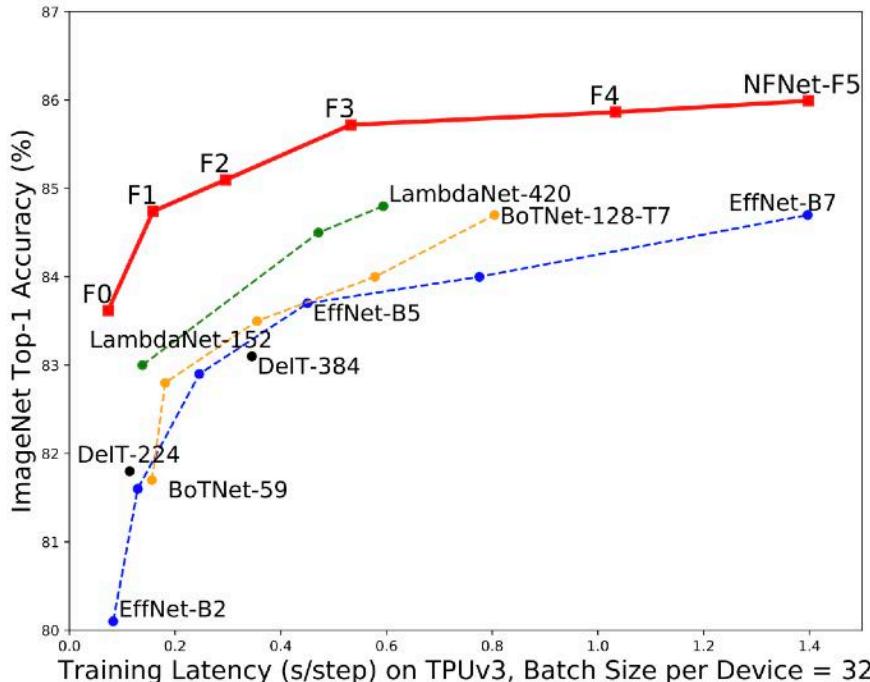


Figure 1. ImageNet Validation Accuracy vs Training Latency.

All numbers are single-model, single crop. Our NFNet-F1 model achieves comparable accuracy to an EffNet-B7 while being 8.7 \times faster to train. Our NFNet-F5 model has similar training latency to EffNet-B7, but achieves a state-of-the-art 86.0% top-1 accuracy on ImageNet. We further improve on this using Sharpness Aware Minimization (Foret et al., 2021) to achieve 86.5% top-1 accuracy.

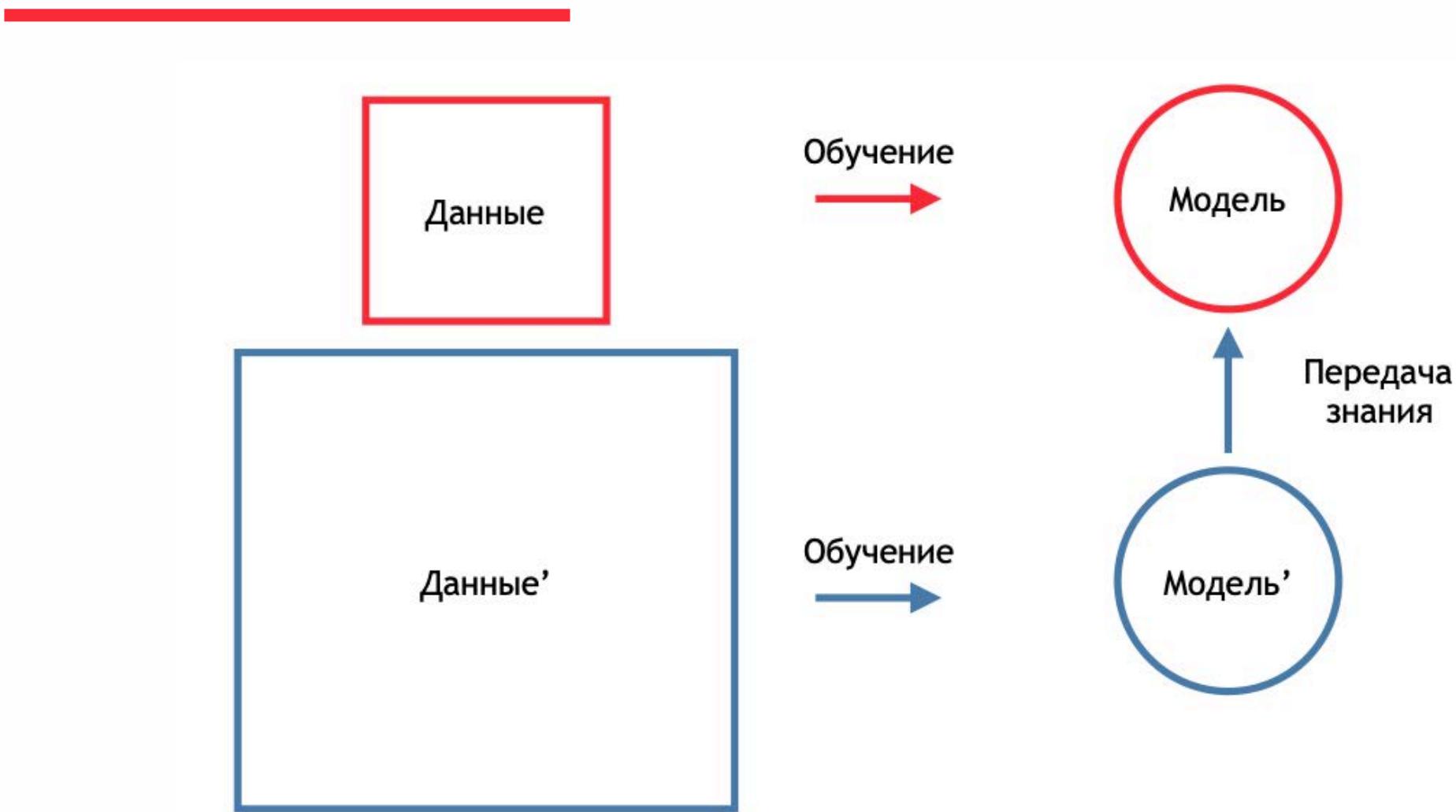
Transfer Learning



Традиционное обучение



Transfer Learning



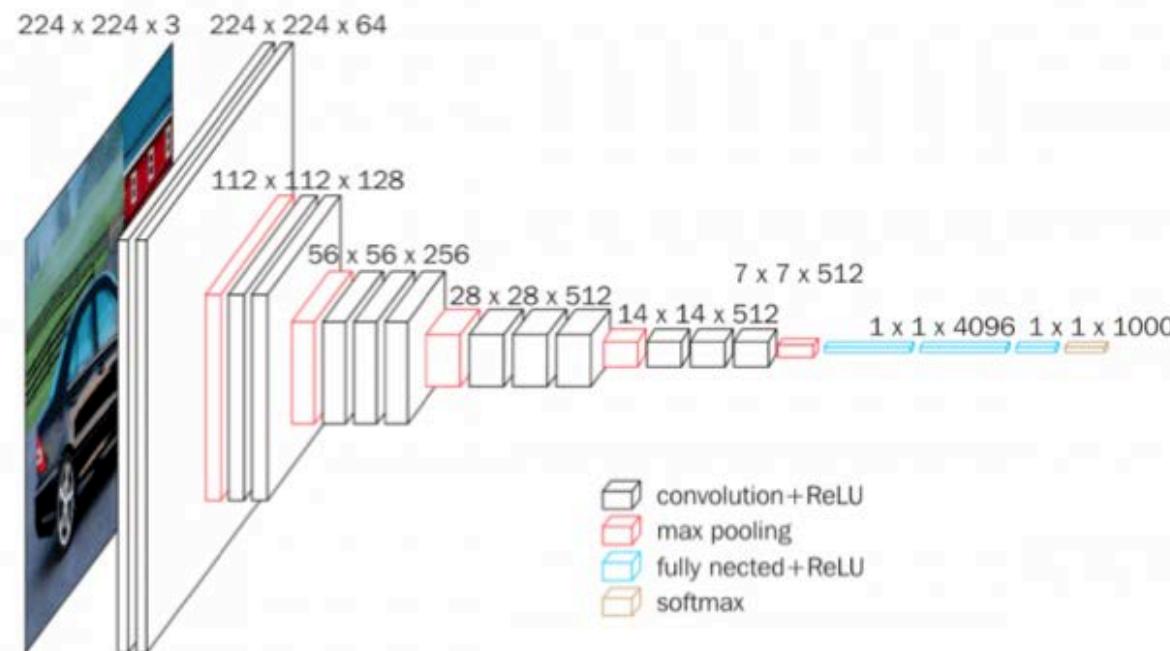


Fine-tuning

- Признаки из “универсальных” датасетов (очень) часто подойдут и для вашей задачи (**fine-tuning**)
 - Чем ближе слой к началу сети, тем более “примитивны” извлекаемые признаки
- Идея: брать предобученные модели и дообучать на своих данных
- Датасеты: ImageNet, COCO, OpenImages, ...

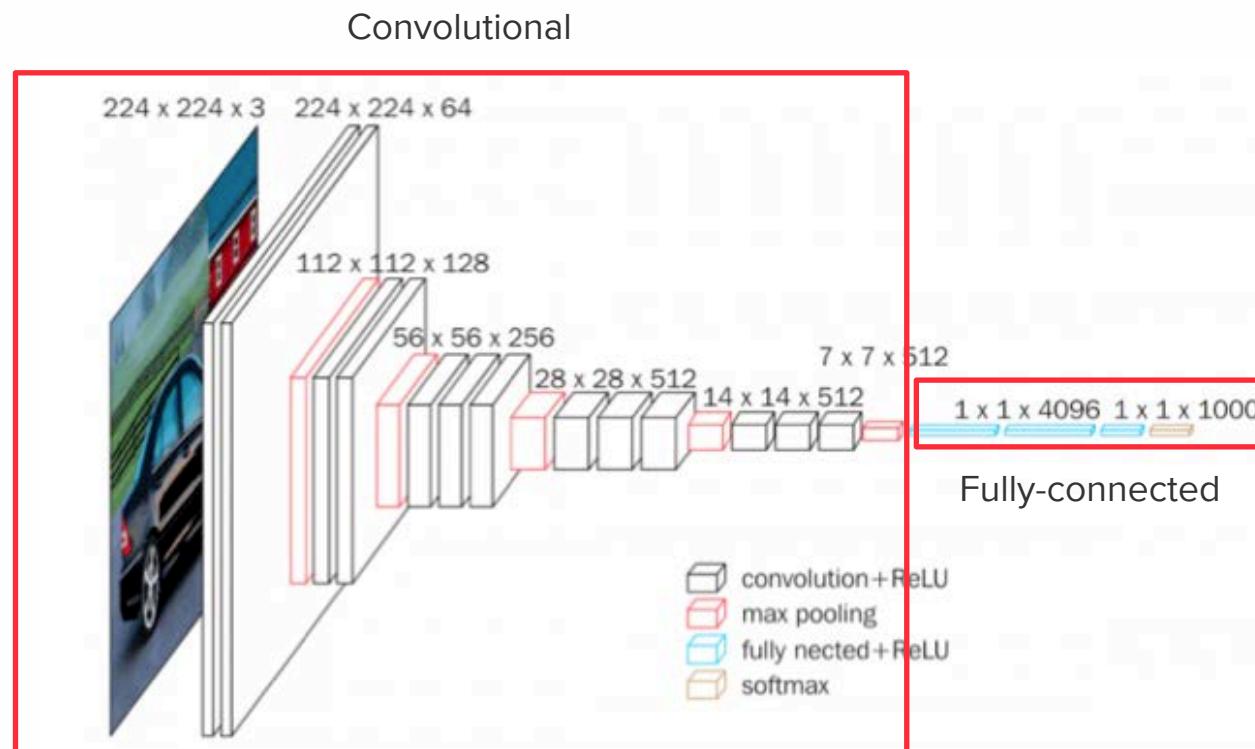
Fine-tuning

- Типичная структура CNN: сверточные слои + классификатор



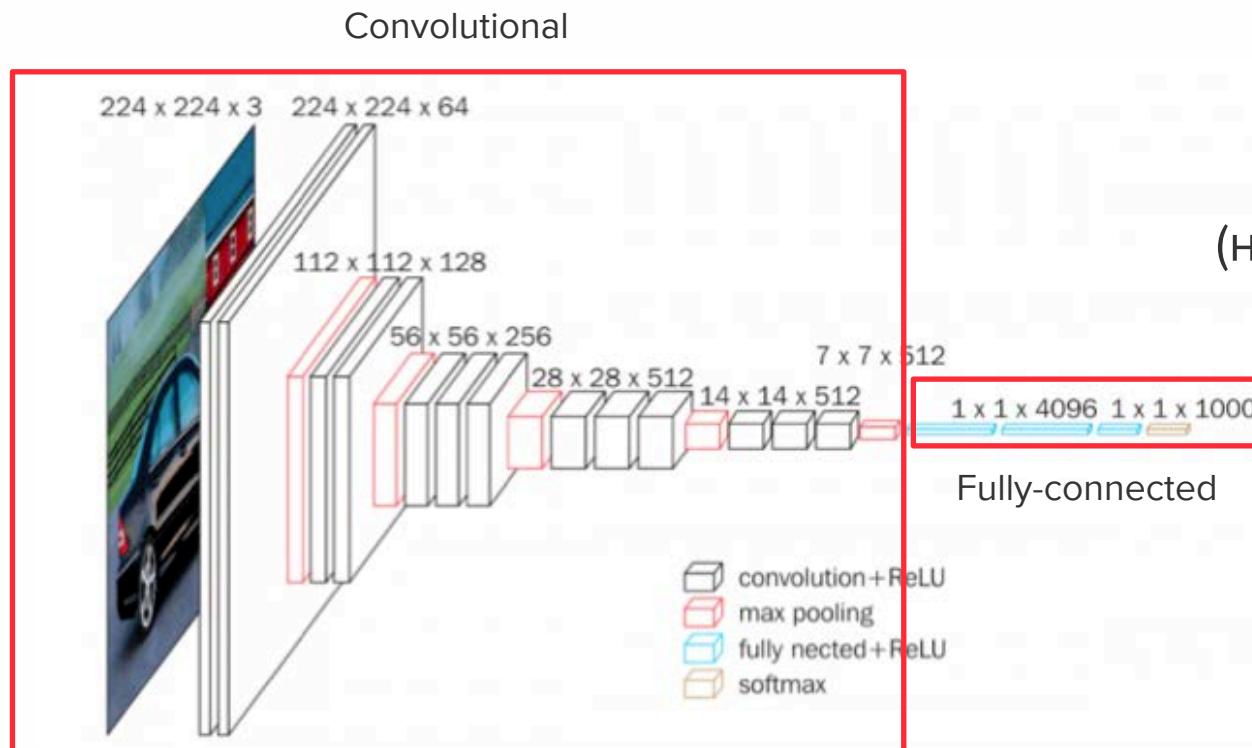
Fine-tuning

- Типичная структура CNN: сверточные слои + классификатор



Fine-tuning

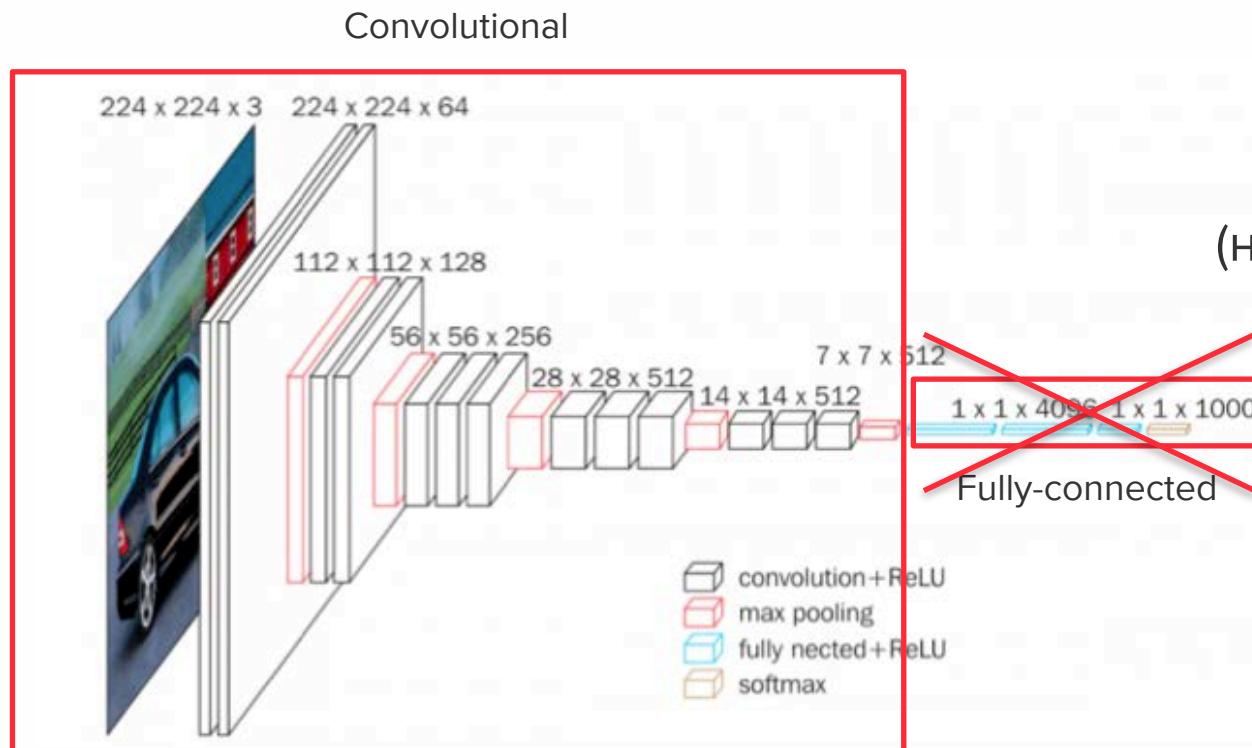
- Типичная структура CNN: сверточные слои + классификатор



Шаг 1: добавим
Global Average Pooling,
заменим FC-часть
(на нужное число выходов)

Fine-tuning

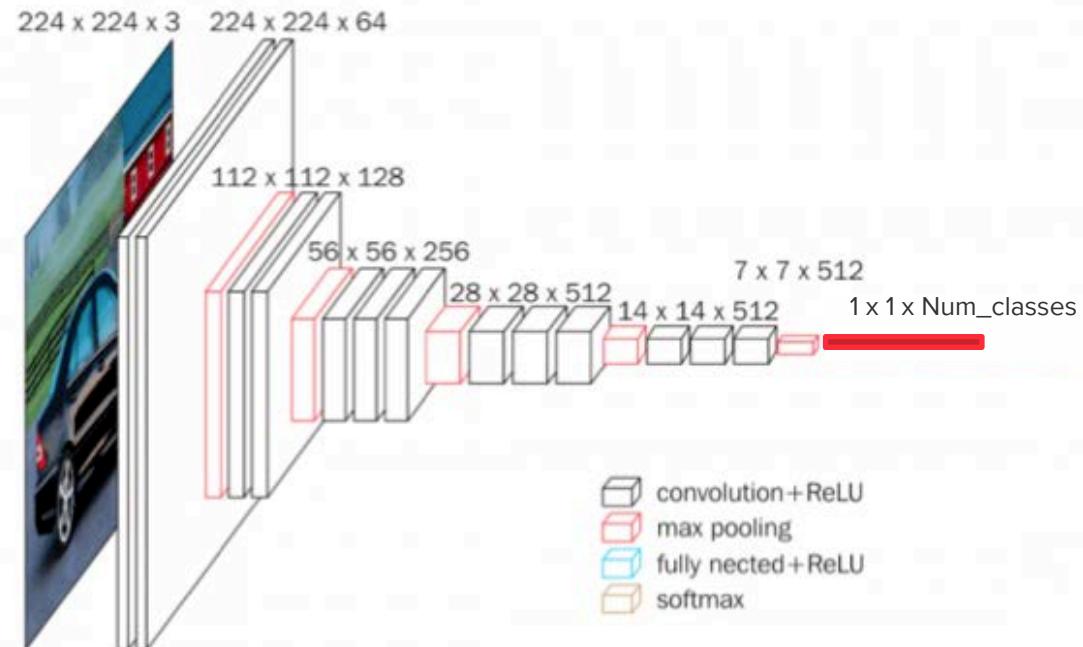
- Типичная структура CNN: сверточные слои + классификатор



Шаг 1: добавим
Global Average Pooling,
заменим FC-часть
(на нужное число выходов)

Fine-tuning

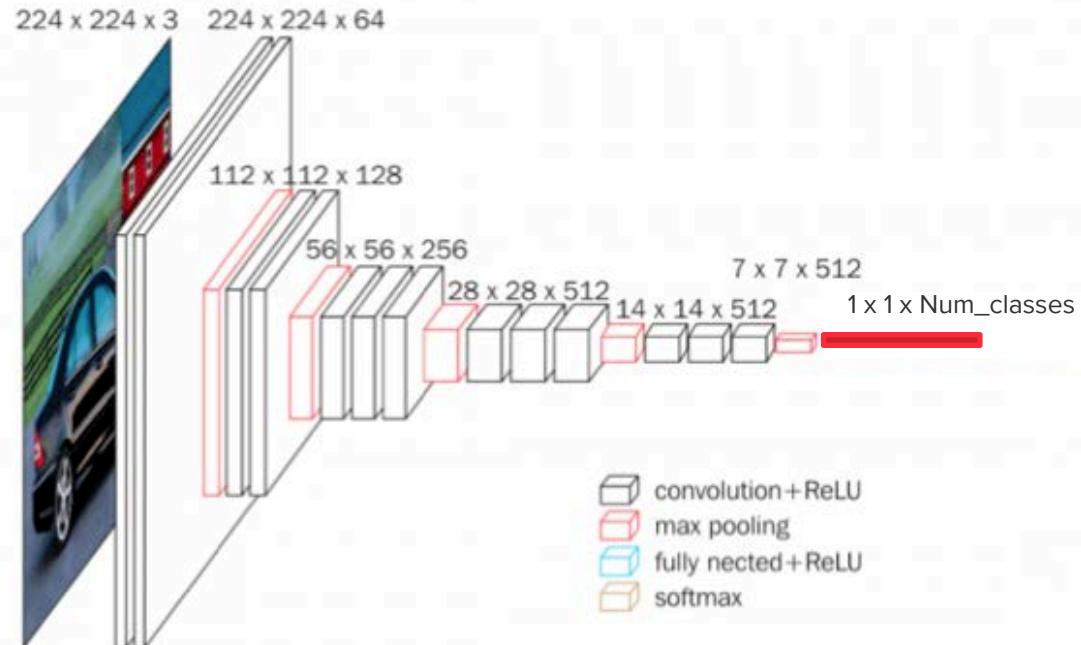
- Типичная структура CNN: сверточные слои + классификатор



Шаг 1: добавим
Global Average Pooling,
заменим FC-часть
(на нужное число выходов)

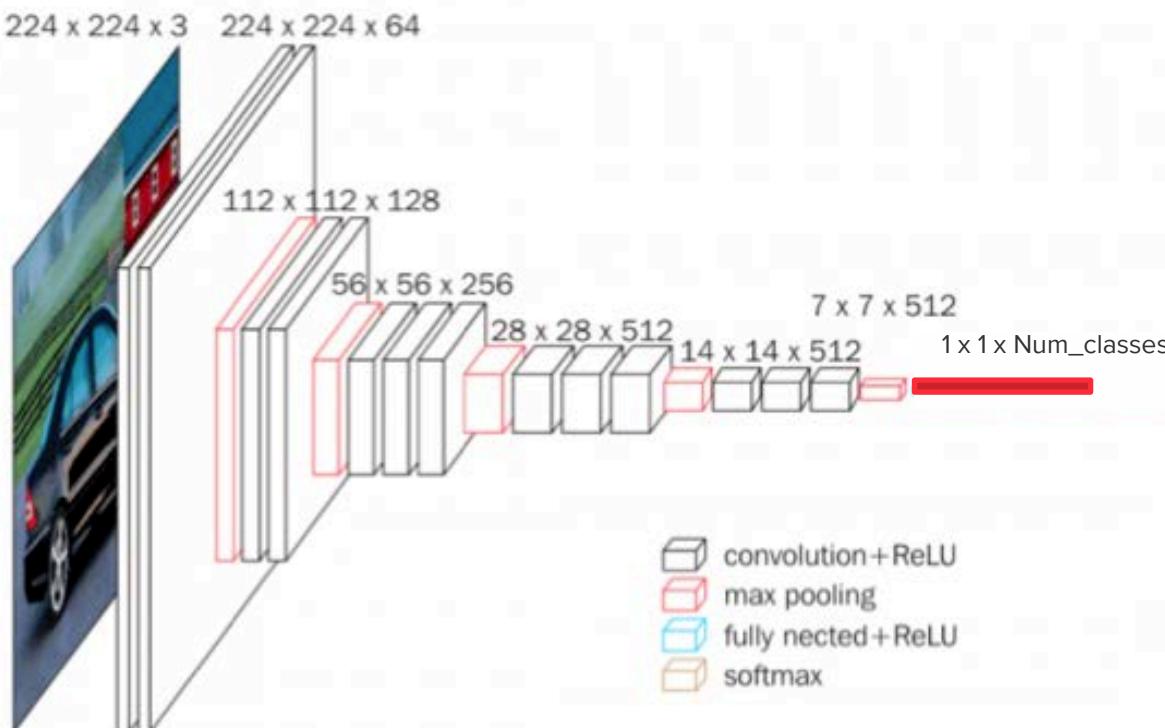
Fine-tuning

- Типичная структура CNN: сверточные слои + классификатор



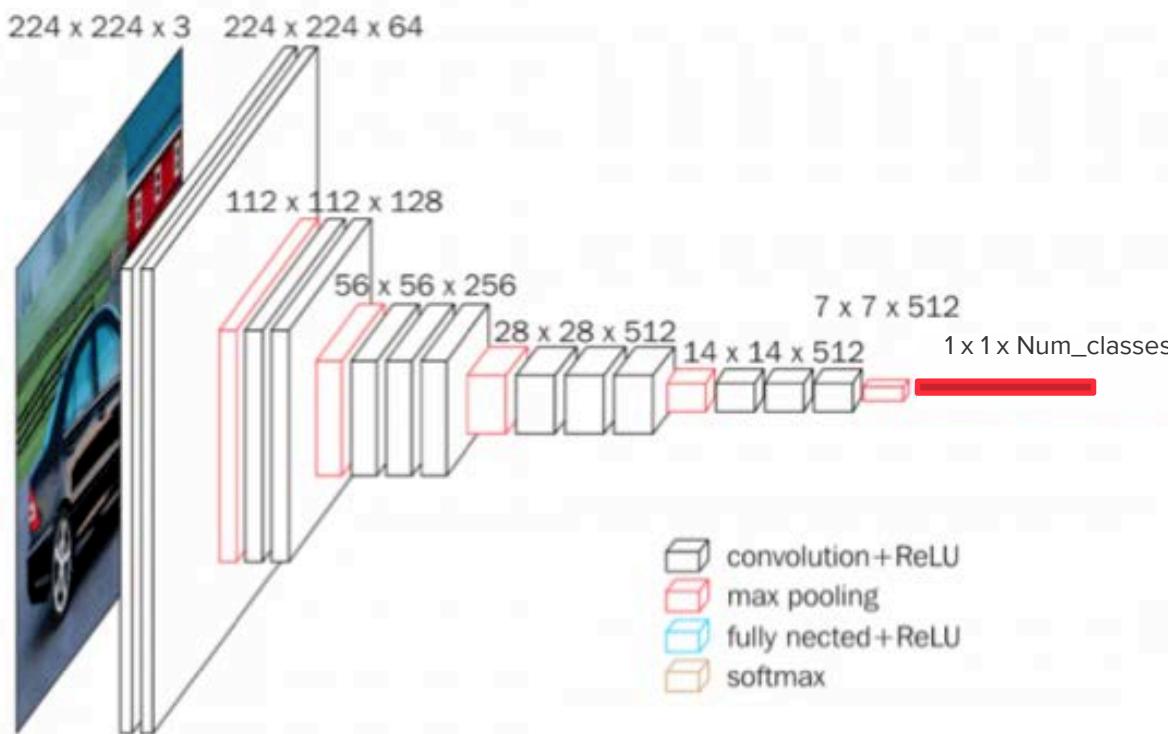
Шаг 1: добавим Global Average Pooling, заменим FC-часть (на нужное число выходов)
Шаг 2: Обучить полученную модель на своих данных

Fine-tuning



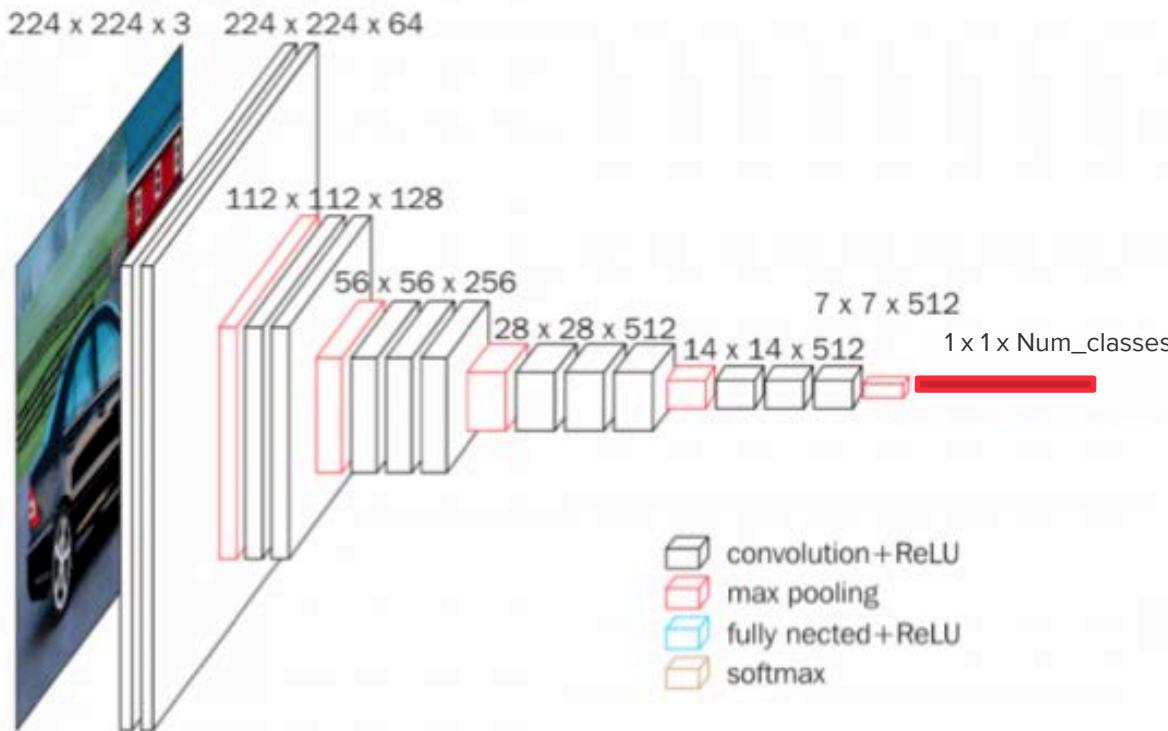
- Как обучать?
 - Только классификатор
 - ?
 - Классификатор + более ранние слои
 - ?

Fine-tuning



- Как обучать?
 - Только классификатор
 - Своих данных мало
(риск переобучения)
 - Домены “похожи”
 - Классификатор + более ранние слои
 - ?

Fine-tuning



- Как обучать?
 - Только классификатор
 - Своих данных мало
(риск переобучения)
 - Домены “похожи”
 - Классификатор + более ранние слои
 - Своих данных много
 - Домены отличаются



Fine-tuning

- Если обучаете только FC-слой, можно предварительно посчитать признаки по всему датасету
- При обучении нескольких слоев стоит попробовать первую эпоху обучать только FC-слой с маленьким LR (warmup)

Где брать модели

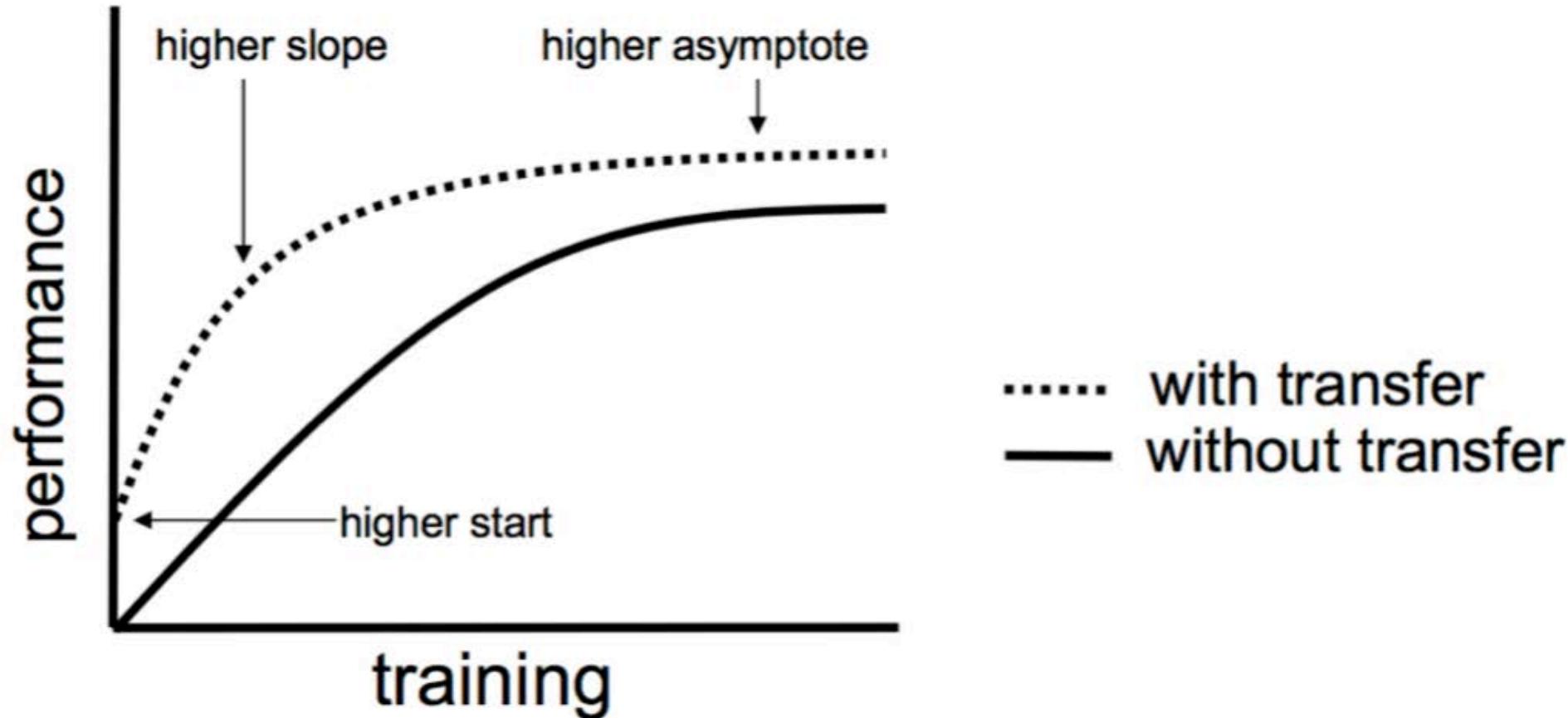
- SOTA (state-of-the-art) модели уже наверняка кем-то реализованы и обучены:
 - [torchvision.models](#)
 - [pretrainedmodels.pytorch](#)
 - [pytorch-image-models](#) (timm)
 - ...

```
from timm.models import create_model

model = create_model(
    model_name="seresnext50_32x4d",
    pretrained=True, # ImageNet-1k pretrain
    num_classes=3    # Choose your own
)
print(model)

ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (act1): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (layer1): Sequential(
```

Transfer Learning



Резюме



Резюме

- Эффективные идеи могут быть очень просты
 - Residual block, BatchNorm,
- Transfer Learning дает хорошую стартовую точку для решения задач
- Последние успехи в CV связаны с Neural Architecture Search



В следующих лекциях

- Задача Object Detection
 - Multi-stage - подходы
 - Single-shot - подходы