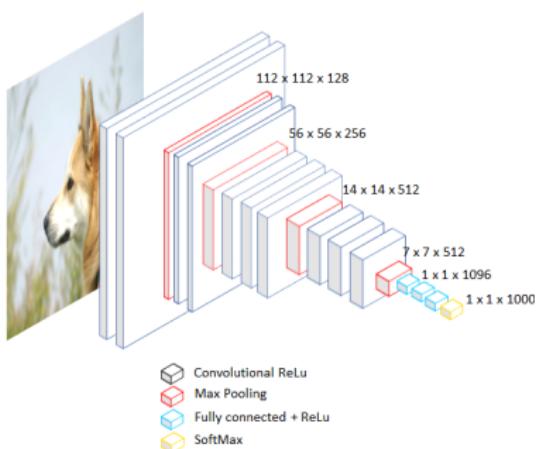


Архитектуры классификационных нейросетей

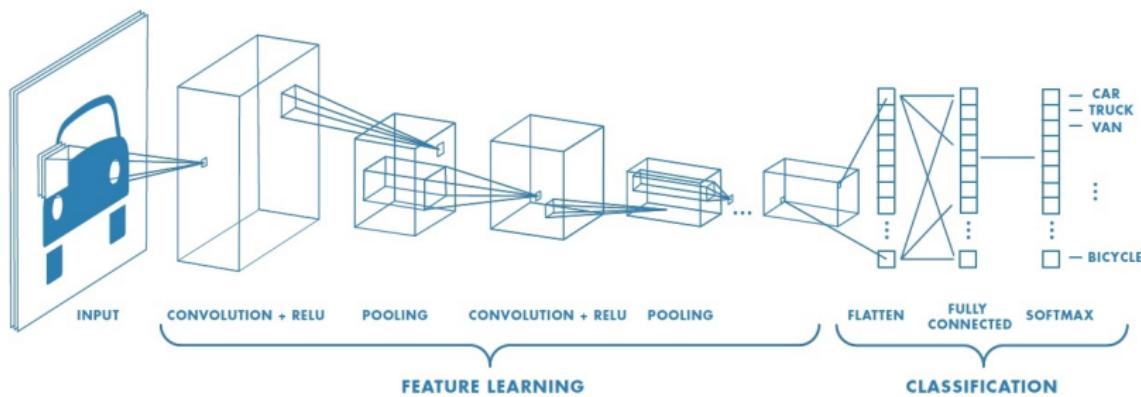
Виктор Китов
victorkitov.github.io



Содержание

- 1 ImageNet и его лидеры
- 2 Развитие идей ResNet
- 3 Идеи вычислительно эффективных архитектур

Сверточная нейросеть

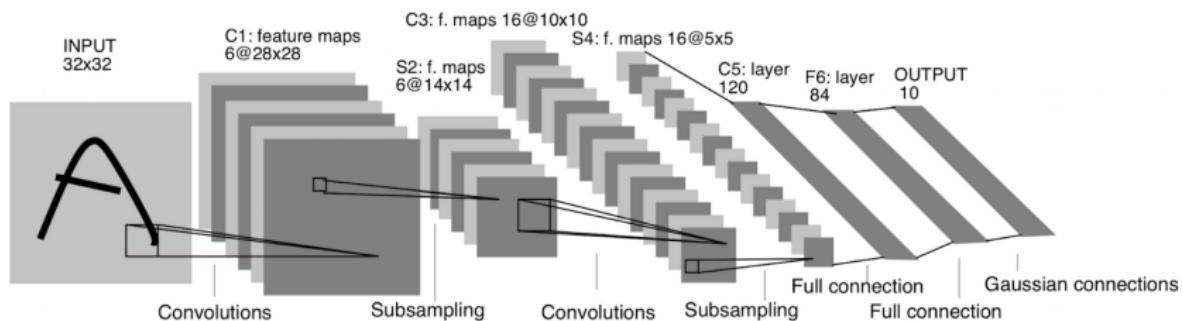


Более глубокие слои :

- имеют большую область видимости (receptive field)
- выучивают более и более общие признаки
 - пример: цвета->границы->линии, углы, изгибы->геометрические фигуры->глаза, нос, рот, брови

LeNet

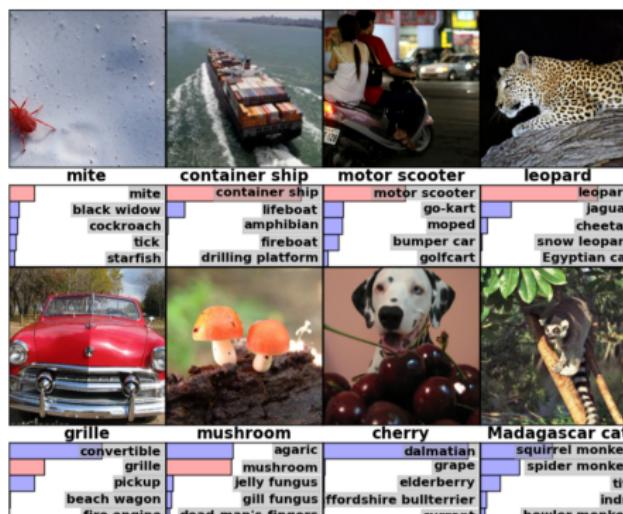
LeNet (1998) - архитектура для распознавания рукописных цифр и символов.



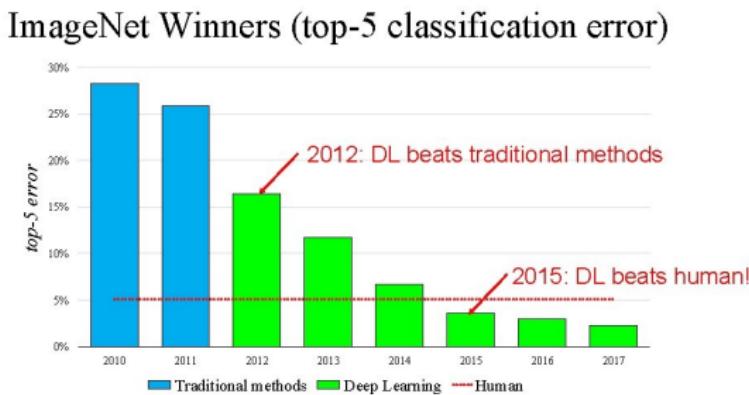
- Нелинейность - tanh. Subsampling реализуется через AvgPooling (2x2).
- В современных архитектурах используют не Gaussian connections в конце, а SoftMax.

Выборка ImageNet

- 1000 классов (включая 120 пород собак!).
- >14 миллионов вручную классифицированных изображений.
- Классификаторы оценивались по top-5 точности
 - (попадает ли верный класс в top-5 рекомендаций)



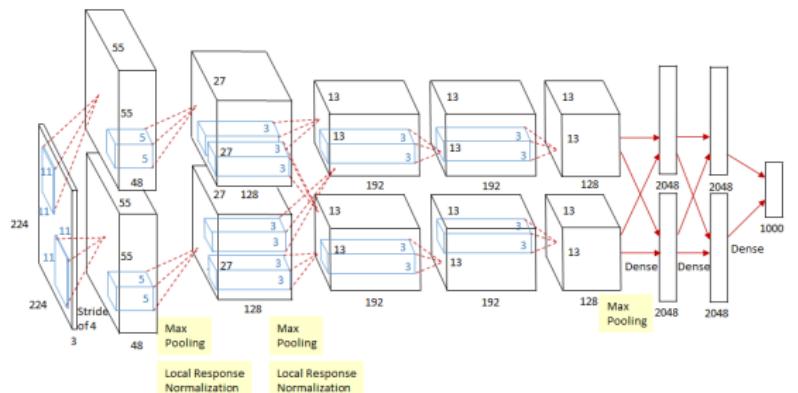
Динамика top-5 точности на ImageNet



- Начиная с 2012 - начинают использоваться глубокие свёрточные нейросети.
- Начиная с 2015 - превосходят точность человека.¹
 - для заданных 1000 классов

¹Andrew Karpathy human test.

AlexNet (2012)



LeNet	AlexNet
Image: 28 (height) × 28 (width) × 1 (channel)	Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 5x5 kernel=2 padding 2x2x4	Convolution with 11x11 kernel=4 stride 4x4=56
Pool with 2x2 average kernel=4 stride 14x14=6	Pool with 3x3 max kernel=4 stride 26x26=58
Convolution with 5x5 kernel=1 stride 10x10=18	Convolution with 5x5 kernel=4 stride 26x26=256
Pool with 2x2 average kernel=2 stride 5x5=16	Pool with 3x3 max kernel=4 stride 12x12=256
Dense: 120 fully connected neurons	Convolution with 3x3 kernel=1 stride 12x12=36
Dense: 84 fully connected neurons	Convolution with 3x3 kernel=1 stride 12x12=364
Dense: 10 fully connected neurons	Convolution with 3x3 kernel=1 stride 12x12=256
	Output: 1 of 10 classes
	Pool with 3x3 max kernel=5x5=256
	Dense: 4096 fully connected neurons
	ReLU, dropout p=0.5
	Dense: 4096 fully connected neurons
	ReLU, dropout p=0.5
	Dense: 1000 fully connected neurons
	Output: 1 of 1000 classes

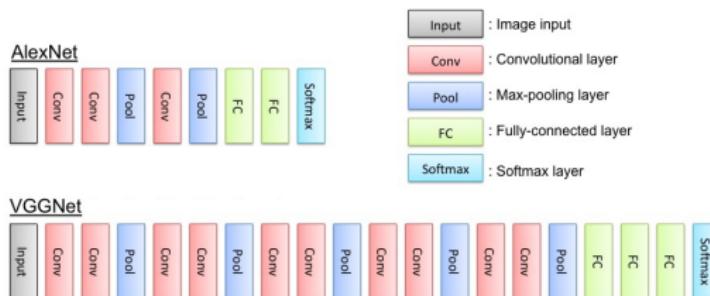
По сравнению с LeNet:

- ↑слоев, ↑нейронов в MLP.
- Свёртки разного размера: 11x11, 5x5, 3x3. AvgPooling->MaxPooling.
- tanh->ReLU, dropout (перед 1м и 2м полностью связанными слоями), аугментация данных.
- сеть разбивалась на две (по слоям, с разными параметрами), которые обучались на 2x GPU (из-за нехватки памяти).

VGG² (2014)

- VGG использовало больше слоёв, чем AlexNet.
- Использовалось постепенное ↓разрешения:
 - conv 11x11 и conv 5x5 -> суперпозиция свёрток 3x3.
Pooling 3x3 -> pooling 2x2.
- Вначале обучалась более короткая сеть, потом настроенными весами инициализировалась более длинная.

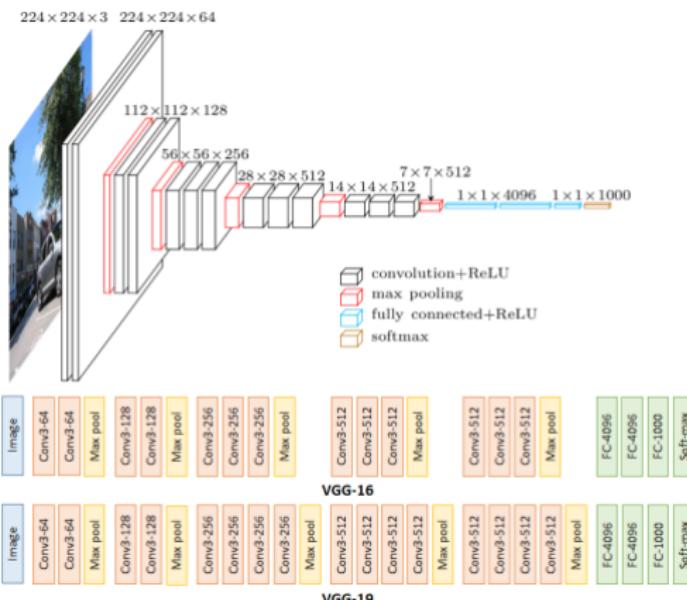
AlexNet vs. VGG



²<https://arxiv.org/pdf/1409.1556.pdf>

VGG (2014)

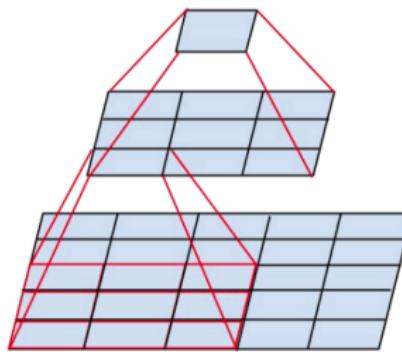
Размерность слоёв VGG ↓ постепенно:



Каскад свёрток в VGG

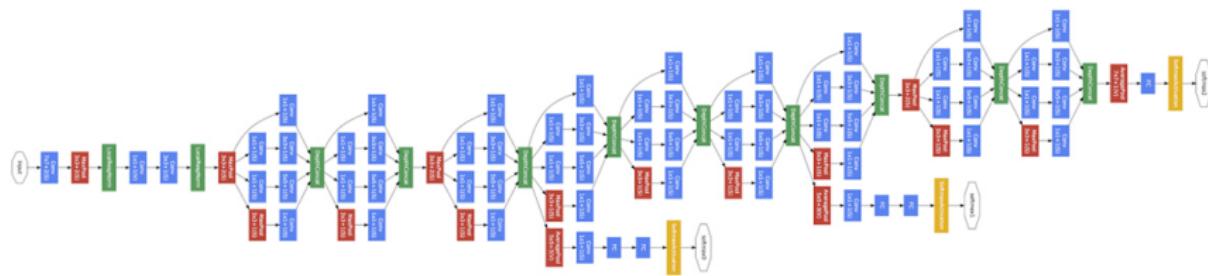
Свёртки с большим ядром заменялись на каскад свёрток 3x3:

- та же область видимости
- быстрее, меньше параметров
- дополнительные нелинейности (в середине, а не только в конце)³



³Если не использовать нелинейности, будет ли выразительная сила одинакова у свёртки 5x5 и суперпозиции 2x 3x3 свёрток для одномерного и двумерного случая?

GoogleNet (Inception v1)⁴

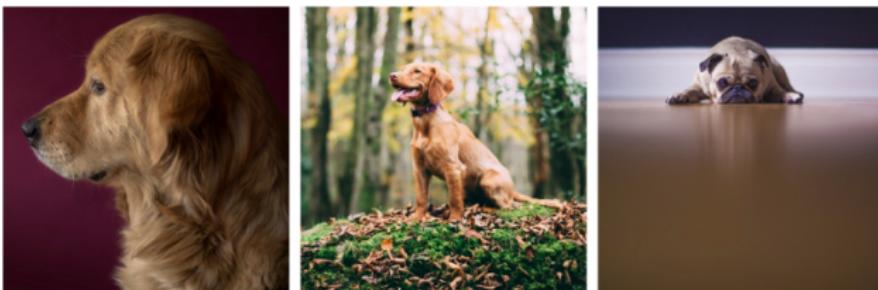


- Существенно меньше пар-ров, чем в VGG-19: в конце: GlobalAvgPooling и только 1 полно связный слой.
- Состоит из единообразных inception-модулей.
- Во время обучения добавляются 2 промежуточных выхода для лучшей настройки ранних слоёв.
- \downarrow # вычислений и # параметров с использованием 1x1 сверток.

⁴<https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf>

Решаемая проблема в inception-модуле GoogleNet

- Искомые объекты могут быть разного размера.
 - Нужно соответствующим образом менять размер применяемых свёрток.
 - Заранее размер объекта не знаем.



- Идея: применим свёртки разных размеров и позволим нейросети самой определять, какую использовать.

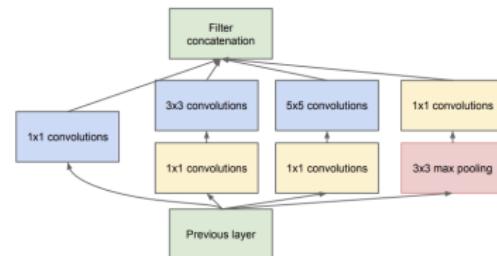
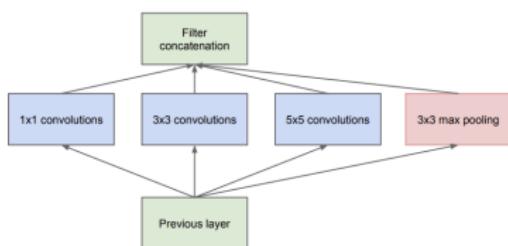
Модуль Inception

Модуль Inception использует идею bottleneck:

- ① свертки $1 \times 1 \downarrow$ #фильтров перед др. свёртками
- ② применяются свёртки разных размеров
- ③ конкатенация результатов

Дальнейшие слои сами определяют со свёртками какого размера работать.

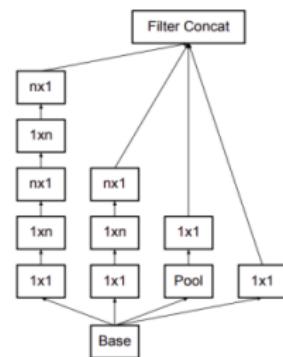
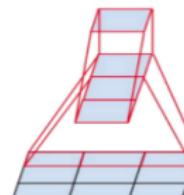
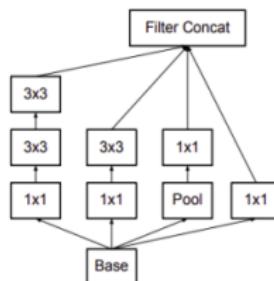
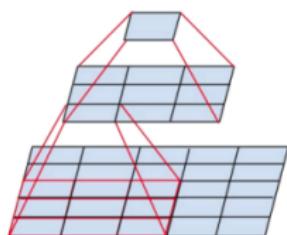
Идея объединения сверток (слева) и фактический inception-блок (с \downarrow размерности)



Inception v2,3 (2015)⁵

В Inception V2 использованы идеи:

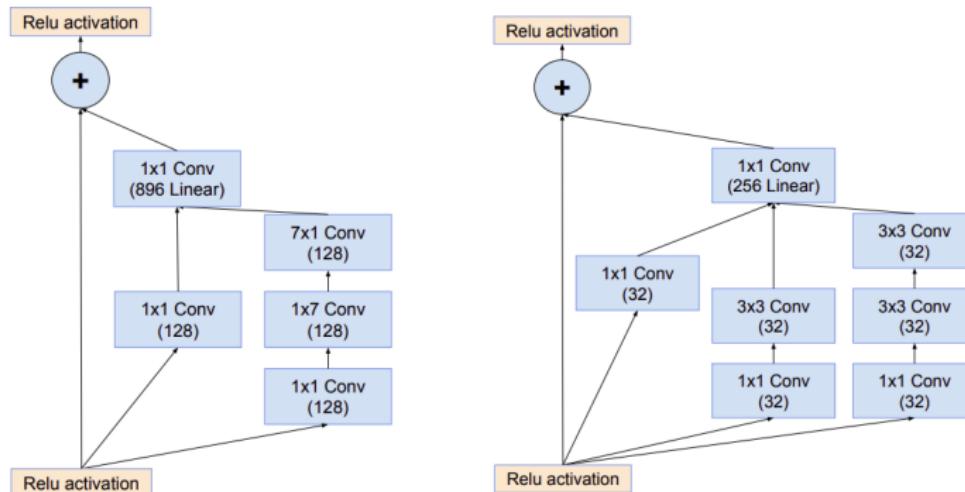
- conv 5x5 -> [conv 3x3, conv 3x3]
- conv 7x7 -> [conv 7x1, conv 1x7]



⁵<https://arxiv.org/pdf/1512.00567v3.pdf>

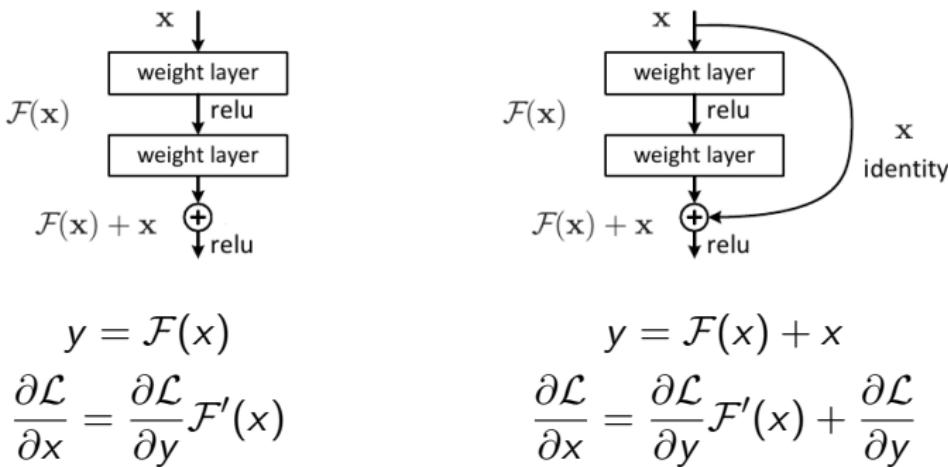
Inception v4 (2016)⁶

Inception v4: добавлены тождественные связи (иdea из ResNet).



⁶<https://arxiv.org/pdf/1602.07261v2.pdf>

ResNet (2015)⁷



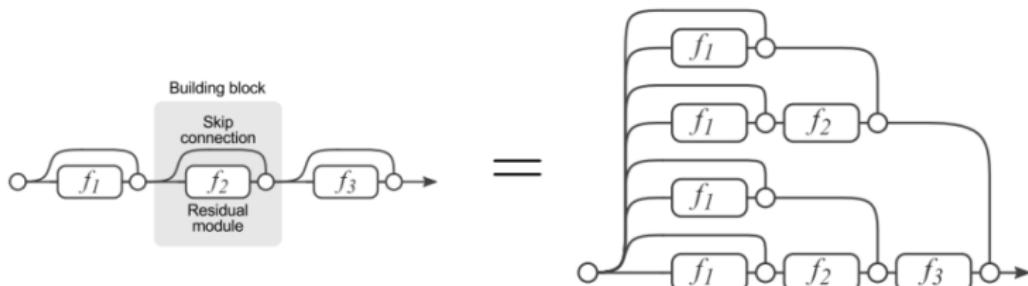
Тождественная связь (skip connection) устанавливает более непосредственную связь между ранними слоями и выходом (борется с нестабильными градиентами).

⁷<https://arxiv.org/pdf/1512.03385.pdf>

ResNet

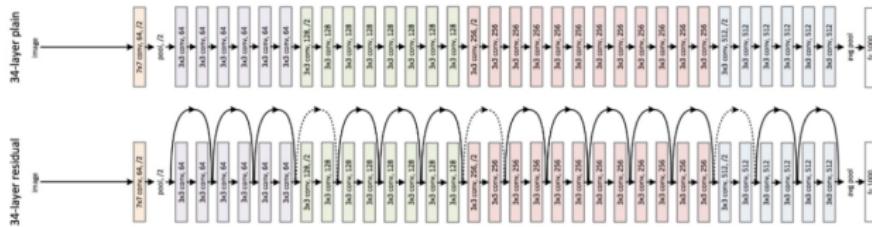
Другие обоснования ResNet блока:

- разумно инициализировать нелинейные слои малыми числами
 - вначале почти тождество
- ведет себя как ансамбль моделей:

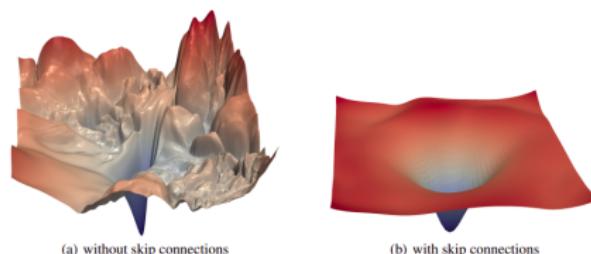


Поверхность ф-ции потерь становится более гладкой⁸

Обычная сеть и ResNet



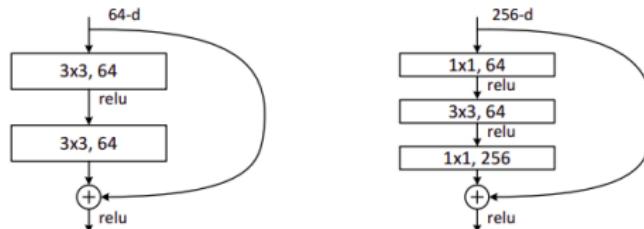
Эффект добавления тождественных связей на функцию потерь:



⁸Visualizing the Loss Landscape of Neural Nets (2018).

Варианты ResNet блока

ResNet блоки для ResNet 18,34 (слева) и ResNet 50,101,152 (справа)

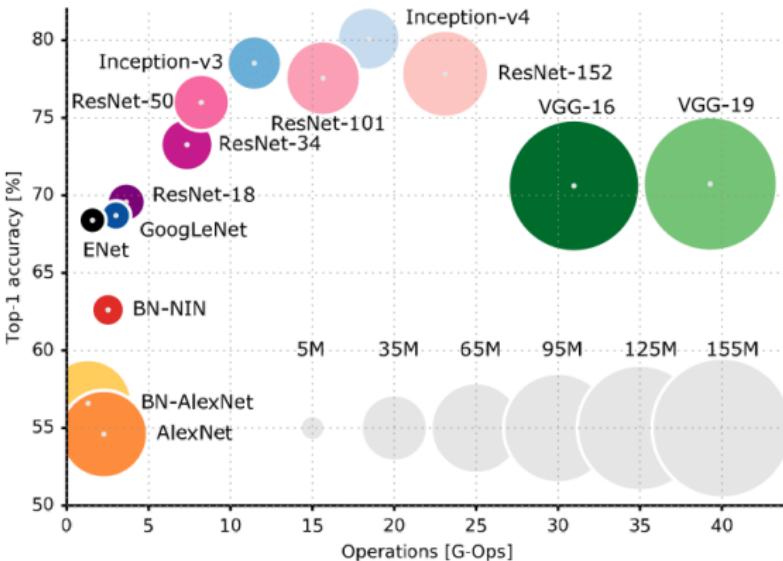


- Во втором ResNet блоке используются conv 1×1 для \downarrow разрешения перед conv 3×3 , потом возврат к старому разрешению (bottleneck).
- Каждый прямоугольник включает: batch-norm, свёртка, нелинейность
- В отдельных слоях, где \downarrow разрешение: внутри x и $\mathcal{F}(x)$ pool 2×2 .

Архитектура конфигураций ResNet 18, 50, 101, 152

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Сравнение моделей классификации на ImageNet⁹



Помимо точности модели важно #операций и требования по памяти.

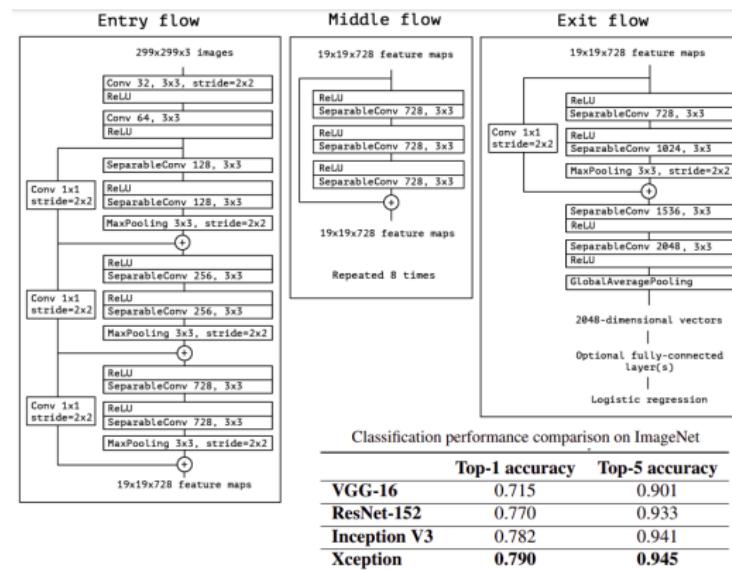
⁹<https://arxiv.org/pdf/1605.07678.pdf>

Содержание

- 1 ImageNet и его лидеры
- 2 Развитие идей ResNet
- 3 Идеи вычислительно эффективных архитектур

Xception (2017)¹⁰

Xception (2017) - используются сепарабельные свертки (т.е. depthwise, потом pointwise) вместо обычных.



¹⁰<https://arxiv.org/pdf/1610.02357.pdf>

ResNeXt (2017)¹¹

ResNeXt (2017) - использование групповых свёрток
(действующих независимо на блоках из 4x каналов).

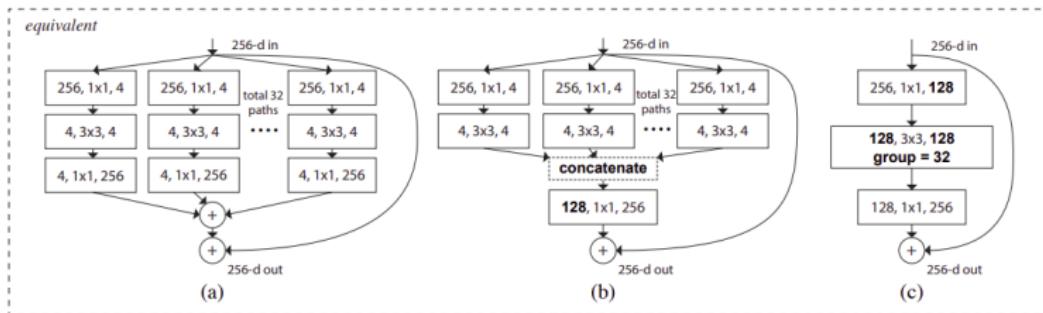
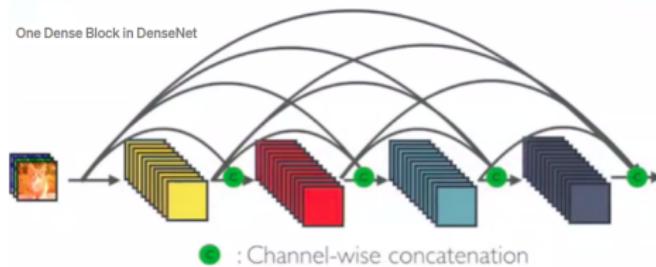
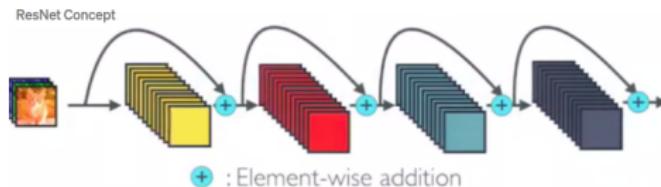
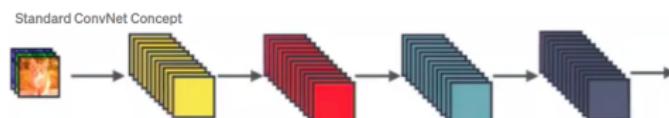


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

	setting	5K-way classification		1K-way classification	
		top-1	top-5	top-1	top-5
Error (%) on ImageNet-5K .	ResNet-50	1 × 64d	45.5	19.4	27.1
	ResNeXt-50	32 × 4d	42.3	16.8	24.4
	ResNet-101	1 × 64d	42.4	16.9	24.2
	ResNeXt-101	32 × 4d	40.1	15.1	22.2
					5.7

¹¹<https://arxiv.org/pdf/1611.05431.pdf>

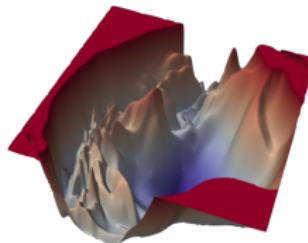
DenseNet (2018)¹²



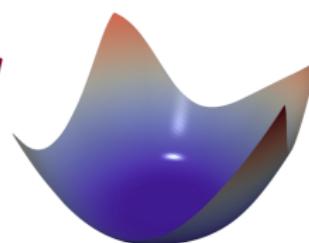
¹²<https://arxiv.org/pdf/1608.06993.pdf>

DenseNet (2018)

- В ResNet признаки суммируются, а в DenseNet конкатенируются.
- Преимущества DenseNet:
 - лучше настройка ранних слоёв
 - в DenseNet меньше параметров-информация о прошлом подаётся напрямую, поэтому не нужно под это резервировать доп. каналы.
 - функция потерь более удобная для оптимизации¹³:



(a) ResNet-110, no skip connections

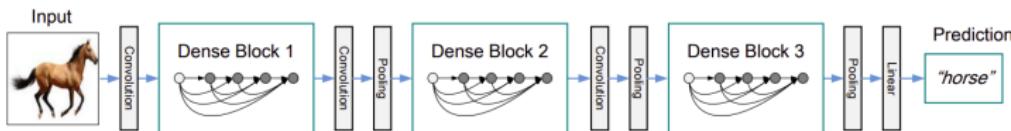


(b) DenseNet, 121 layers

¹³Visualizing the Loss Landscape of Neural Nets (2018).

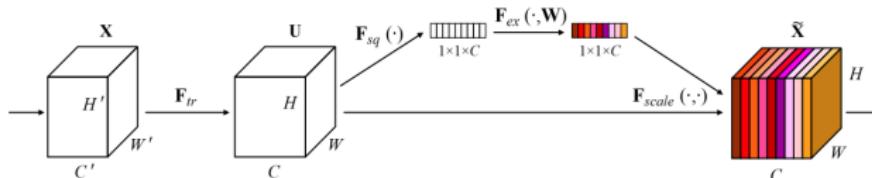
DenseNet блоки

- #каналов растет линейно. Чтобы ↓сложность сеть разбивается на dense-блоки и переходные слои, в которых ↓ #каналов свёртками 1×1 и ↓ разрешение пулингом.



SENet (2017)¹⁴

- Squeeze-and-exitation network (SENet) использует простую полносвязную сеть над GlobalAvgPooling, чтобы перенормировать каналы:



- GlobalAvgPool канала c :

$$z = F_{sq}(u_c) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j), c = 1, 2, \dots, C.$$
- Расчет весов: $s = \sigma(W_2 \cdot \text{ReLU}(W_1 z))$
 - $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ - сжатие, $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ - разжатие (bottleneck)
 - Перевзвешивание: $u'_c = F_{scale}(u_c, s) = s_c u_c$

¹⁴<https://arxiv.org/pdf/1709.01507.pdf>

SENet (2017)

- Используется "самовнимание" (self-attention): \uparrow вклад важных признаков и \downarrow неважных в зависимости от самих признаков.
 - \bullet концентрируем внимание на важных признаков в зависимости от объекта
- \uparrow выразительности сети: используются не только локальные характеристики (активации свёрток), но и глобальные (средние значения признаков).
- SENet-154 точнее ранее изученных архитектур:

Single-crop error rates (%) of state-of-the-art CNNs on ImageNet validation set with crop sizes 224×224 and $320 \times 320 / 299 \times 299$.

	224×224		$320 \times 320 / 299 \times 299$	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [13]	23.0	6.7	21.3	5.5
ResNet-200 [14]	21.7	5.8	20.1	4.8
Inception-v3 [20]	-	-	21.2	5.6
Inception-v4 [21]	-	-	20.0	5.0
Inception-ResNet-v2 [21]	-	-	19.9	4.9
ResNeXt-101 (64 \times 4d) [19]	20.4	5.3	19.1	4.4
DenseNet-264 [17]	22.15	6.12	-	-
Attention-92 [58]	-	-	19.5	4.8
PyramidNet-200 [77]	20.1	5.4	19.2	4.7
DPN-131 [16]	19.93	5.12	18.55	4.16
SENet-154	18.68	4.47	17.28	3.79

Содержание

- 1 ImageNet и его лидеры
- 2 Развитие идей ResNet
- 3 Идеи вычислительно эффективных архитектур

MobileNet

- MobileNet v1¹⁵:
 - MaxPooling 2x2 -> свёртки со stride=2
 - стандартная свёртка -> depthwise separable conv
- MobileNet v2: добавлены тождественные связи
 - как в ResNet
- MobileNet v3: добавлено перевзвешивание каналов
 - как в SENet
- SqueezeNet:
 - не используется MLP в конце,
выход=SoftMax(GlobalAvgPooling)
 - архитектура похожа на Inception: conv1x1-снижение размерности, затем параллельно conv1x1 и conv3x3 и конкатенация выходов.
 - используются тождественные связи

¹⁵ <https://arxiv.org/abs/1704.04861>

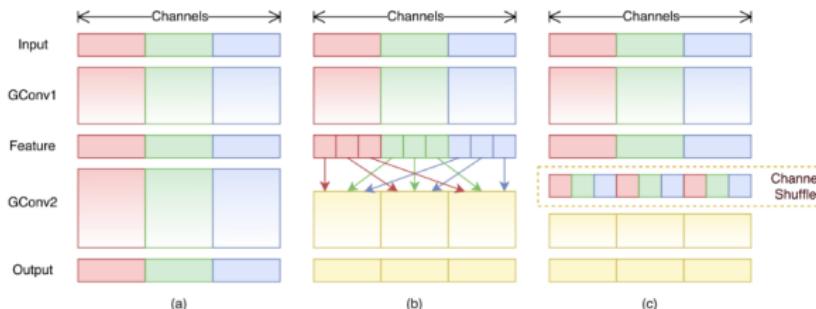
ShuffleNet¹⁶

- SqueezeNet:

- используются последовательность conv1x3 и conv3x1.
- вначале изображение сильно сжимается conv7x7, затем работаем уже в малом разрешении

- ShuffleNet:

- используются последовательно групповые свёртки
- чтобы признаки не замыкались в рамках групп, каналы переупорядочиваются:



¹⁶Обзор мобильных архитектур.

Заключение

- Если задача связана с изображениями, берут слои хороших архитектур на ImageNet.
- Важны точность, скорость и требования по памяти.
- Улучшение обучения ранних слоёв:
 - обучение простой сети, инициализация полученными весами более сложной.
 - использование промежуточных выходов во время обучения
 - использование тождественных связей (skip connection)
- Упрощения больших многомерных свёрток:
 - предварительное ↓ разрешения через conv 1x1
 - групповые свертки, сепарабельные свёртки.