Архитектуры CNN (II)

Даниил Лысухин, ML Team Lead @ Ozon

План

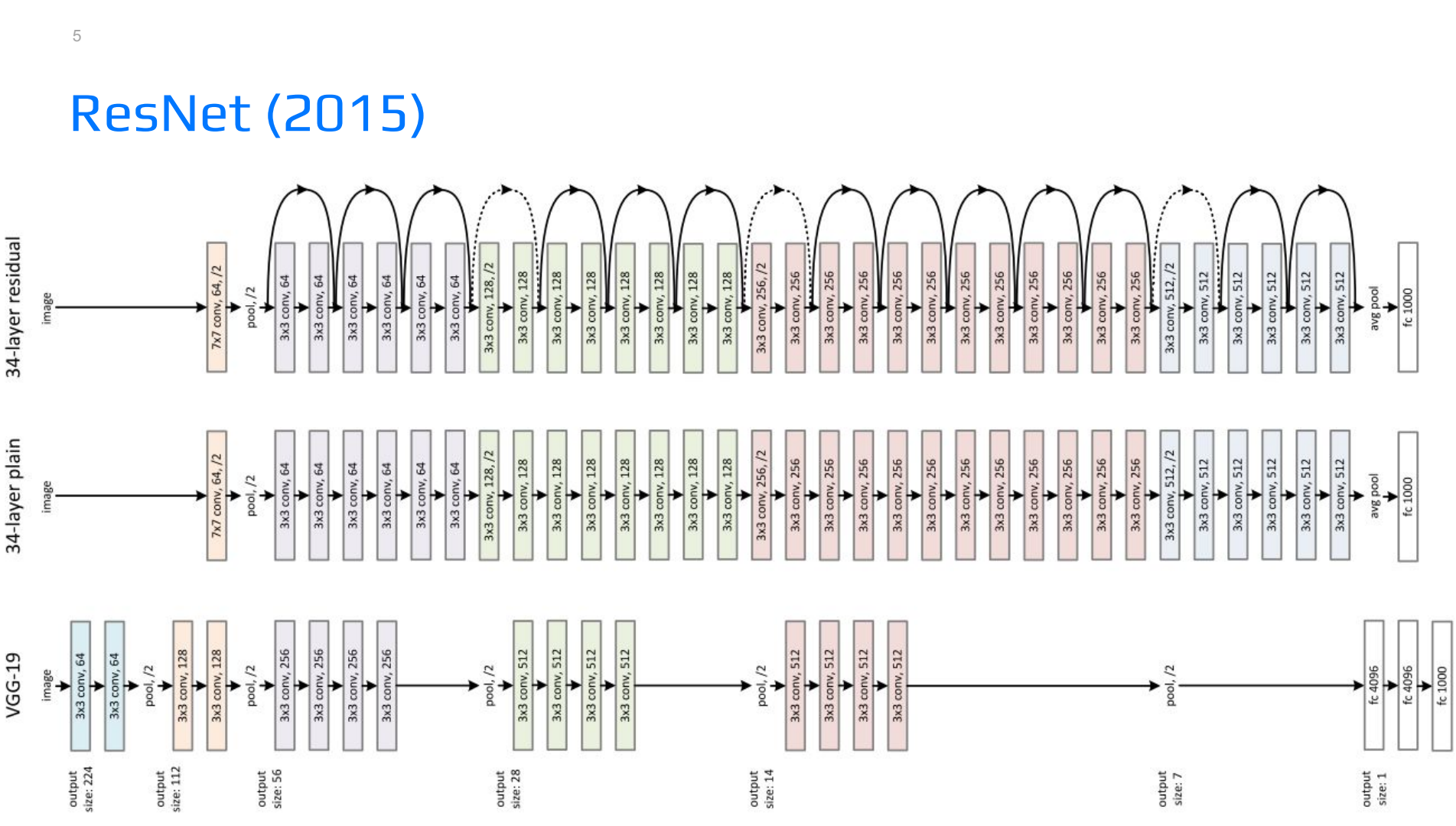
- (Recap) ResNet
- Еще немного ResNet
- NAS
- MobileNet(s)*
- Парадигма Transfer Learning
- Итоги

Recap: ResNet

ResNet (2015)

- Добавление т.н. skip-connections (они же residual-connections) позволило улучшить сходимость глубоких сетей

ResNet (2015)


VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

ResNet (2015)

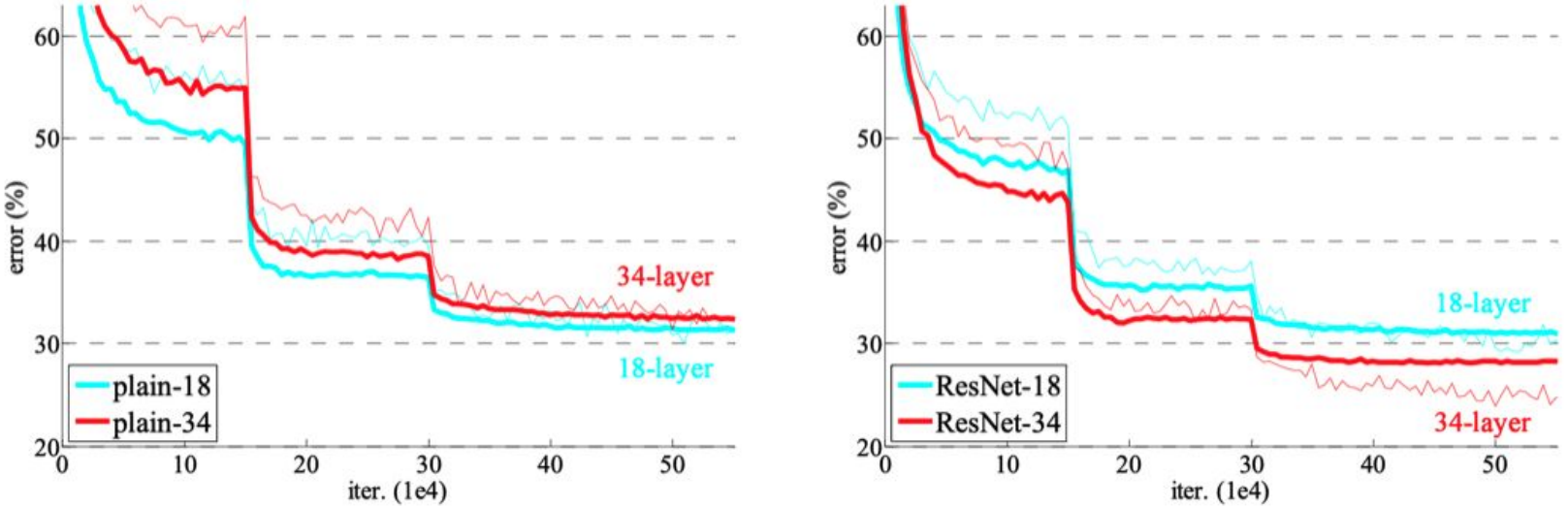


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

https://arxiv.org/pdf/1512.03385v1.pdf

ResNet (2015)

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

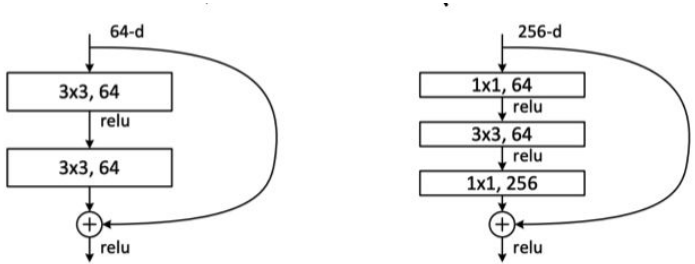


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

Ещё немного ResNet

ResNeXt (2017)

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

ResNeXt (2017)

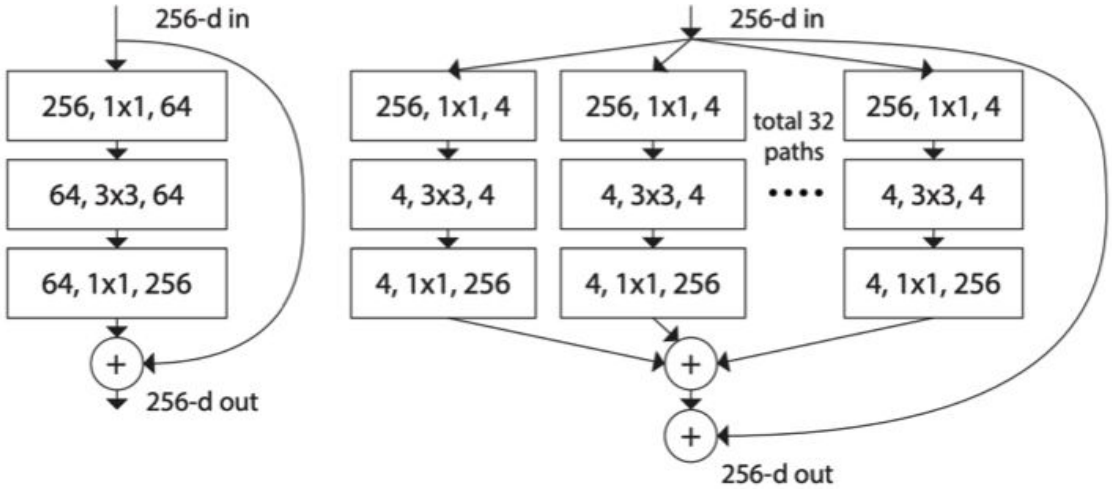


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

ResNeXt (2017)

ResNeXt-50_32x4d

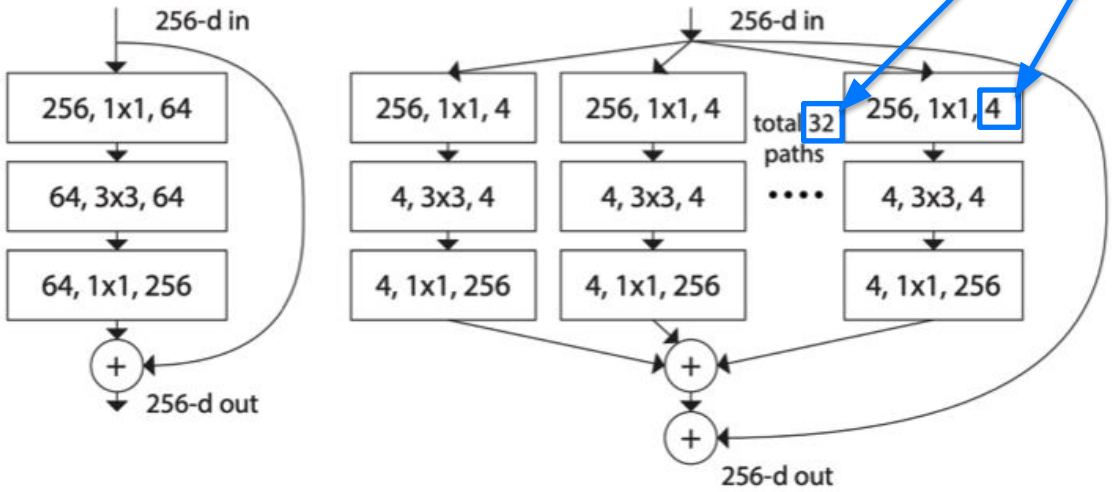
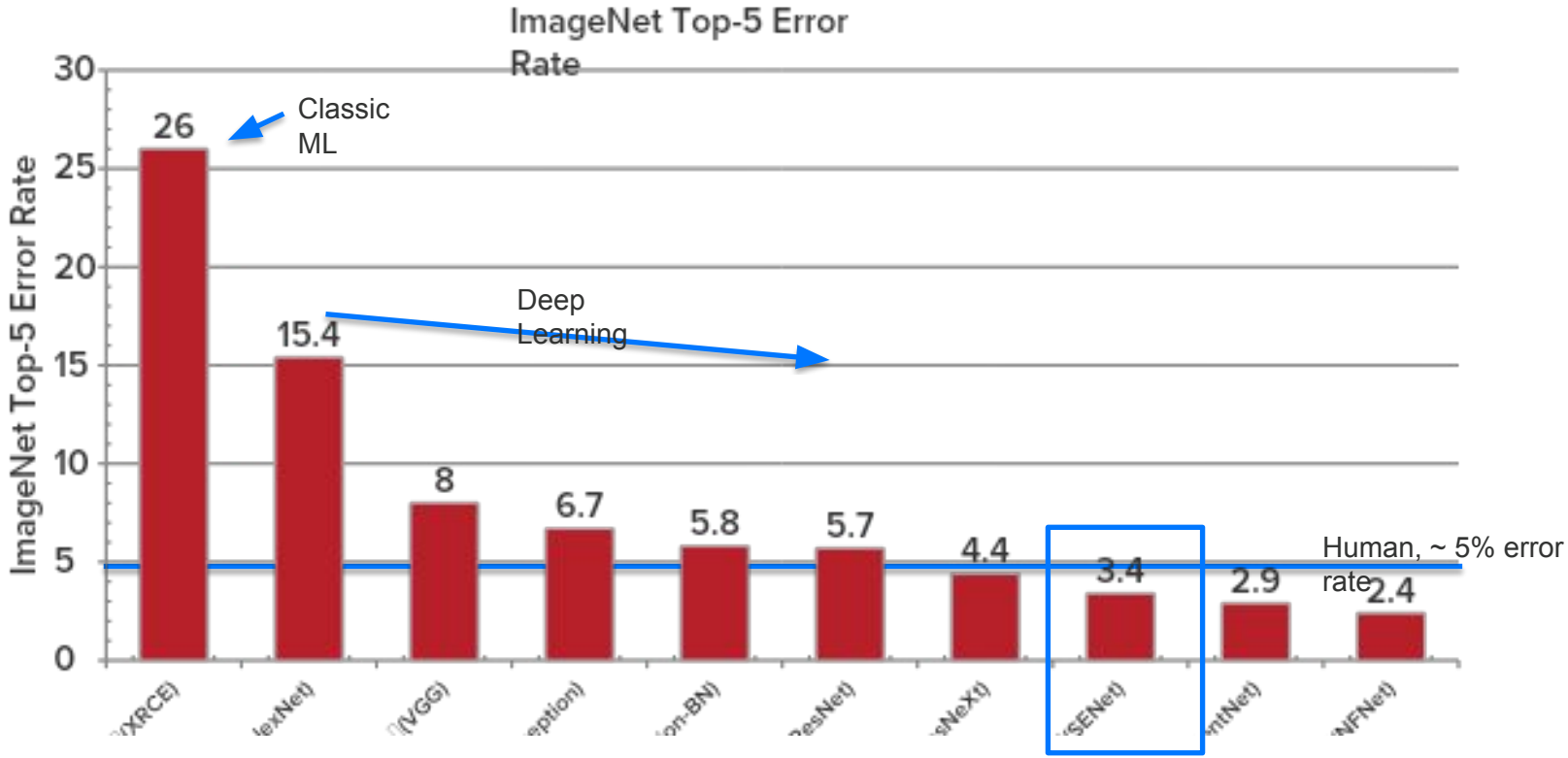


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).


ImageNet Top-5 Error Rate
ImageNet Top-5 Error Rate
30
25
20
15
10
5
0
26
15.4
8
6.7
5.8
5.7
4.4
3.4
2.9
2.4
Classic ML
Deep Learning
Human, ~ 5% error rate

Squeeze-n-Excitation (SENet) (2017)

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

Squeeze-n-Excitation (SENet) (2017)

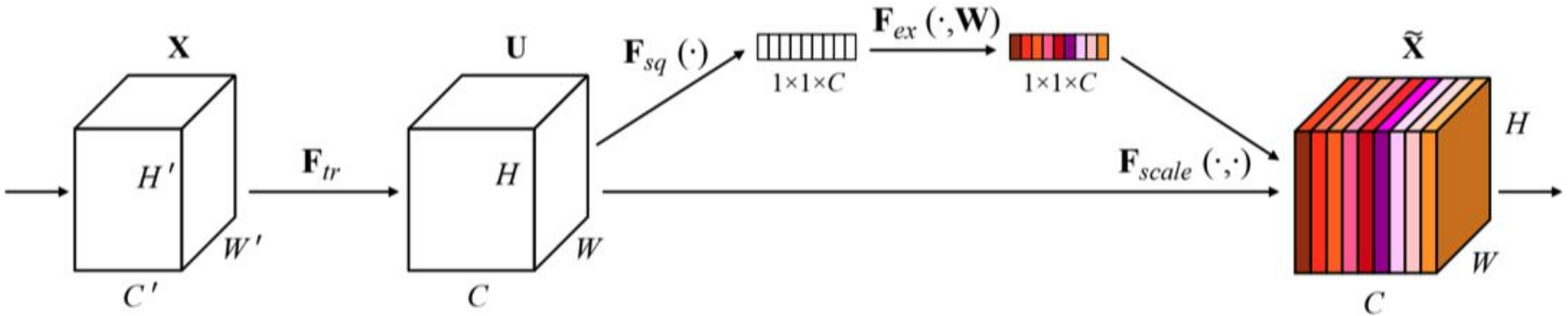
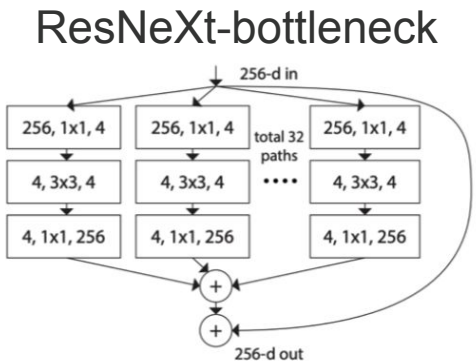


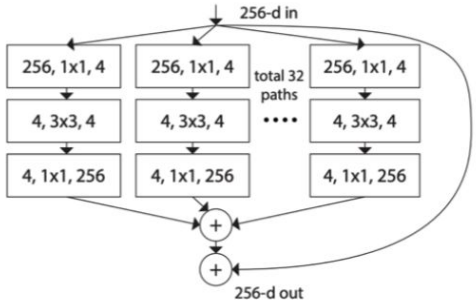
Fig. 1. A Squeeze-and-Excitation block.

ResNeXt

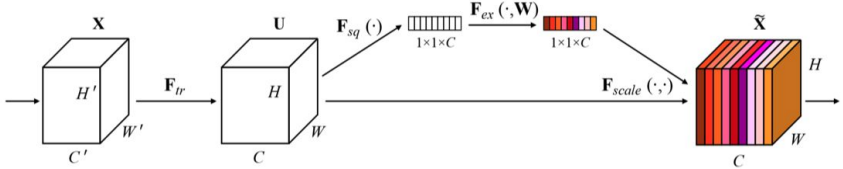

ResNeXt-bottleneck
256-d in
256, 1x1, 4
256, 1x1, 4
total 32 paths
256, 1x1, 4
4, 3x3, 4
4, 3x3, 4
4, 3x3, 4
4, 1x1, 256
4, 1x1, 256
4, 1x1, 256
+
+
256-d out

ResNeXt + SE-ResNet =

ResNeXt-bottleneck

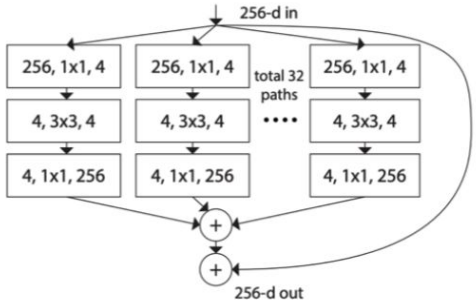


SE-block

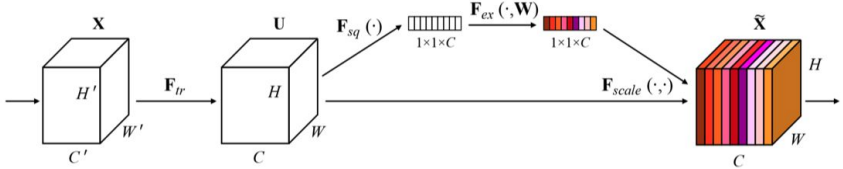


ResNeXt + SE-ResNet = SE-ResNeXt

ResNeXt-bottleneck



SE-block



https://paperswithcode.com/model/seresnext?variant=seresnext50-32x4d

Interlude

- Мы рассмотрели множество вариантов "строительных блоков" для нейросетей - Conv2d, Pooling(s), BN, Residual block / bottleneck, ...
- Есть ли какой-либо "наилучший" способ организации этих блоков в одно архитектуру?

Neural Architecture Search

Neural Architecture Search

- Группа подходов для **подбора архитектуры** нейросетей из **готовых** блоков

Neural Architecture Search

- Группа подходов для **подбора архитектуры** нейросетей из **готовых** блоков
- Подходы к NAS итеративны:
  - На каждой итерации получается некоторое распределение архитектур-кандидатов
  - "Лучшие" попадают в следующую итерацию (вероятно, с изменениями)
- Для **очень** любознательных: пост

Neural Architecture Search

- Основные компоненты любой разновидности NAS:

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации
  - **Алгоритм поиска** (Search algorithm): способ отбора лучших архитектур из множества кандидатов

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации
  - **Алгоритм поиска** (Search algorithm): способ отбора лучших архитектур из множества кандидатов
  - **Стратегия оценки** (Evaluation strategy): способ оценки качества архитектур-кандидатов

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации
  - **Алгоритм поиска** (Search algorithm): способ отбора лучших архитектур из множества кандидатов
  - **Стратегия оценки** (Evaluation strategy): способ оценки качества архитектур-кандидатов

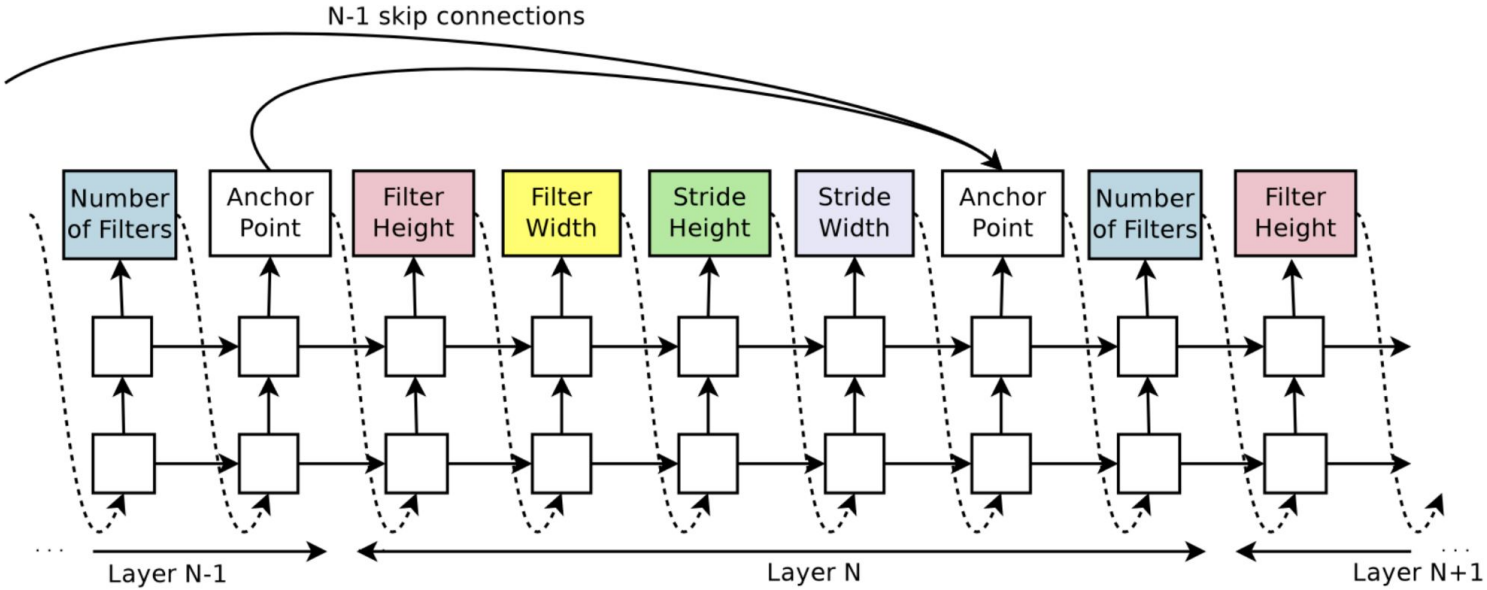
NAS: Search space

- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?

NAS: Search space

- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?
  - Как последовательность операций: новые операции добавляются "по одной"

NAS: Search space



https://arxiv.org/abs/1611.01578

NAS: Search space

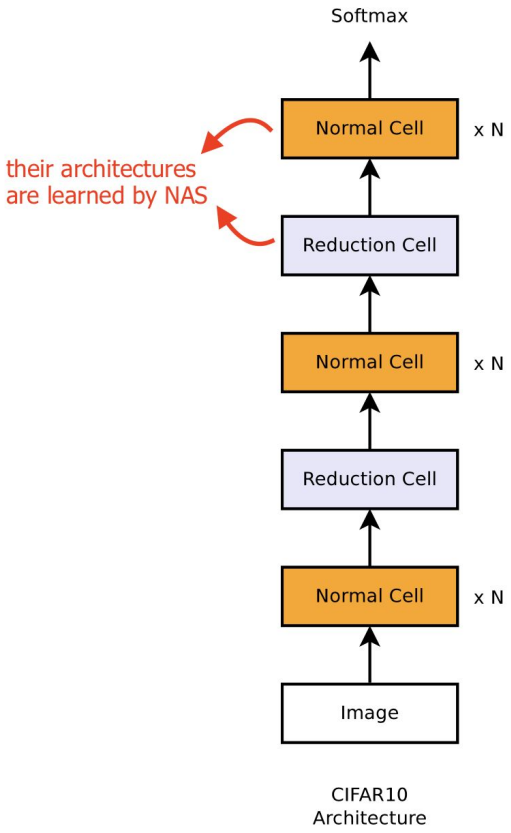
- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?
  - Как последовательность операций: новые операции добавляются "по одной"

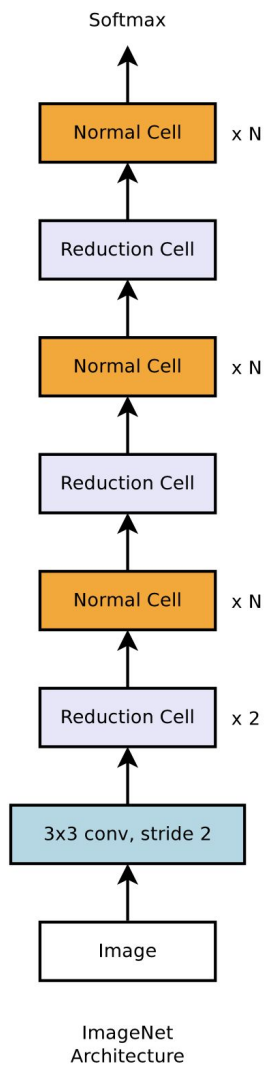
NAS: Search space

- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?
  - Как последовательность операций: новые операции добавляются "по одной"
  - Как последовательность одинаковых блоков: подбирается один блок

NAS: Search space

https://arxiv.org/abs/1707.07012





NAS: Search space

- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?
  - Как последовательность операций: новые операции добавляются "по одной"
  - Как последовательность одинаковых блоков: подбирается один блок

NAS: Search space

- Базовые операции (~= слои) примерно известны
- В каком виде собирать из них архитектуру?
  - Как последовательность операций: новые операции добавляются "по одной"
  - Как последовательность одинаковых блоков: подбирается один блок
  - ...

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации
  - **Алгоритм поиска** (Search algorithm): способ отбора лучших архитектур из множества кандидатов
  - **Стратегия оценки** (Evaluation strategy): способ оценки качества архитектур-кандидатов

NAS: Search algorithm

- Случайный поиск: случайное сэмплирование архитектур из пространства поиска

NAS: Search algorithm

- Случайный поиск: случайное сэмплирование архитектур из пространства поиска
- Reinforcement Learning:
  - action ~ обновление архитектуры
  - reward ~ качество текущей архитектуры (после обучения и валидации)
  - Пример: NASNet

NAS: Search algorithm

- Случайный поиск: случайное сэмплирование архитектур из пространства поиска
- Reinforcement Learning:
  - action ~ обновление архитектуры
  - reward ~ качество текущей архитектуры (после обучения и валидации)
  - Пример: NASNet
- Эволюционные алгоритмы:
  - обновления = случайные "мутации"
  - Пример: AmoebaNet

Neural Architecture Search

- Основные компоненты любой разновидности NAS:
  - **Пространство поиска** (Search space): множество базовых операций, которые могут быть включены в архитектуру (свертка, skip-connection, ...), а также способы их организации
  - **Алгоритм поиска** (Search algorithm): способ отбора лучших архитектур из множества кандидатов
  - **Стратегия оценки** (Evaluation strategy): способ оценки качества архитектур-кандидатов

NAS: Evaluation strategy

- Обучать все архитектуры с нуля "до упора" или

NAS: Evaluation strategy

- Обучать все архитектуры с нуля "до упора" или
  - Обучать с переиспользованием весов
  - Обучать меньше эпох
  - Обучать уменьшенную версию
  - ...

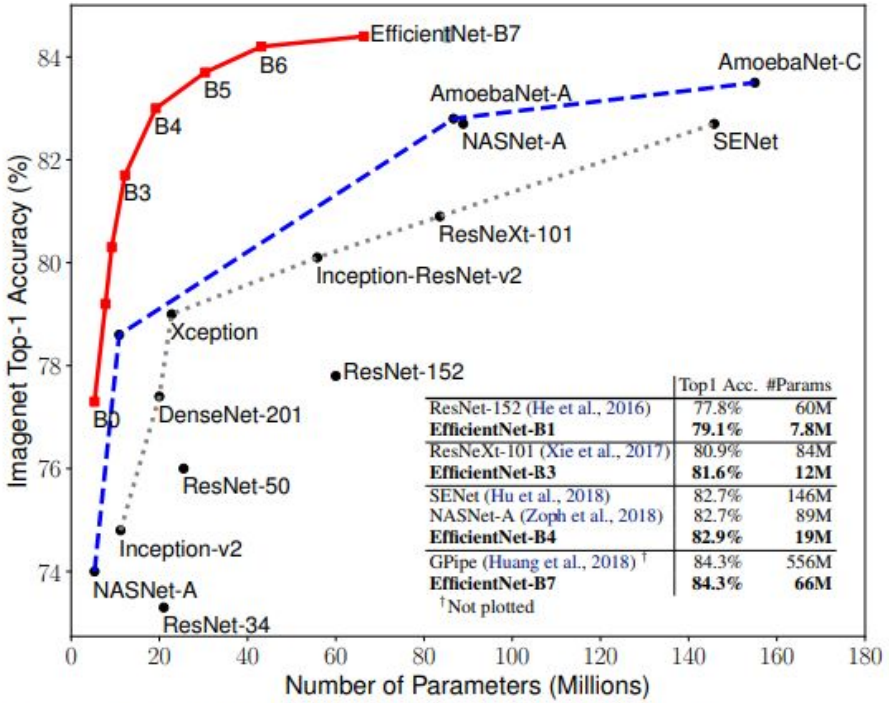
NAS

- (NASNet) Learning Transferable Architectures for Scalable Image Recognition
- (AmoebaNet) Regularized Evolution for Image Classifier Architecture Search

NAS: это дорого

- NASNet: 800 GPU x 28 days
- AmoebaNet: 450 GPU x 7 days
- ...

NAS



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

† Not plotted

Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

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 ("разрешение")
  - Число фильтров в сверточных слоях (“ширина”)
  - Число сверточных слоев (“глубина”)

EfficientNet (2019)

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

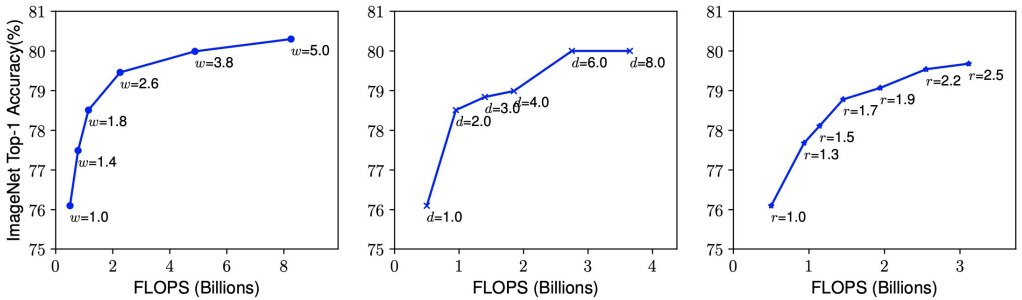


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

EfficientNet (2019)

- Пусть некая архитектура CNN имеет вычислительную сложность (FLOPS) ***X***
- Как она изменится, если:

EfficientNet (2019)

- Пусть некая архитектура CNN имеет вычислительную сложность (FLOPS) X
- Как она изменится, если:
  - Увеличить глубину всей сети в k раз?

EfficientNet (2019)

- Пусть некая архитектура CNN имеет вычислительную сложность (FLOPS) ***X***
- Как она изменится, если:
  - Увеличить глубину всей сети в ***k*** раз? ***~kX***
  - Увеличить ширину каждого conv-слоя в ***k*** раз?

EfficientNet (2019)

- Пусть некая архитектура CNN имеет вычислительную сложность (FLOPS) ***X***
- Как она изменится, если:
  - Увеличить глубину всей сети в ***k*** раз? ***~kX***
  - Увеличить ширину каждого conv-слоя в ***k*** раз? ***~kkX***
  - Увеличить размер входного изображения вдвое в ***k*** раз?

EfficientNet (2019)

- Пусть некая архитектура CNN имеет вычислительную сложность (FLOPS) ***X***
- Как она изменится, если:
  - Увеличить глубину всей сети в ***k*** раз? ***~kX***
  - Увеличить ширину каждого conv-слоя в ***k*** раз? ***~kkX***
  - Увеличить размер входного изображения вдвое в ***k*** раз? ***~kkX***

EfficientNet (2019)

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:

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

- Предлагается масштабировать глубину, ширину и разрешение так, чтобы вычислительная сложность менялась по заданному закону
  - ~ 2^***ϕ***
- **Compound scaling**

EfficientNet (2019)

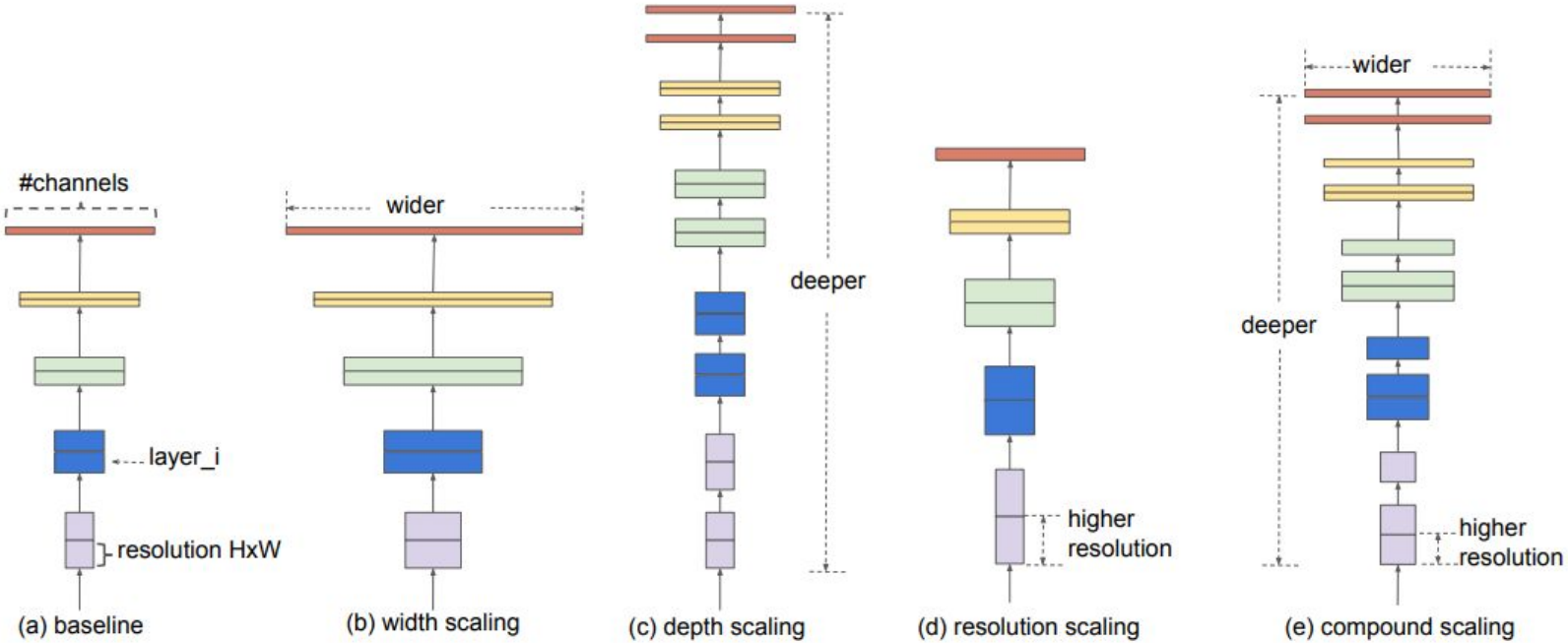


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

EfficientNet (2019)

- В статье показано, что усложнение архитектуры по отдельным размерностям не так эффективно повышает качество модели, как при **compound scaling**

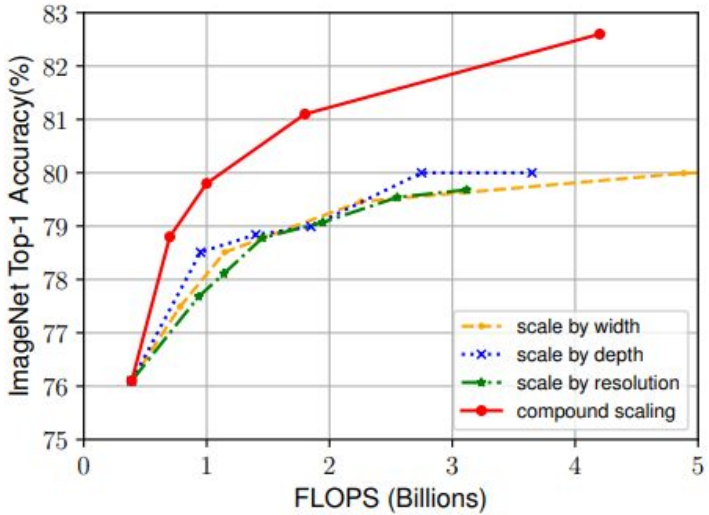


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

EfficientNet (2019)

- Также авторы предложили свое семейство архитектур возрастающей сложности
  - EfficientNet-B0
  - ...
  - EfficientNet-B7
- Базовый блок: MBConv + SE
- Не обошлось без NAS

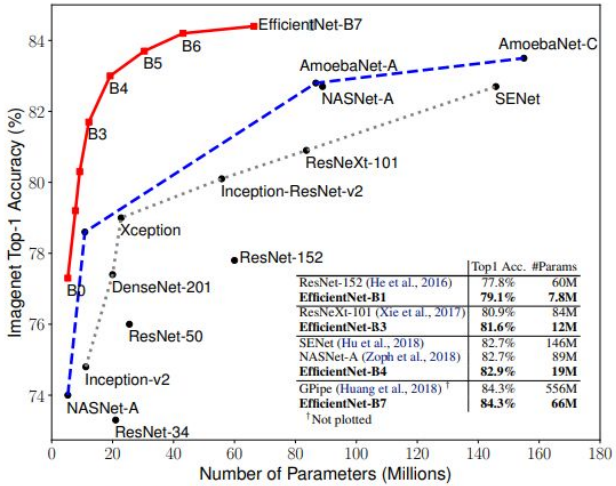


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

MORE

- (NFNet) High-Performance Large-Scale Image Recognition Without Normalization
  - Избавились от BatchNorm, сэкономив ресурсы
  - NAS
- (RegNet) Regularized Evolution for Image Classifier Architecture Search
  - Оригинальный подход к процессу подбора архитектуры

Interlude

- Рассмотренные подходы улучшали результаты за счет усложнения архитектур (вширь и вглубь)
- Чем сложнее архитектура, тем дольше ее вычислять
- Для применимости в реальном мире требуется не только высокое качество, но и приемлемая скорость работы

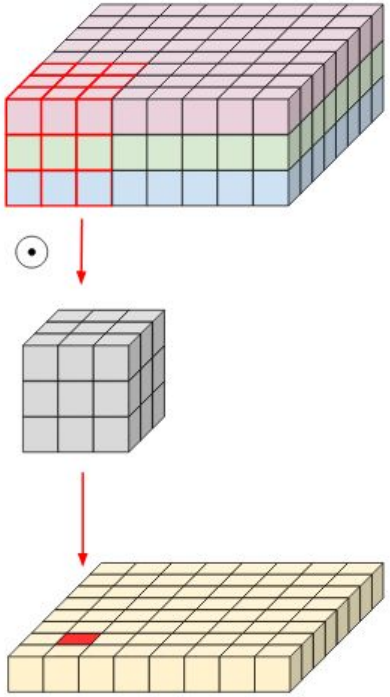
MobileNet(s)

MobileNet(s) (2017 - ...)

- Династия легких моделей, предназначенных для работы на мобильных устройствах
- V1 (2017): MobileNets: Efficient CNNs for Mobile Vision Applications
  - Разделение вычислений по HW и по D (**depthwise-separable convolutions**)
- V2 (2018): MobileNetV2: Inverted Residuals and Linear Bottlenecks
- V3 (2019): Searching for MobileNetV3

Simple Convolution

- В обычном сверточном слое 1 фильтр выполняет одновременно
  - свертку по ширине/высоте
  - смешивание различных каналов

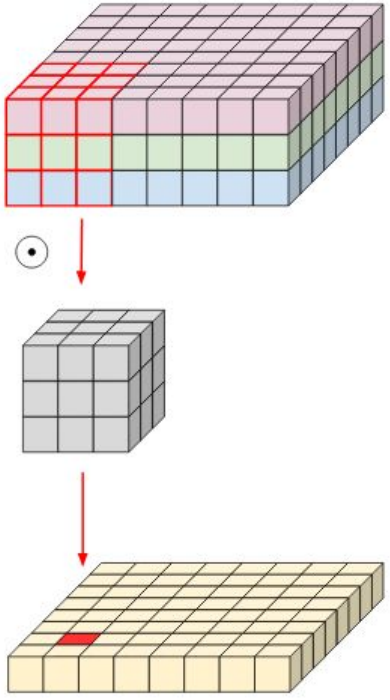


https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Simple Convolution

- В обычном сверточном слое 1 фильтр выполняет одновременно
  - свертку по ширине/высоте
  - смешивание различных каналов

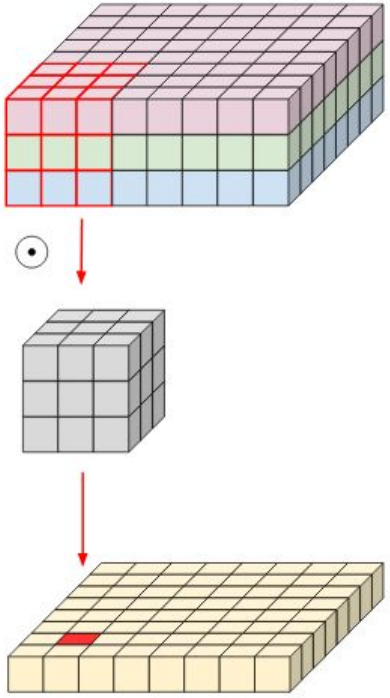
Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?



https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Simple Convolution

- В обычном сверточном слое 1 фильтр выполняет одновременно
  - свертку по ширине/высоте
  - смешивание различных каналов



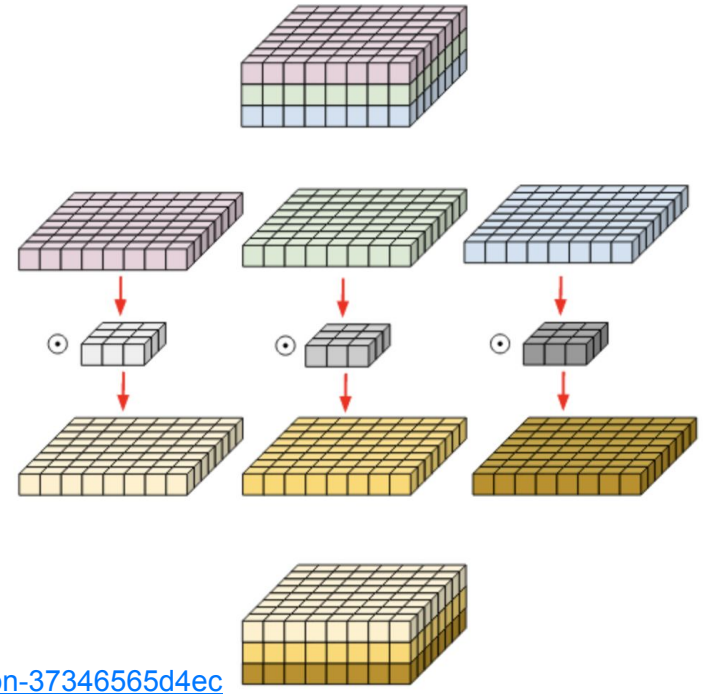
Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x D) x (H x W) x N
При K = 3, H = W = 64, D = 256, N = 256: 2.4E09

https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise Convolution

- Позволим 1 фильтру видеть только 1 канал входного тензора

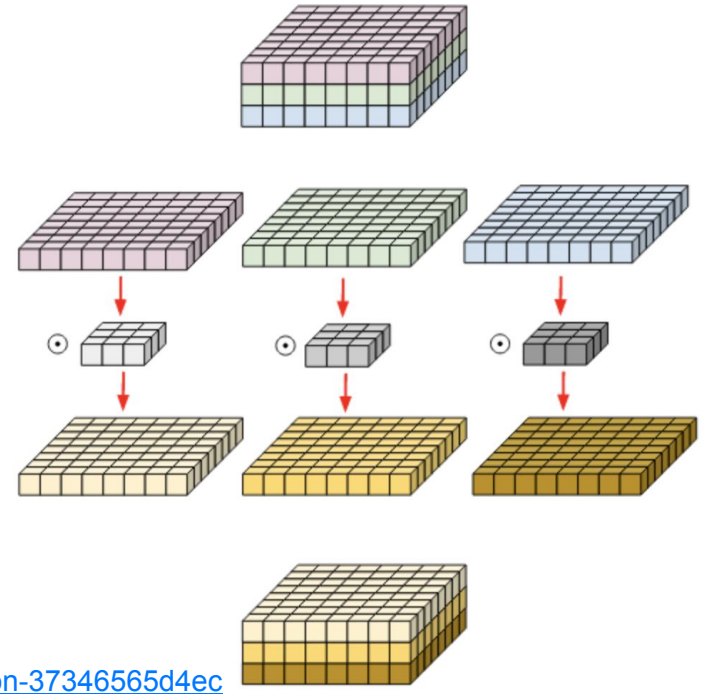


https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise Convolution

- Позволим 1 фильтру видеть только 1 канал входного тензора

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?



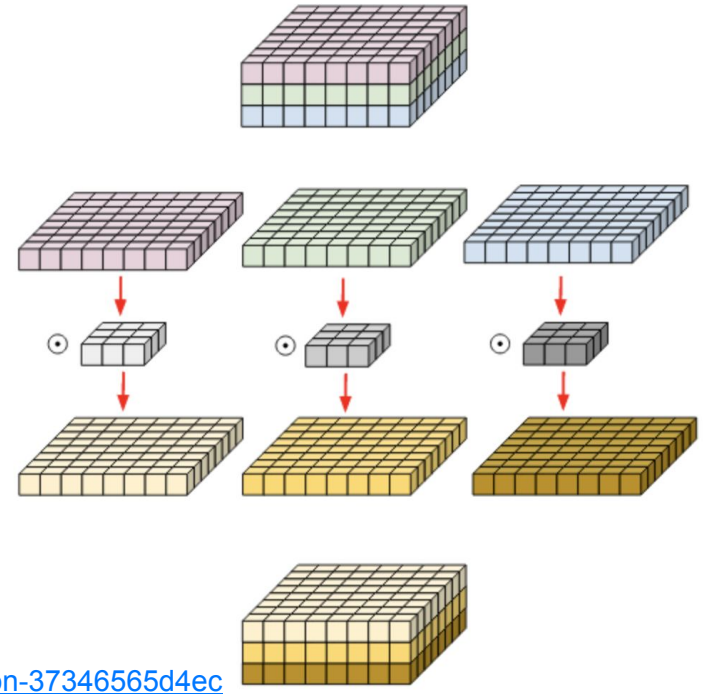
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise Convolution

- Позволим 1 фильтру видеть только 1 канал входного тензора

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N ~ 9.5E06



https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

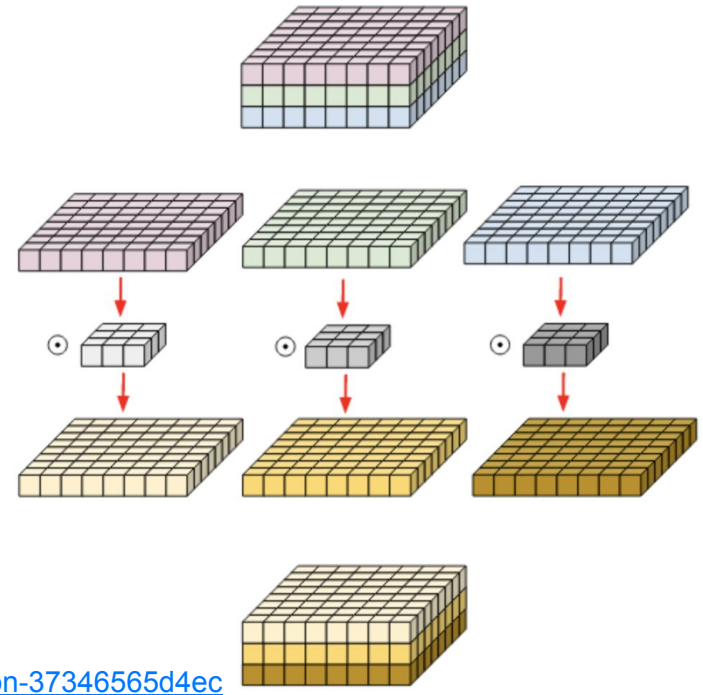
Depthwise Convolution

- Позволим 1 фильтру видеть только 1 канал входного тензора

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N ~ 9.5E06

- Но каналы теперь независимы!



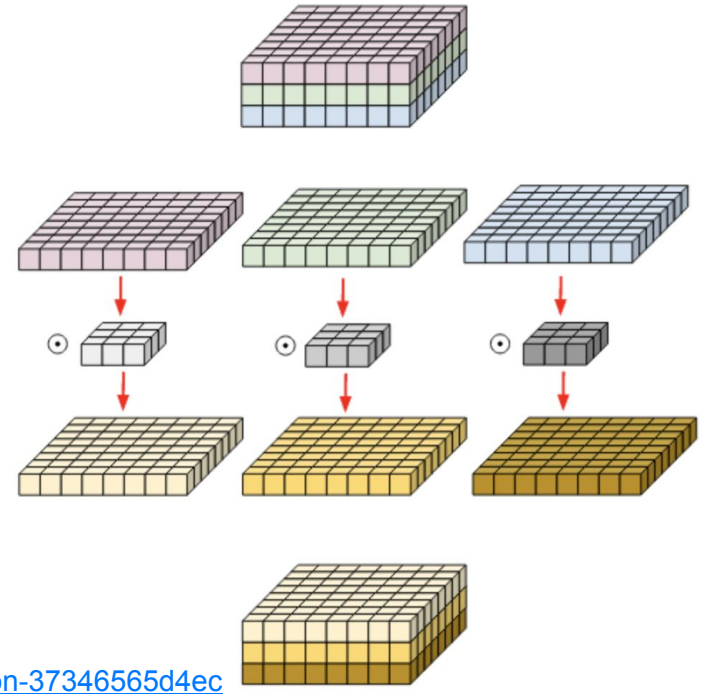
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise Convolution

- Позволим 1 фильтру видеть только 1 канал входного тензора

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N ~ 9.5E06

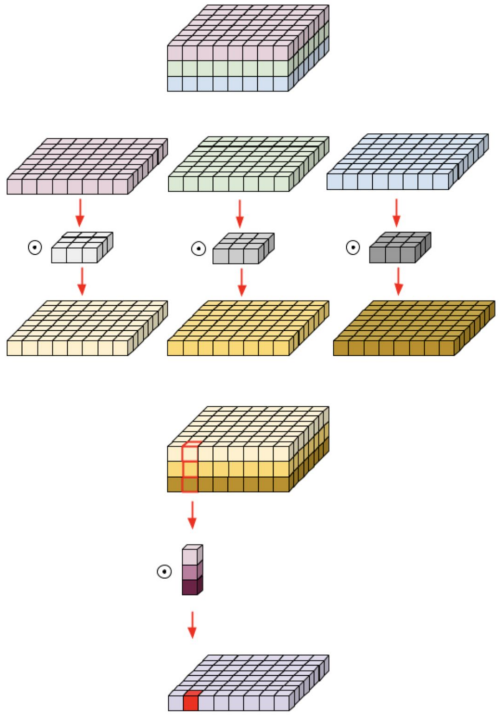


- Но каналы теперь независимы!
  - Смешаем их сверткой 1x1
    - "Pointwise" Convolution

https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)



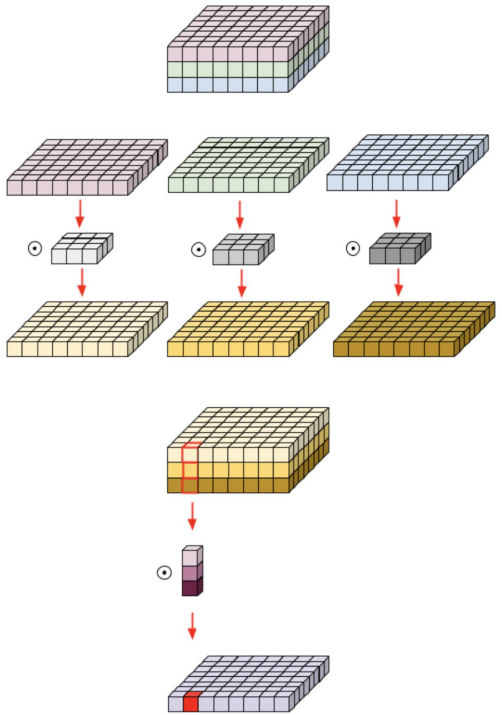
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N (depthwise)



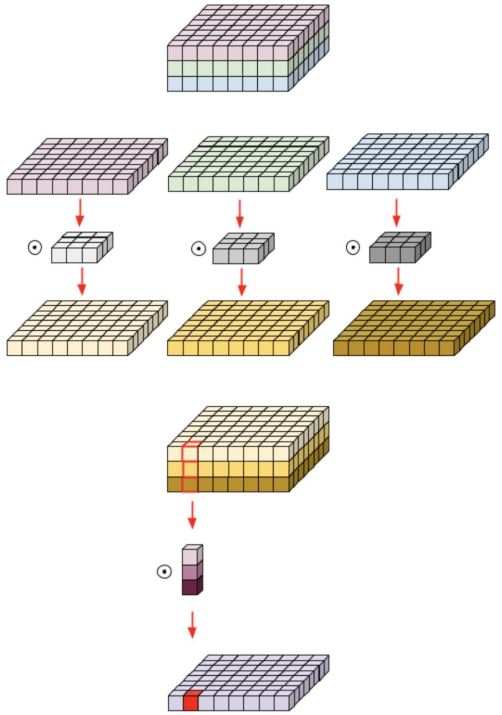
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N (depthwise)
(1 x 1 x D) x (H x W) x N (pointwise)



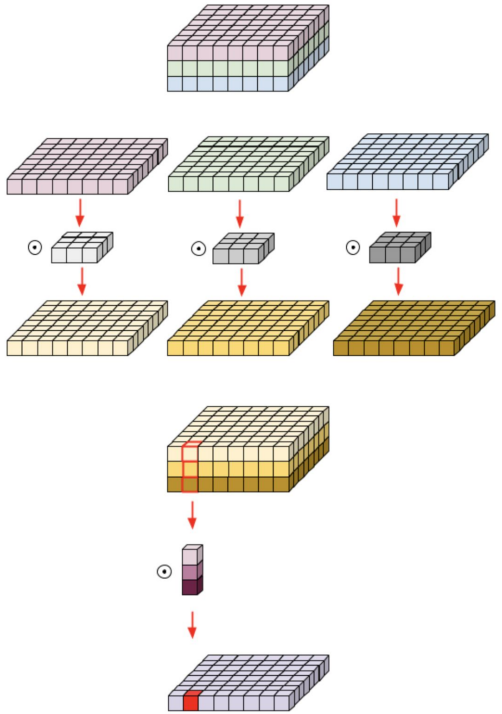
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N (depthwise)
(1 x 1 x D) x (H x W) x N (pointwise)
(K x K + D) x (H x W) x N (total)



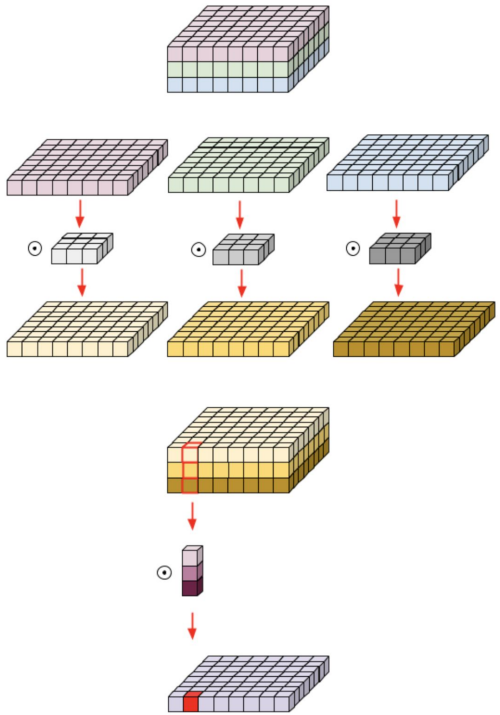
https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N (depthwise)
(1 x 1 x D) x (H x W) x N (pointwise)
(K x K + D) x (H x W) x N (total)
При K = 3, H = W = 64, D = 256, N = 256: **2.8E08**



https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

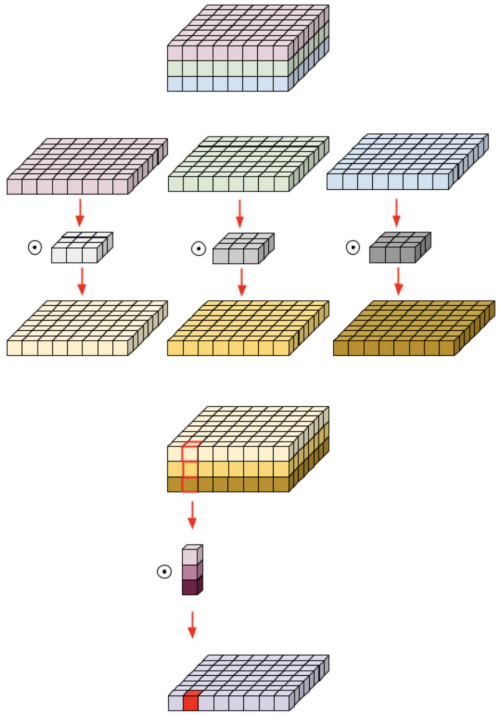
Depthwise-separable Convolution

- Сначала свертка depthwise (KxK)
- Затем свертка pointwise (1x1)

Вход размера HxWxD, ядро KxK, всего N фильтров
Сколько операций требуется для вычисления?

(K x K x **1**) x (H x W) x N (depthwise)
(1 x 1 x D) x (H x W) x N (pointwise)
(K x K + D) x (H x W) x N (total)
При K = 3, H = W = 64, D = 256, N = 256: **2.8E08**

У обычной свертки было **2.4E09**



https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4e

Transfer Learning

Supervised Learning

- Привычная формулировка проблемы ML:

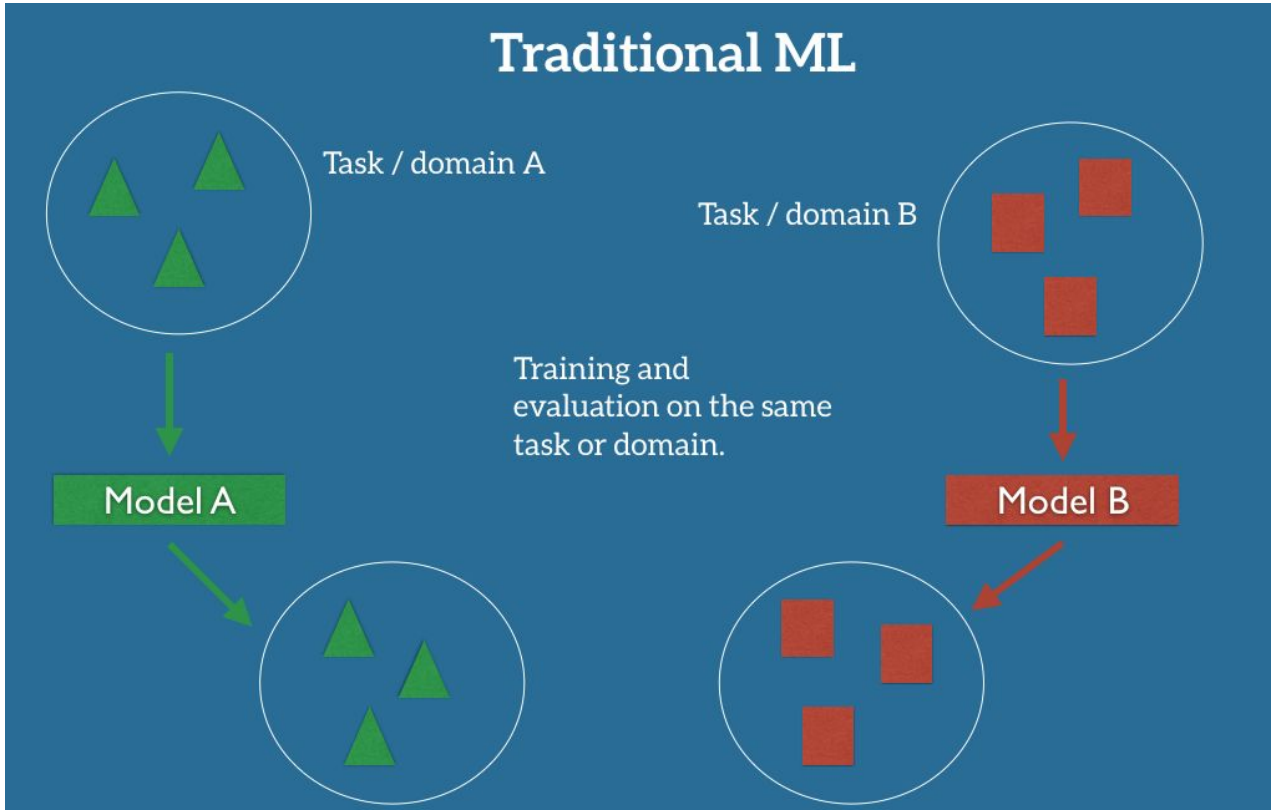
Supervised Learning

- Привычная формулировка проблемы ML:
  - Есть данные некоторой природы (**domain**)
    - Изображения: фотографии, рисунки, спутниковые снимки, ...
    - Тексты: литература, новости, комментарии, отзывы, ...
    - ...

Supervised Learning

- Привычная формулировка проблемы ML:
  - Есть данные некоторой природы (**domain**)
    - Изображения: фотографии, рисунки, спутниковые снимки, ...
    - Тексты: литература, новости, комментарии, отзывы, ...
    - ...
  - Есть задача, которую надо решить (**task**)
    - Классификация на заданное число классов
    - Генерация
    - Регрессия / детектирование (для CV-задач)
    - ...

Supervised Learning



https://ruder.io/transfer-learning/index.html

Supervised Learning

- Это работает, когда данных для решения конкретной **задачи** на целевом **домене** *достаточно*

Supervised Learning

- Это работает, когда данных для решения конкретной **задачи** на целевом **домене** *достаточно*
- Чаще всего, их *недостаточно*

Supervised Learning OR ...

- Но зато бывает, что есть много:

Supervised Learning OR ...

- Но зато бывает, что есть много:
  - Неразмеченных данных

Supervised Learning OR ...

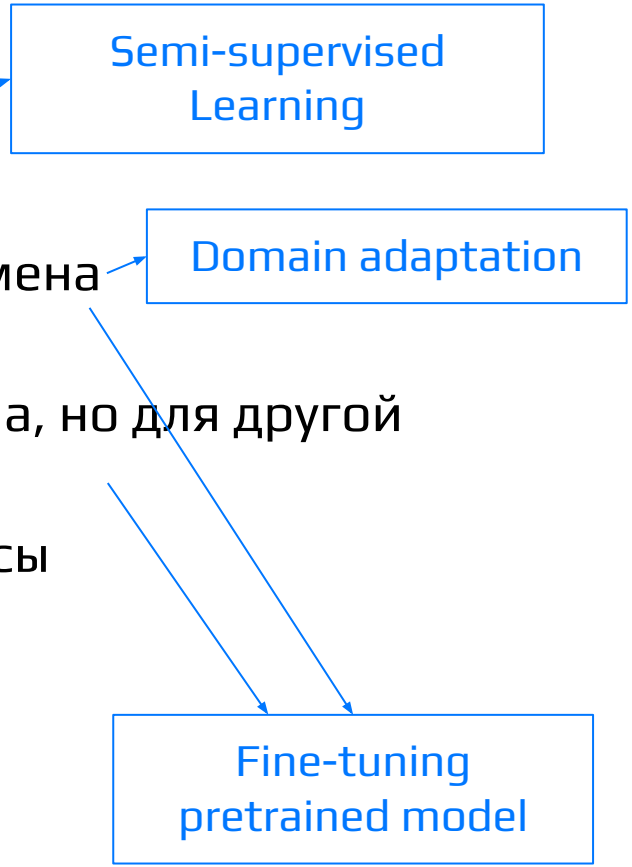
- Но зато бывает, что есть много:
  - Неразмеченных данных
  - Размеченных данных, но из другого домена
    - Например, из симулятора

Supervised Learning OR ...

- Но зато бывает, что есть много:
  - Неразмеченных данных
  - Размеченных данных, но из другого домена
    - Например, из симулятора
  - Размеченных данных из нужного домена, но для другой задачи
    - Например, вместо регрессии - классы

Supervised Learning OR ...

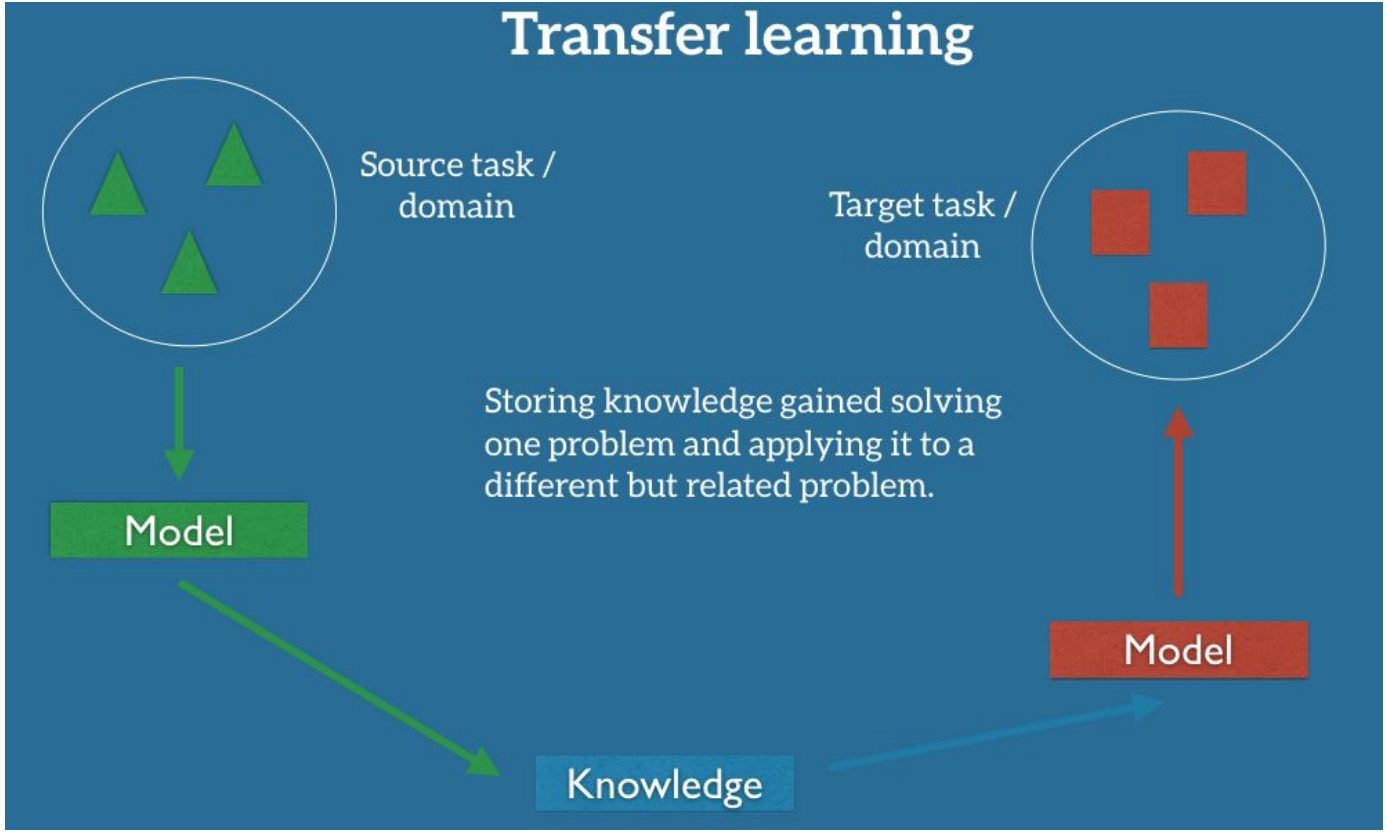
- Но зато бывает, что есть много:
  - Неразмеченных данных
  - Размеченных данных, но из другого домена
    - Например, из симулятора
  - Размеченных данных из нужного домена, но для другой задачи
    - Например, вместо регрессии - классы



Transfer Learning

- **Transfer Learning** - это парадигма "передачи знания" от одной модели к другой

Transfer Learning



https://ruder.io/transfer-learning/index.html

Transfer Learning

- **Transfer Learning** - это парадигма "передачи знания" от одной модели к другой

Transfer Learning

- **Transfer Learning** - это парадигма "передачи знания" от одной модели к другой
  - Самый простой способ - это дообучение уже "готовой" модели (**fine-tuning**)
    - ... предобученной на другом датасете (domain)
    - ... и/или для другой задачи (task)

Fine-tuning - пример

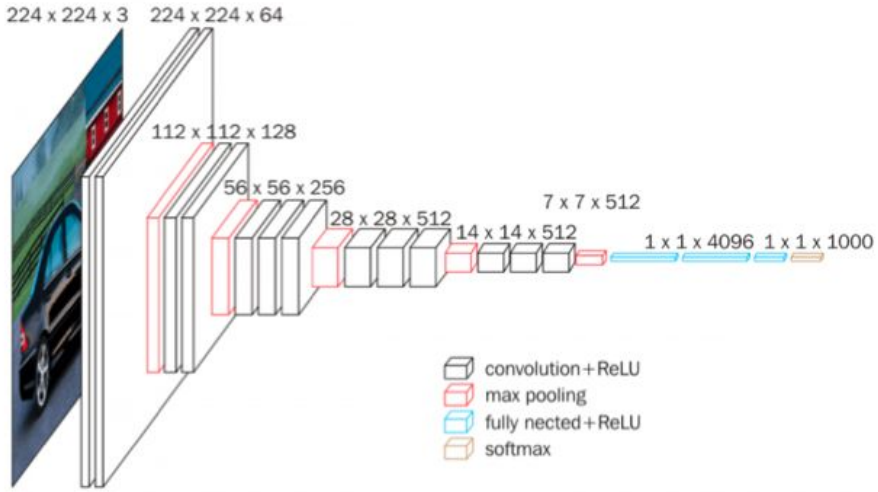
- Типичная CNN =
  - сверточные слои (извлекают признаки) +
  - решающая "голова" в конце (классификатор, регрессор, ...)
- Слои для извлечения признаков можно переиспользовать
  - Чем ближе к началу сети, тем признаки "примитивнее"
  - Чем дальше, тем признаки "абстрактнее"

Fine-tuning - пример

- Пусть имеется CNN, обученная на датасете ImageNet
- Хотим дообучить ее на своих данных для своей задачи

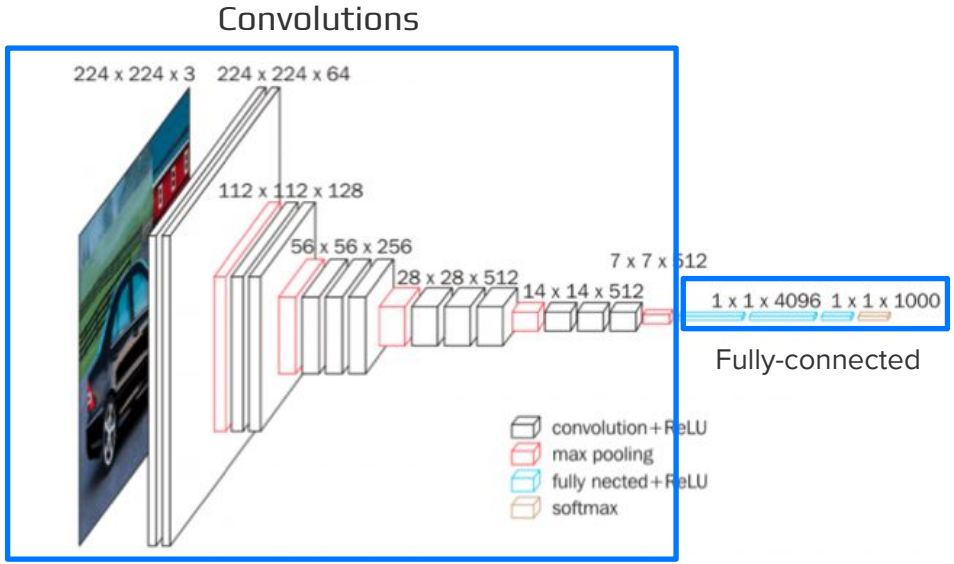
Fine-tuning - пример

- Пусть имеется CNN, обученная на датасете ImageNet
- Хотим дообучить ее на своих данных для своей задачи



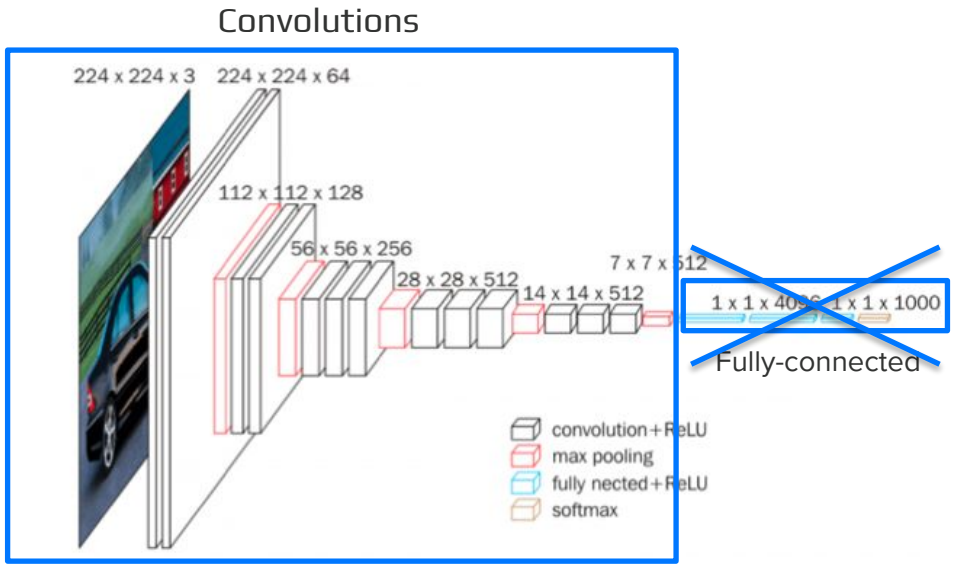
Fine-tuning - пример

- Пусть имеется CNN, обученная на датасете ImageNet
- Хотим дообучить ее на своих данных для своей задачи



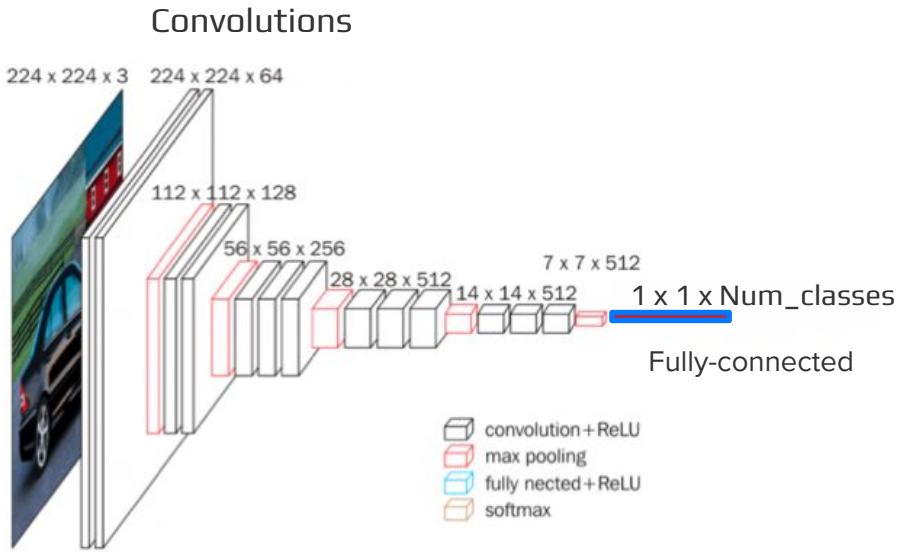
Fine-tuning - пример

- Удалим FC слои



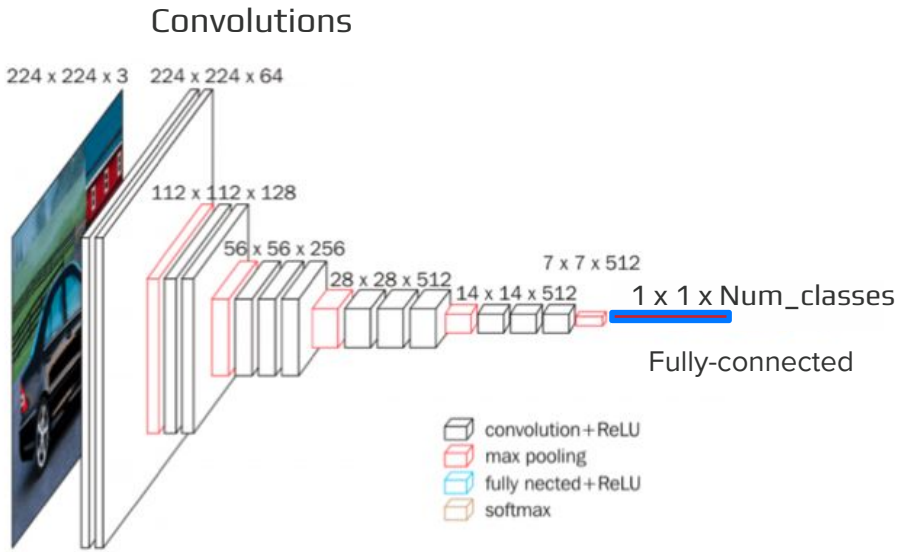
Fine-tuning - пример

- Добавим Global Average Pooling (если не было)
- Добавим новый FC-слой с нужным количеством выходов



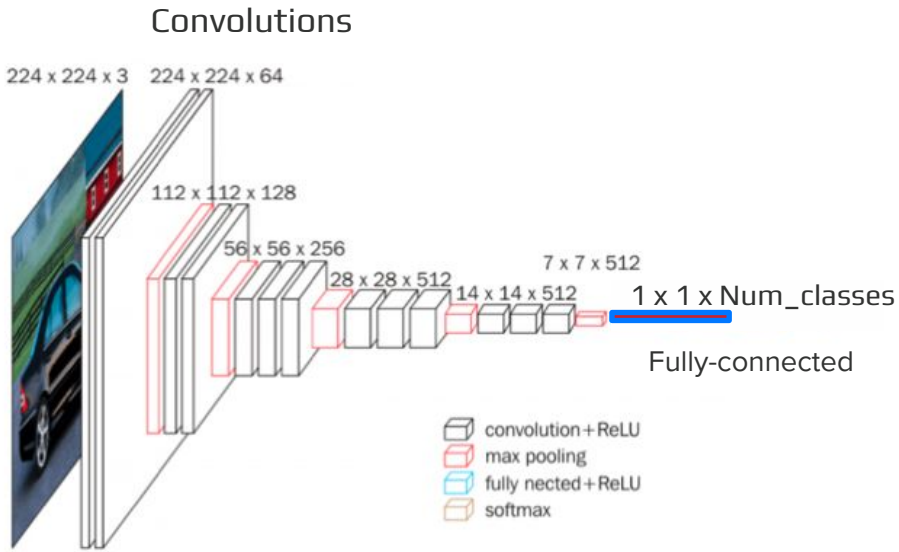
Fine-tuning - пример

- Можно дообучать на своих данных



Fine-tuning - пример

- Можно дообучать на своих данных
  - Какие слои?



Fine-tuning - пример

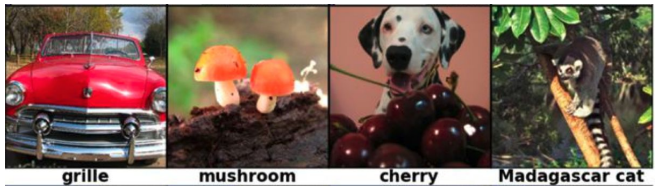
- Если исходный и целевой домены "похожи", можно заморозить сверточные слои и учить только FC

Fine-tuning - пример

- Если исходный и целевой домены "похожи", можно заморозить сверточные слои и учить только FC

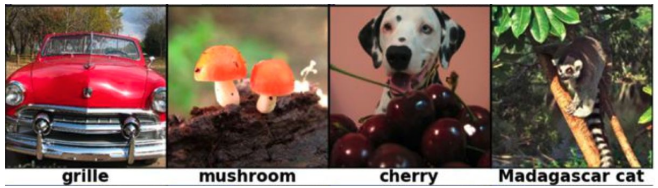
Fine-tuning - пример

- Если исходный и целевой домены "похожи", можно заморозить сверточные слои и **учить только FC**



Fine-tuning - пример

- Если исходный и целевой домены "похожи", можно заморозить сверточные слои и **учить только FC**



Fine-tuning - пример

- Если исходный и целевой домены различаются, можно учить больше последних слоев / всю сеть целиком

Fine-tuning - пример

- Если исходный и целевой домены различаются, можно учить больше последних слоев / всю сеть целиком
  - В этом случае предобученные веса фактически используются для более качественной инициализации

Где брать модели?

- SOTA (state-of-the-art) архитектуры уже наверняка кем-то реализованы и обучены
  - torchvision.models
  - pretrainedmodels.pytorch
  - pytorch-image-models (timm)
  - ...

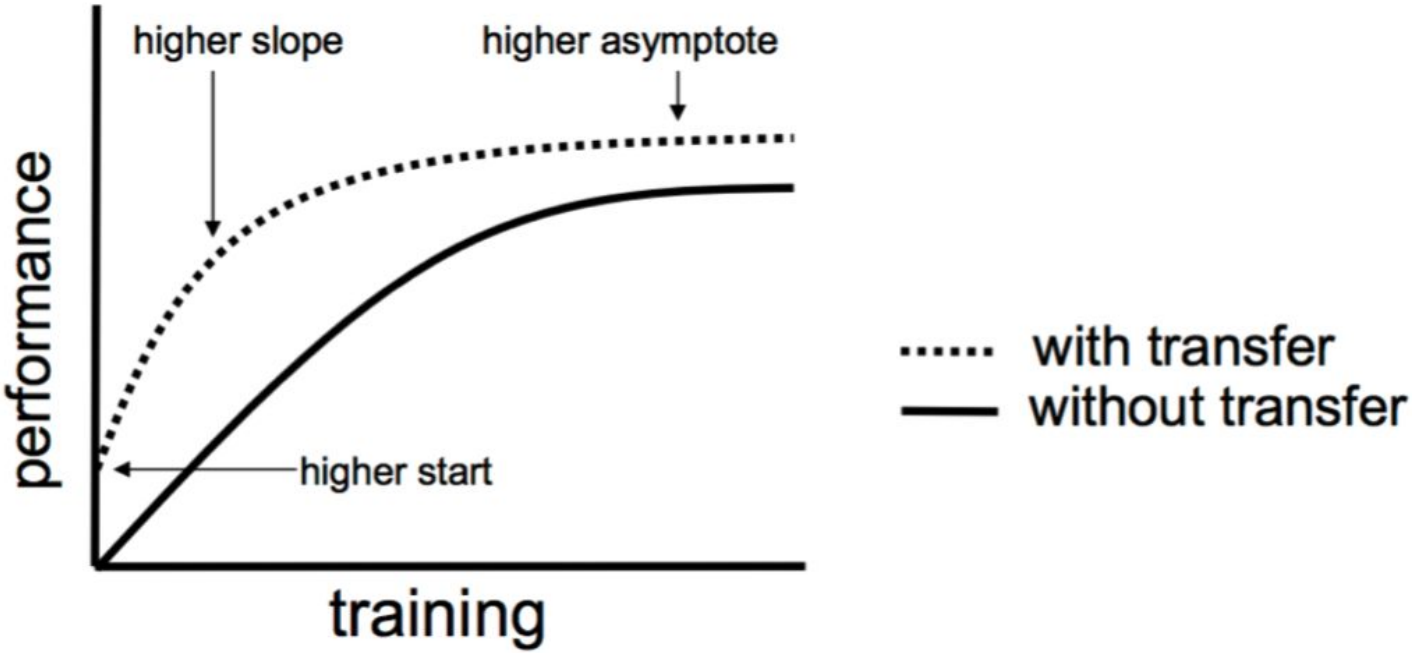
Где брать модели?

```python
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



Transfer Learning

- Для любознательных: Transfer Learning - Machine Learning's Next Frontier

Итоги

Итоги

- **Neural Architecture Search** - метод "автоматического" подбора архитектуры под конкретную задачу
  - Полезно, но дорого
- **Transfer Learning** - идея о передаче знания от модели к модели, даже если они учились на разных доменах под разные задачи
  - **Fine-tuning** полезно использовать почти **всегда**

В следующих сериях

- Задача **Object Detection**
  - **Two-stage:** *-RCNN
  - **One-stage:** SSD, YOLO, ...