

Современные сверточные архитектуры

Сергей Николенко

НИУ ВШЭ – Санкт-Петербург

03 октября 2020 г.

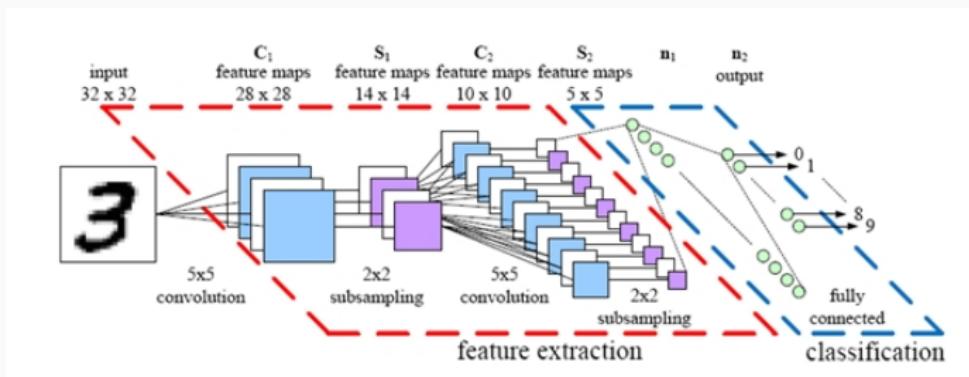
Random facts:

- 3 октября в Корее – День основания государства, официальный выходной (один из пяти в году) в честь образования первого государства корейской нации в 2333 году до нашей эры легендарным королем-богом Тангун Вангомом; а в России 3 октября – День ОМОН
- 3 октября 1945 г. Элвис Пресли принял участие в конкурсе юных талантов и завоевал второй приз: \$5 за песенку «Old Shep»
- 3 октября 1990 г. состоялось объединение Германии, в честь чего 3 октября в Германии – это День немецкого единства
- 3 октября 1995 г. на телеканале TV Tokyo состоялась премьера сериала «Евангелион» режиссёра Хидэаки Анно по одноимённой манге Ёсиюки Садамото
- 3 октября 2003 г. в Тибете прошёл конкурс красоты; на него отважилась явиться только одна девушка, 20-летняя Церинг Кьи, которая и получила титул «Мисс Тибет»

Свёрточные сети

Свёрточные нейронные сети: идея

- Эти идеи нашли отражение в искусственных сетях.
- Свёртки впервые появились в *Neocognitron* (Fukushima, 1979; 1980).
- В современной форме – в группе Лекуна во второй половине 1980-х.
- Главная идея: хочется применить одну и ту же операцию к разным частям изображения.



Свёрточные нейронные сети: идея

- Разобьём изображение на окна:



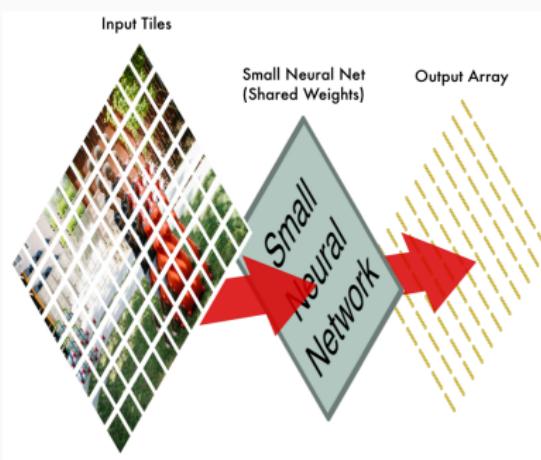
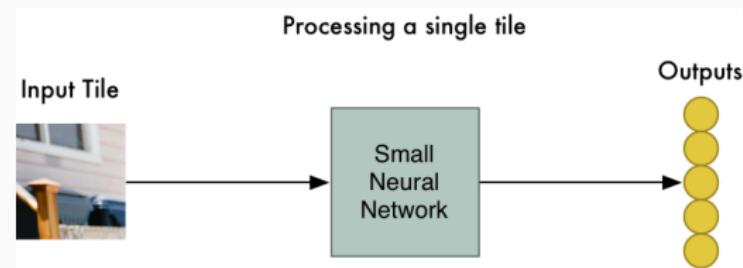
Свёрточные нейронные сети: идея

- Разобьём изображение на окна:



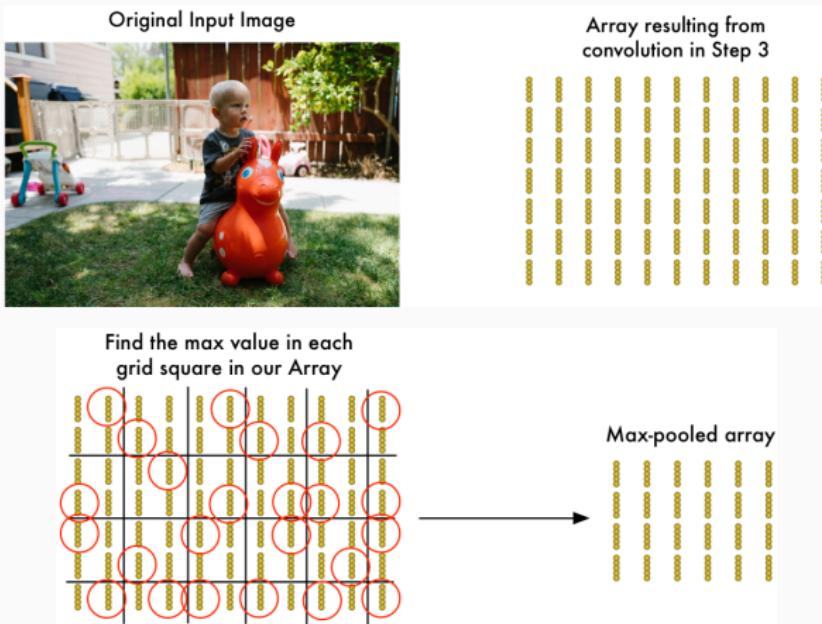
Свёрточные нейронные сети: идея

- Применим маленькую нейронную сеть к каждому окну:



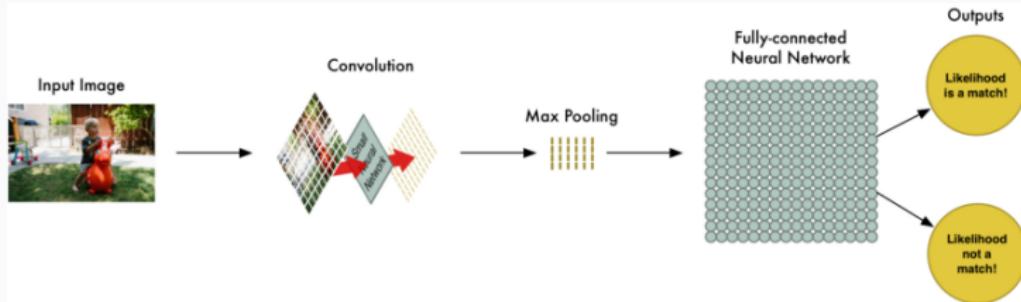
Свёрточные нейронные сети: идея

- Затем сожмём результат, сделав субдискретизацию (pooling; например, max-pooling) по маленьким окнам:

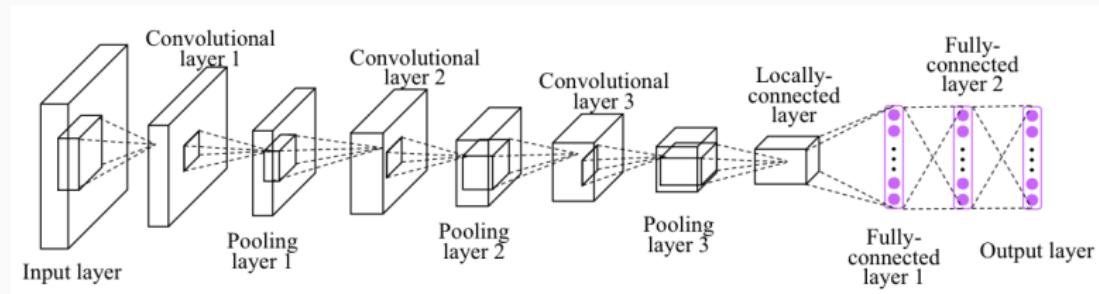


Свёрточные нейронные сети: идея

- А затем уже используем эти признаки, чтобы делать предсказания, например, полно связной сетью:

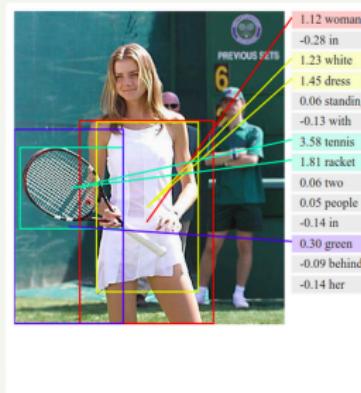
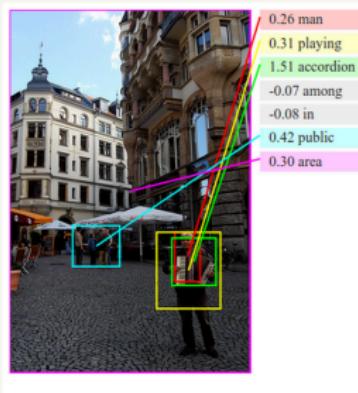
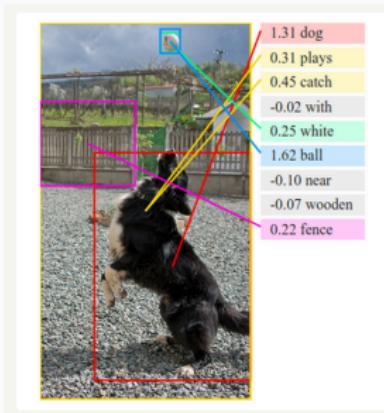


- Добавляя новые уровни, можно улучшить обобщающую способность, сделать признаки более общими:



Свёрточные нейронные сети: идея

- Более того, можно посмотреть, какие именно части картинки вызывают активации отдельных нейронов:



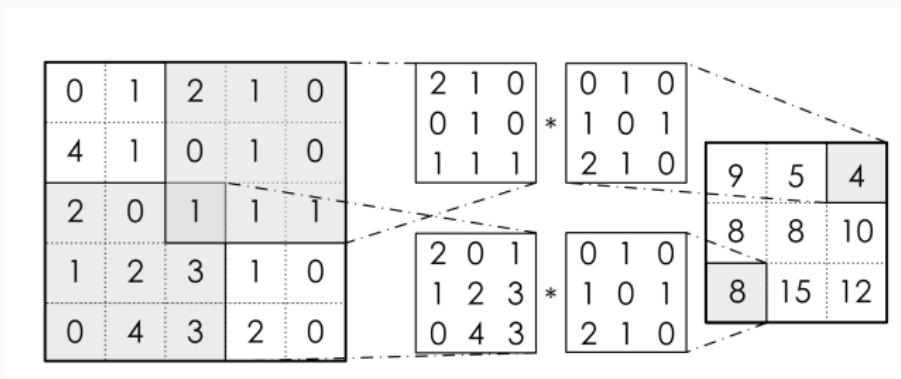
Свёрточные нейронные сети: идея

- Основная идея: мы применяем *одни и те же веса* по всей картинке.
- Это автоматически даёт устойчивость к переносам и т.п.
- И радикально сокращает число весов.
- Т.е. свёрточная структура – это такая радикальная форма регуляризации.

CNN формально

- Формально говоря, мы к каждому окну применяем одну и ту же маленькую матрицу.
- Если x^l – карта признаков на слое l , то двумерная свёртка размера $2d + 1$ с матрицей весов W размера $(2d + 1) \times (2d + 1)$ выглядит как

$$y_{i,j}^l = \sum_{-d \leq a,b \leq d} W_{a,b} x_{i+a, j+b}^l.$$



CNN формально

- Если x^l – карта признаков на слое l , то двумерная свёртка размера $2d + 1$ с матрицей весов W размера $(2d + 1) \times (2d + 1)$ выглядит как

$$y_{i,j}^l = \sum_{-d <= a,b <= d} W_{a,b} x_{i+a,j+b}^l.$$

- Затем обычно идёт нелинейность:

$$z_{i,j}^l = h(y_{i,j}^l).$$

- В современных сетях – почти всегда ReLU.

CNN формально

- И затем max-pooling (иногда average-pooling, но обычно max):

$$x_{i,j}^{l+1} = \max_{-d \leq a,b \leq d} z_{i+a,j+b}^l.$$

0	1	2	1
4	1	0	1
2	0	1	1
1	2	3	1

(a)

4	2	2
4	1	1
2	3	3

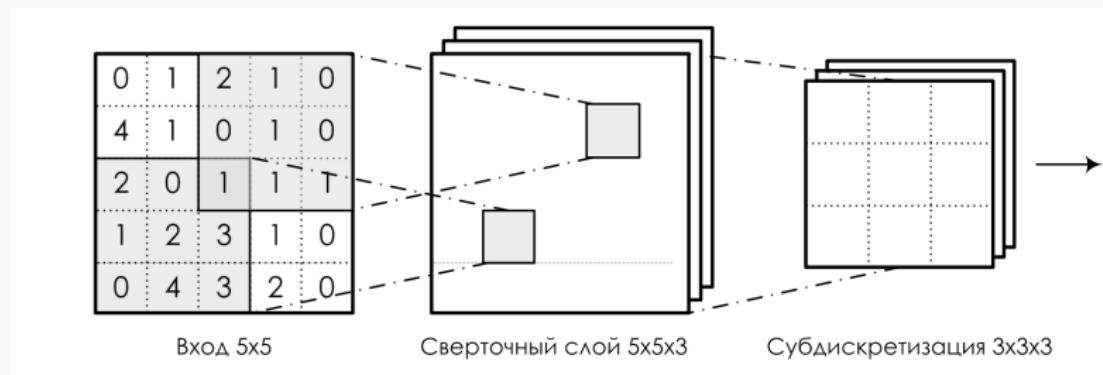
(б)

4	2
2	3

(в)

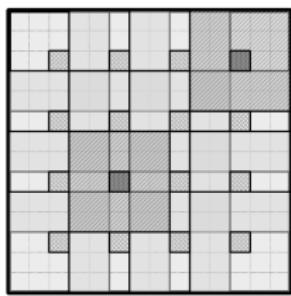
CNN формально

- Вот и получилась общая схема одного свёрточного слоя:

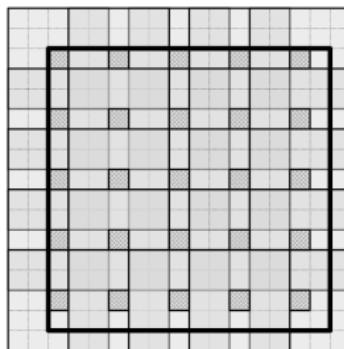


CNN формально

- Окнами можно покрывать по-разному; обычно всё-таки сохраняют исходный размер:



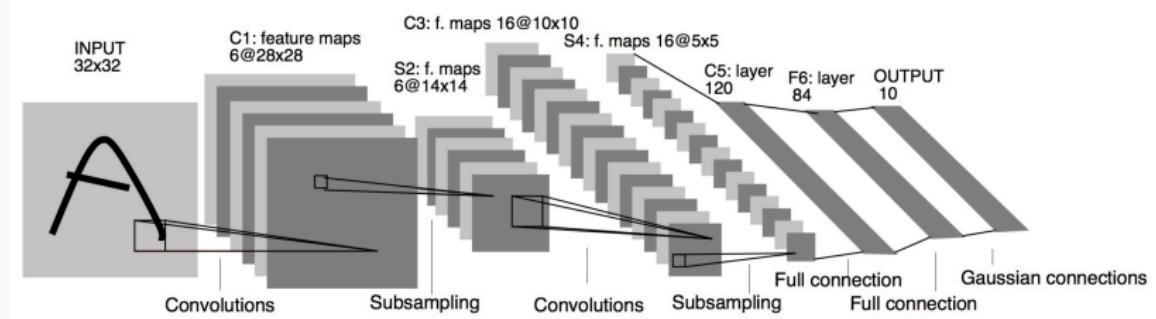
(а) окна 5×5 с шагом в 3 пикселя,
padding='VALID'



(б) окна 5×5 с шагом в 3 пикселя,
padding='SAME'

CNN формально

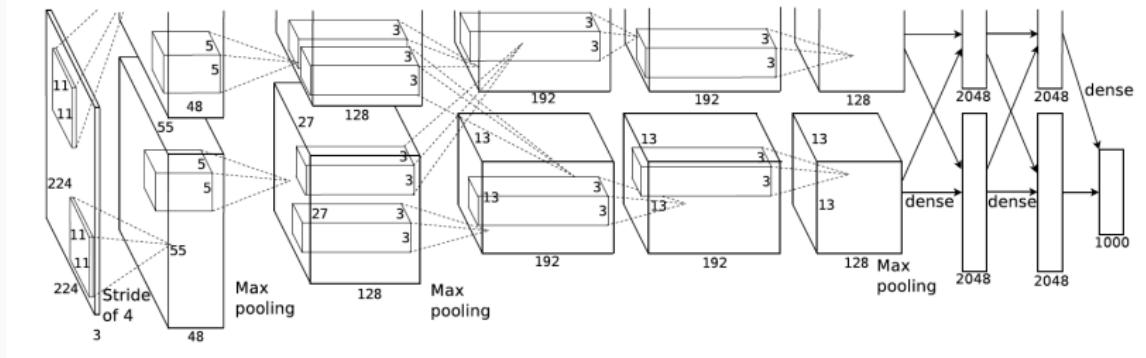
- Классическая архитектура *LeNet*:



- Современные CNN стали очень глубокими за счёт новых трюков – к ним мы скоро перейдём.

Imagenet и AlexNet

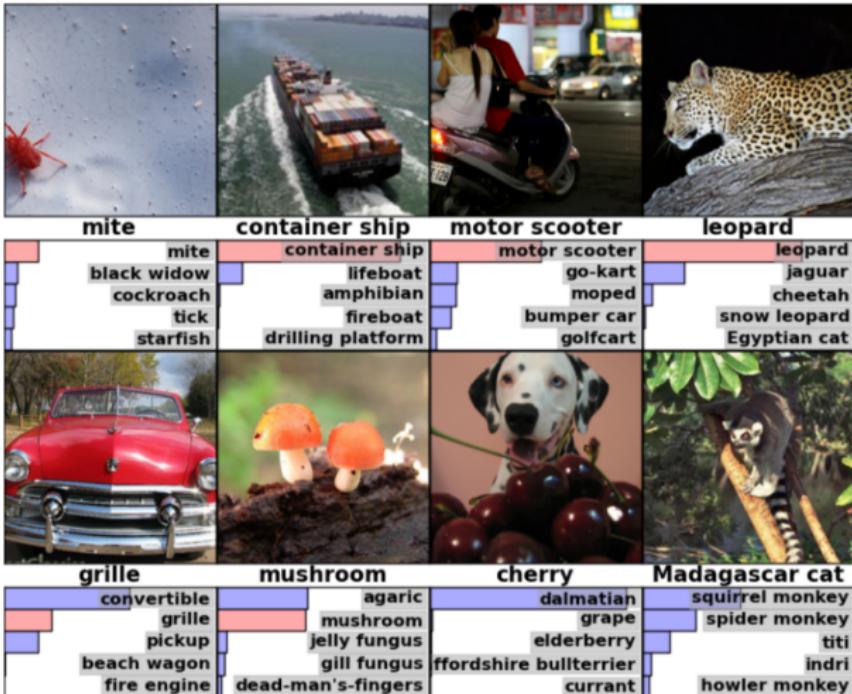
- ImageNet: > 15 миллионов картинок, ≈ 22000 категорий.
- Размечены руками (Amazon Mechanical Turk).
- Классическая сеть AlexNet (Krizhevsky, Sutskever, Hinton, 2012).



Предобработка данных

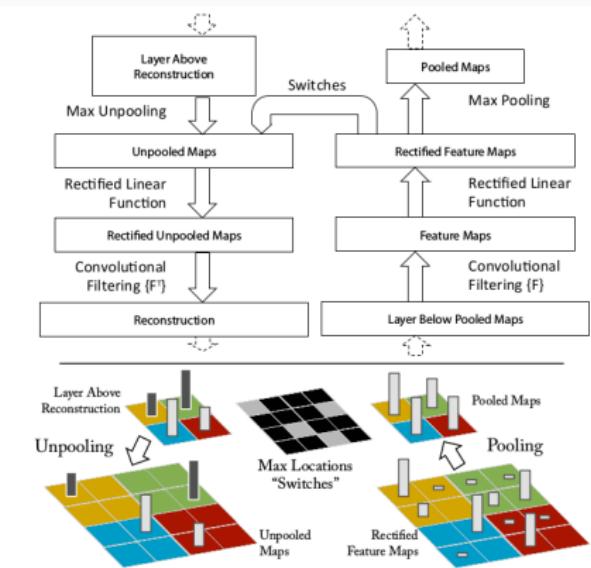
- С картинками можно делать data augmentation: изменять данные без изменения правильного ответа.
- (Krizhevsky, Sutskever, Hinton, 2012):
 - вырежем из 256×256 окна 224×224 (во время применения вырежем пять таких окон и усредним результаты);
 - отразим картинку по горизонтали;
 - поменяем интенсивности и освещённость (добавляя кратные главных компонент RGB-значений);
 - это увеличивает датасет в 2048 раз совершенно бесплатно. :)
- В AlexNet без data augmentation был бы оверфиттинг, это реально помогает.

AlexNet на ImageNet



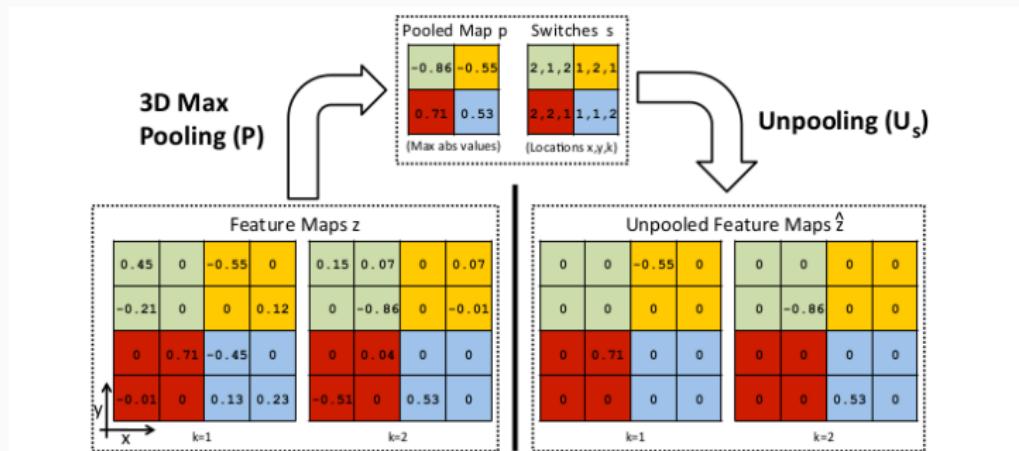
Деконволюция

- (Zeiler et al., 2011; Zeiler, Fergus, 2014): деконволюция как обратная свёртке операция.
- Чтобы обратить свёртку, достаточно транспонировать.



Деконволюция

- Интересный вопрос – как обратить субдискретизацию.
- Она, конечно, необратима, но можно запомнить положение максимумов и обратить примерно.

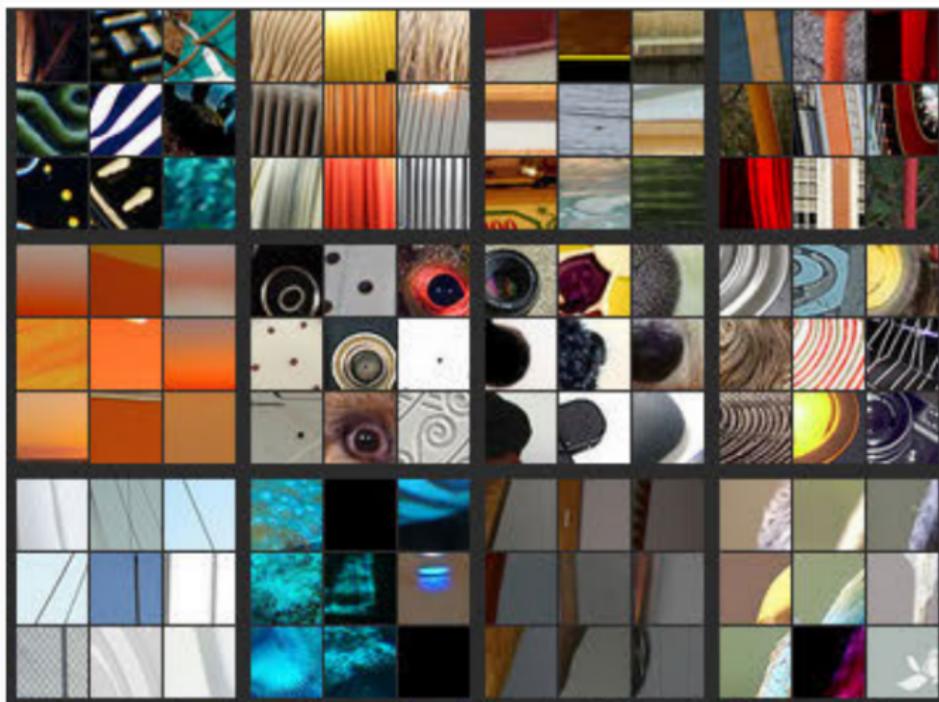


Уровень 1

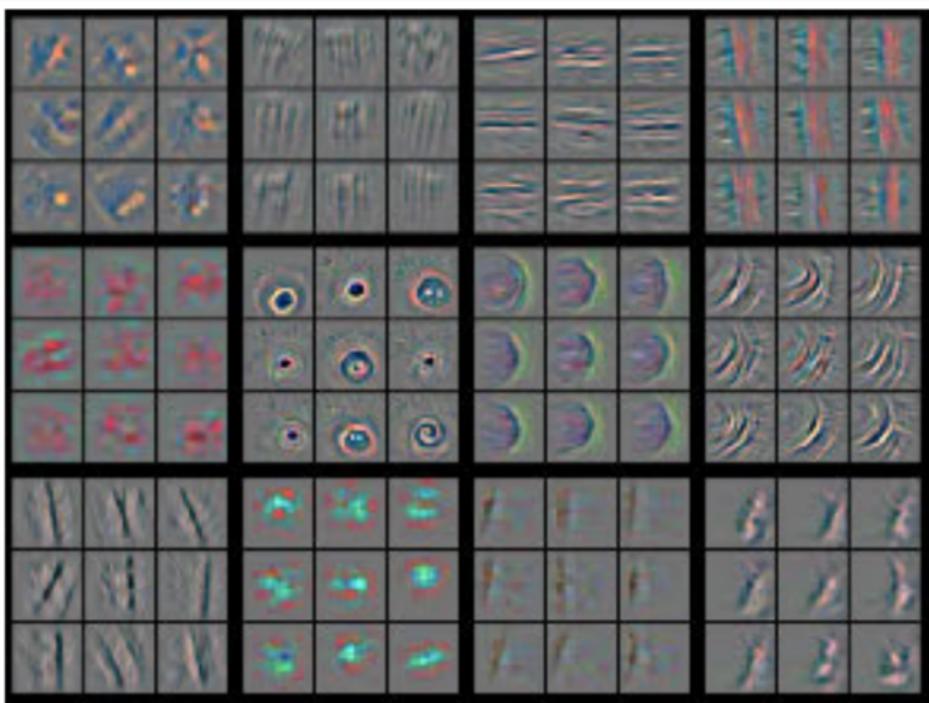
(Zeiler, Fergus, 2014): визуализируем признаки через деконволюцию. Это сеть, обученная на ImageNet.



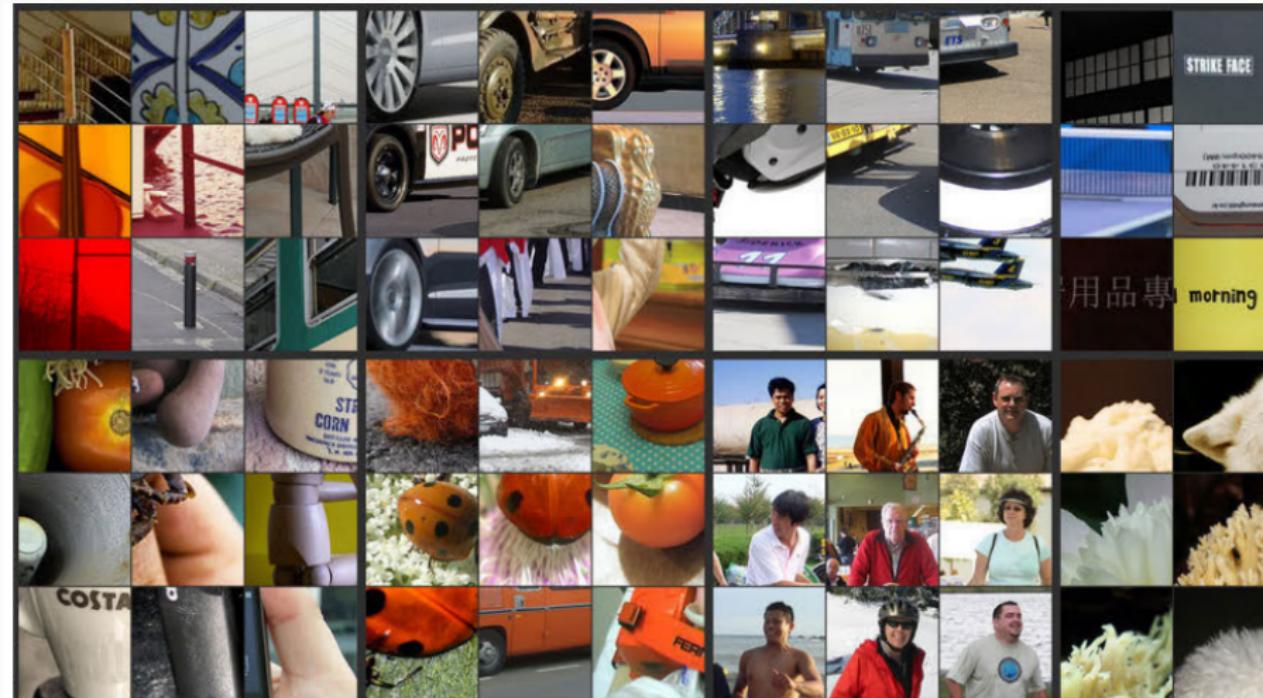
(Zeiler, Fergus, 2014): уровень 2



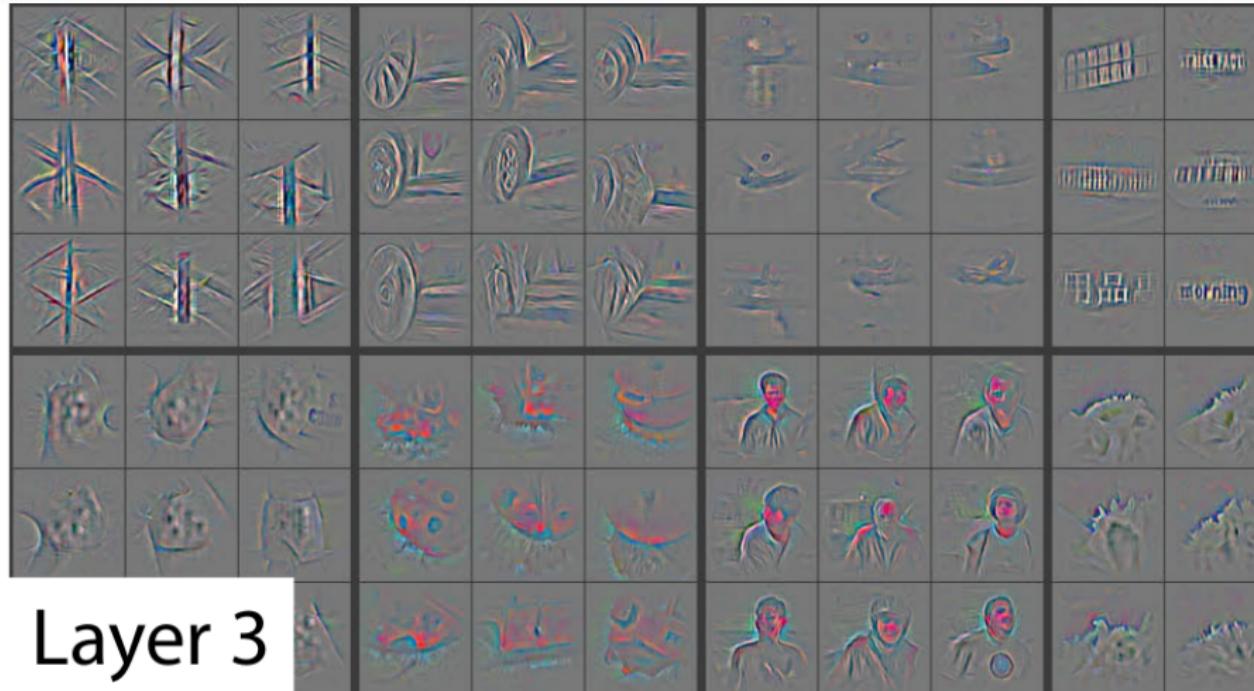
(Zeiler, Fergus, 2014): уровень 2



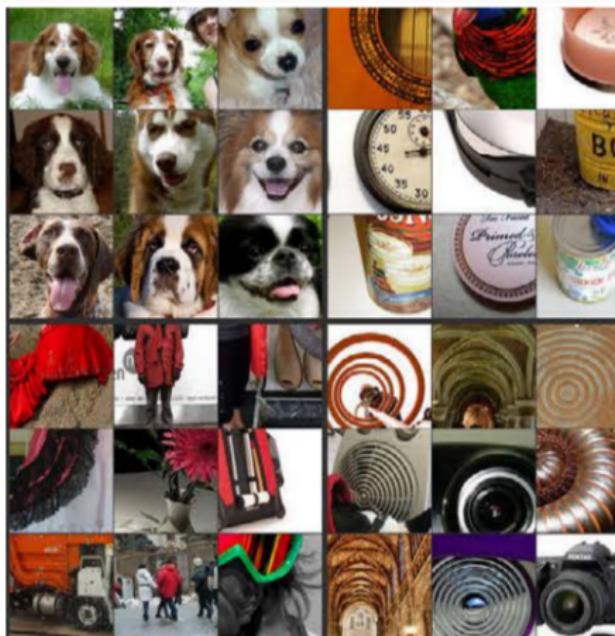
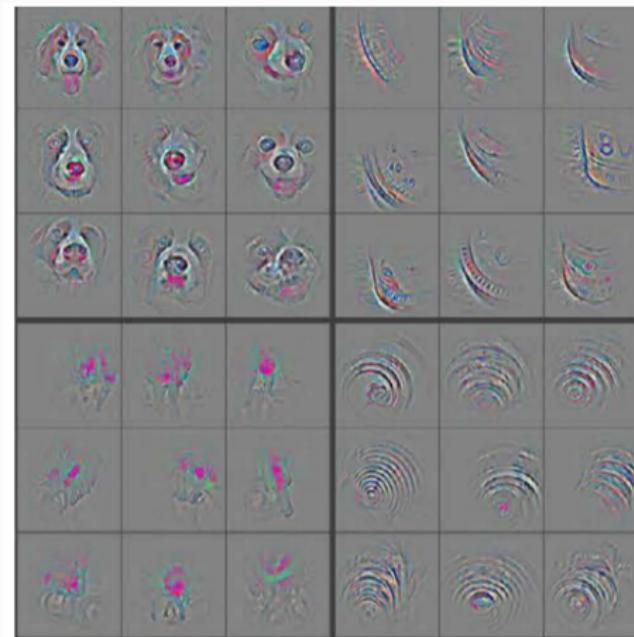
(Zeiler, Fergus, 2014): уровень 3



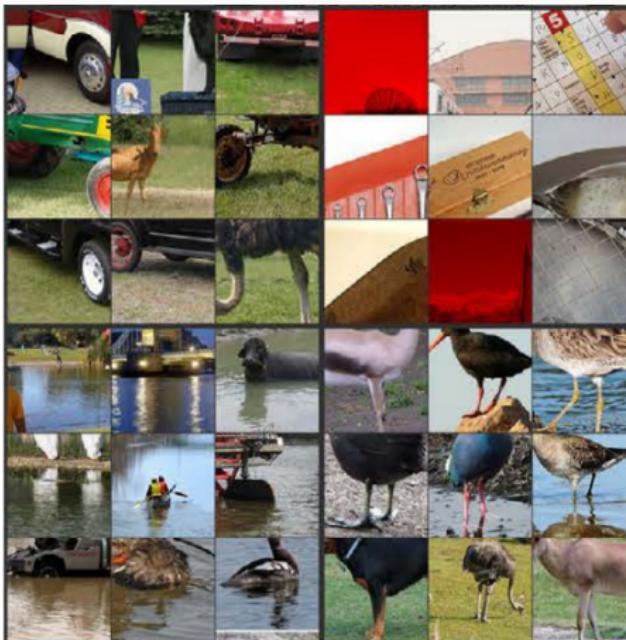
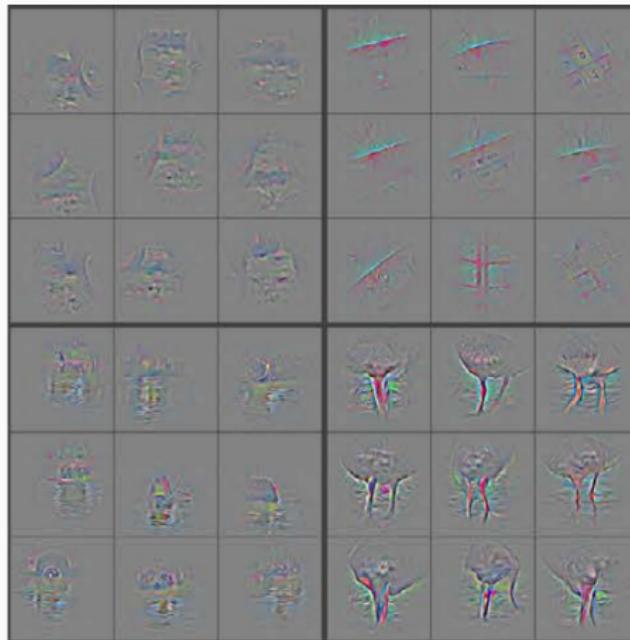
(Zeiler, Fergus, 2014): уровень 3



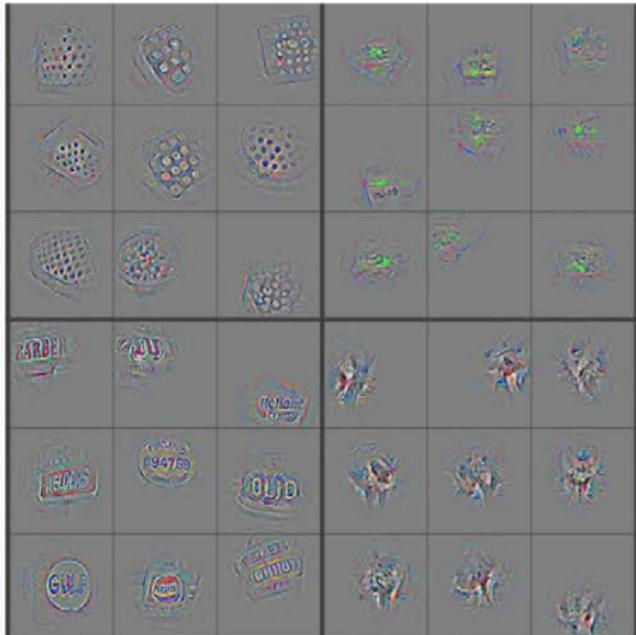
(Zeiler, Fergus, 2014): уровень 4



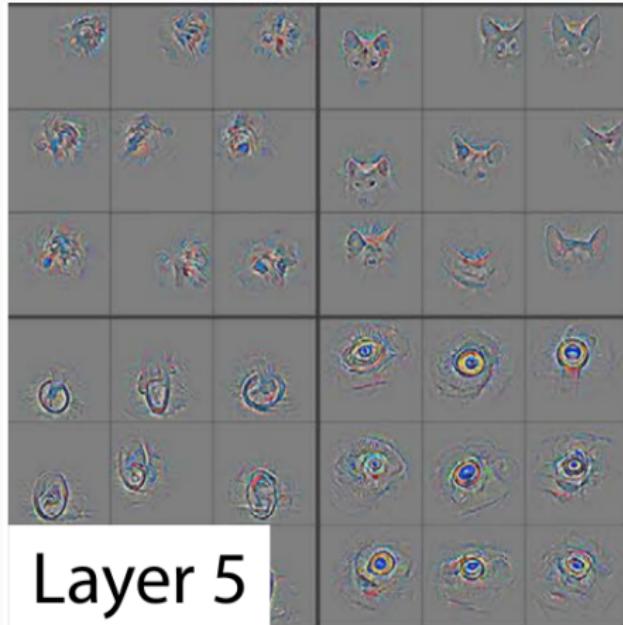
(Zeiler, Fergus, 2014): уровень 4



(Zeiler, Fergus, 2014): уровень 5



(Zeiler, Fergus, 2014): уровень 5

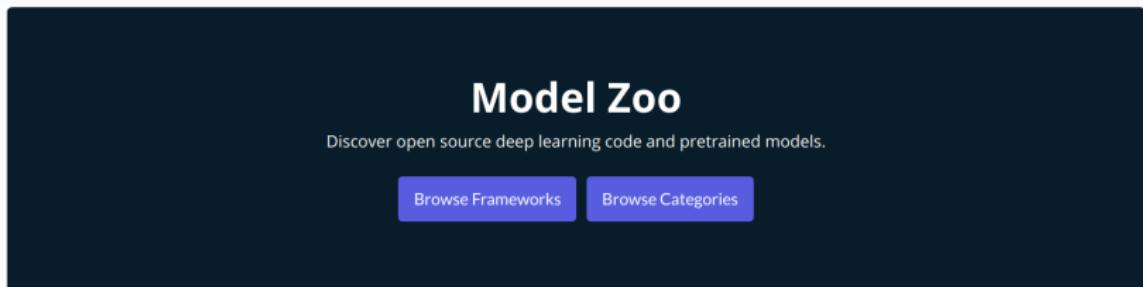


Layer 5

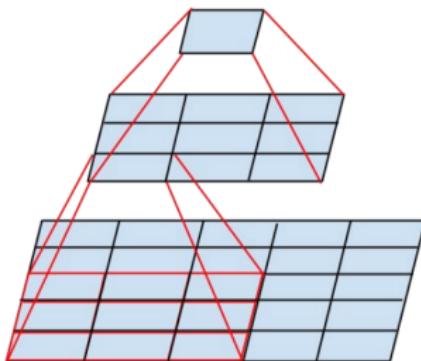


Современные свёрточные архитектуры

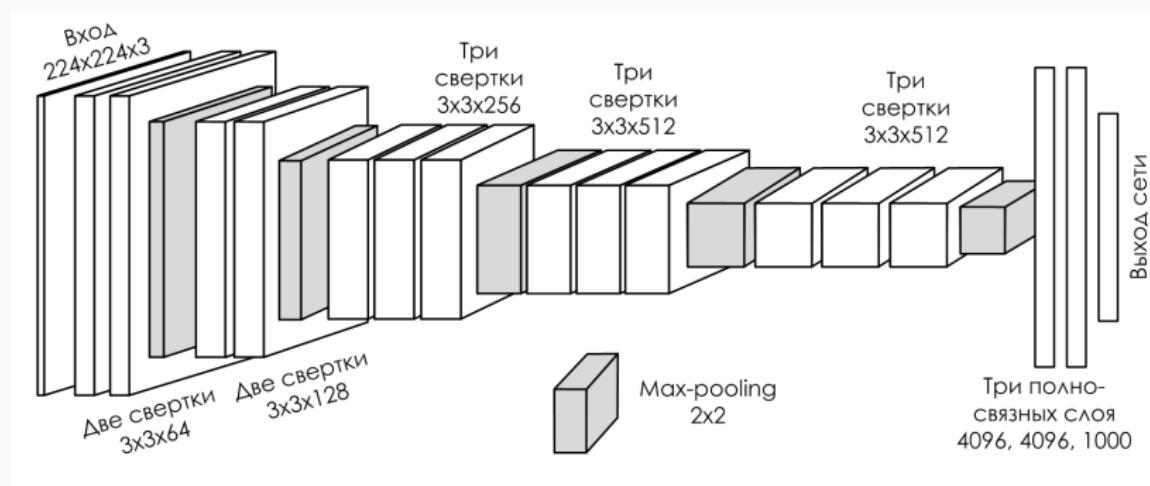
- Разных моделей, использующих CNN, очень много, и всё время появляются новые.
- Всё можно скачать, часто даже уже веса, предобученные на стандартных датасетах, есть.
- Но обычно они основаны на нескольких относительно стандартных идеях. Их-то мы и рассмотрим.



- VGG (Oxford Visual Geometry Group): давайте представлять большие свёртки как комбинации свёрток 3×3 .
- Это уменьшает число весов и делает сеть глубже.



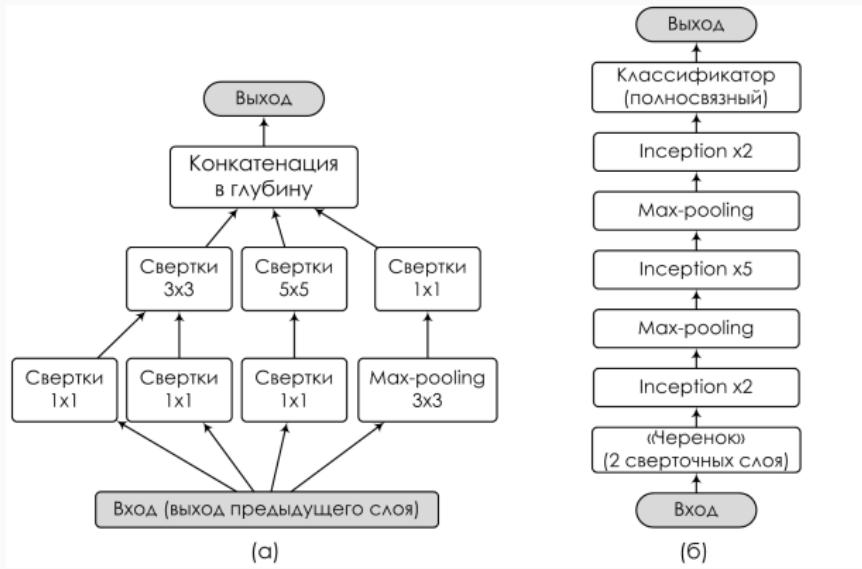
- Сеть VGG, использованная в ImageNet Large Scale Visual Recognition Competition (ILSVRC-2014).



- А сейчас NVIDIA специально оптимизирует GPU для свёрток 3×3 .

Inception

- Inception, разработанная командой GoogLeNet: давайте используем идею “сеть в сети” (network in network), чтобы сократить веса ещё больше!
- И дополнительные классификаторы (просто лишние слагаемые в целевую функцию).

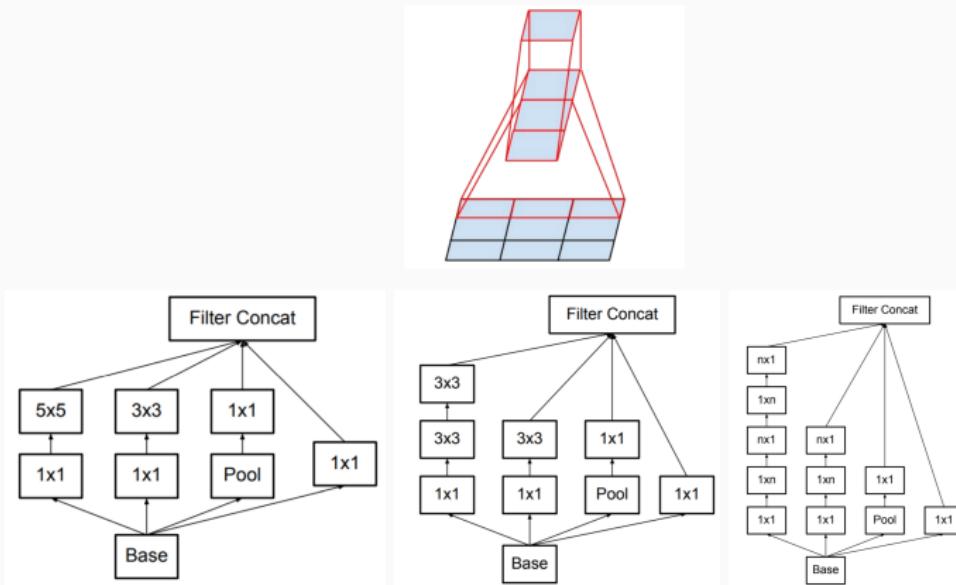


(a)

(б)

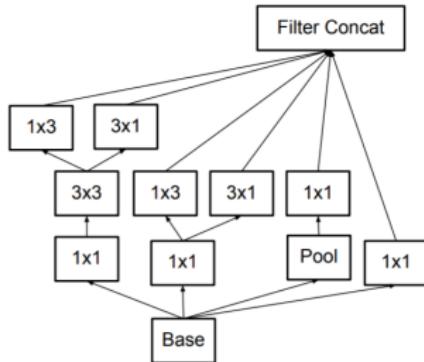
Inception

- Во второй версии (Szegedy et al., 2015) свёртки $n \times n$ заменили на комбинации свёрток $n \times 1$ и $1 \times n$:



Inception

- А банки фильтров в Inception v2 сделали широкими, а не глубокими:



- В той же статье – Inception v3; то же самое, но ещё RMSProp, факторизованные свёртки 7×7 , batchnorm во вспомогательных классификаторах и...

- Label smoothing:
 - в обычном классификаторе мы обучаем результаты softmax с целевой переменной $q(k) = [k = y]$, т.е. аргументы softmax хотят расти неограниченно;
 - модели становятся слишком уверены в себе, оверфиттинг;
 - решение: давайте вместо этого оптимизировать

$$q'(k) = (1 - \epsilon)[k = y] + \epsilon u(k)$$

для какого-то априорного $u(k)$, например $u(k) = \frac{1}{k}$.

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	18.77%	4.2%

- Остаточное обучение (residual learning): давайте обучим разности между очередным уровнем и предыдущим.
- Тогда градиенты смогут беспрепятственно проходить куда надо.
- Функция, реализуемая остаточным блоком, выглядит как

$$\mathbf{y}^{(k)} = F(\mathbf{x}^{(k)}) + \mathbf{x}^{(k)},$$

где $\mathbf{x}^{(k)}$ — входной вектор слоя k , $F(x)$ — функция, которую вычисляет слой нейронов, а $\mathbf{y}^{(k)}$ — выход остаточного блока, который потом станет входом следующего слоя $\mathbf{x}^{(k+1)}$.

- И градиент будет проходить через этот блок беспрепятственно и не будет затухать:

$$\frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{x}^{(k)}} = 1 + \frac{\partial F(\mathbf{x}^{(k)})}{\partial \mathbf{x}^{(k)}}.$$

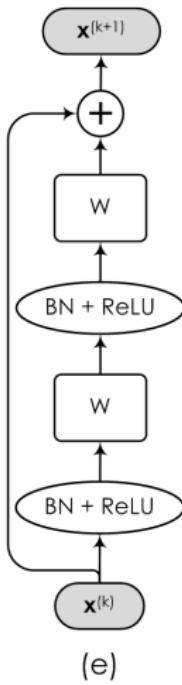
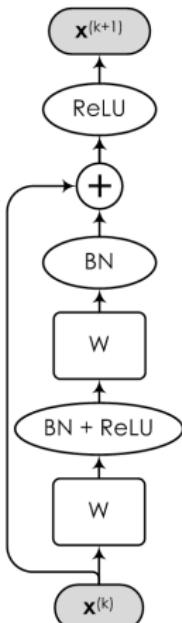
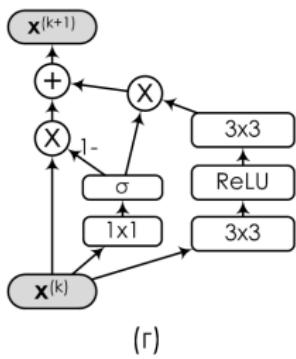
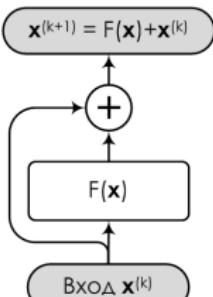
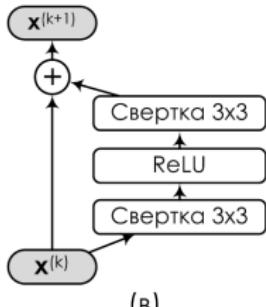
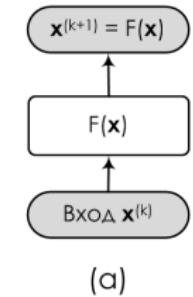
- Это привело к очень глубоким сетям.
- Другой аналогичный подход – *highway networks* (магистральные сети) Шмидхубера.
- Тоже представляем $y^{(k)}$, выход слоя k , как линейную комбинацию входа этого слоя $x^{(k)}$ и результата $F(x^{(k)})$, веса которой управляются другими преобразованиями:

$$y^{(k)} = C(x^{(k)})x^{(k)} + T(x^{(k)})F(x^{(k)}),$$

где C – это гейт переноса (carry gate), а T – гейт преобразования (transform gate); обычно комбинацию делают выпуклой, $C = 1 - T$.

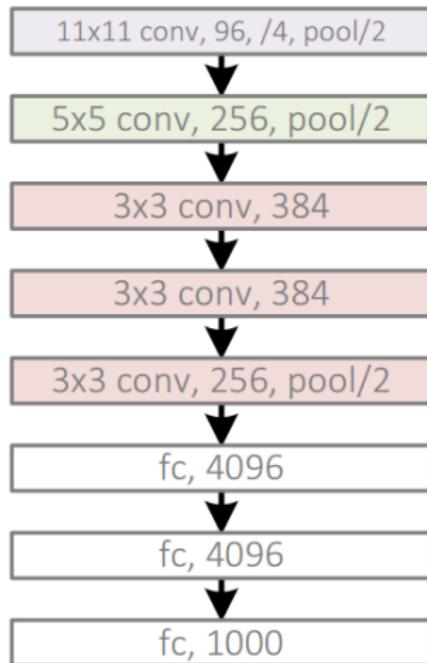
- Практика показывает, что чем «прямее», тем лучше.

ResNet: варианты



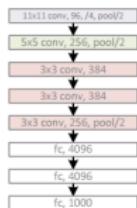
Revolution of Depth (Kaiming He)

AlexNet, 8 layers
(ILSVRC 2012)

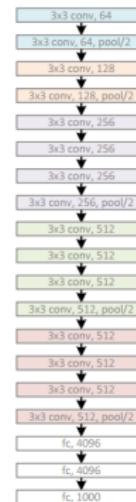


Revolution of Depth (Kaiming He)

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



Revolution of Depth (Kaiming He)

ResNet led to the revolution of depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

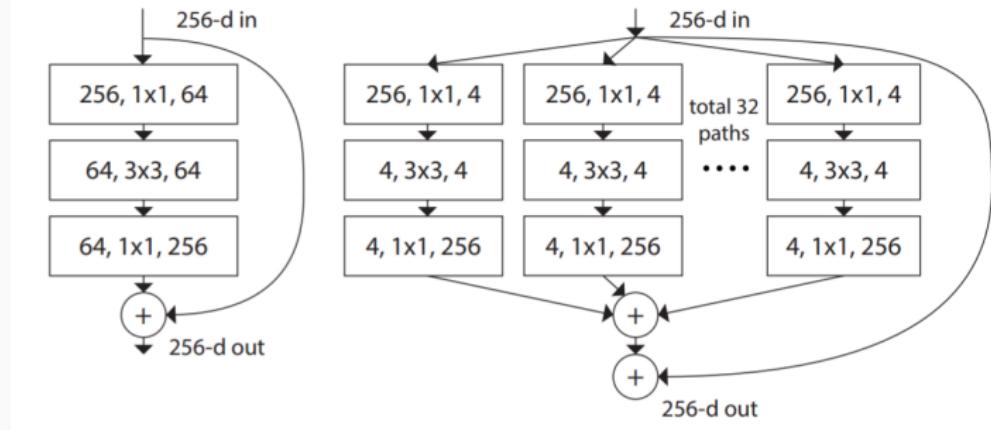


ResNet, **152 layers**
(ILSVRC 2015)



ResNeXt

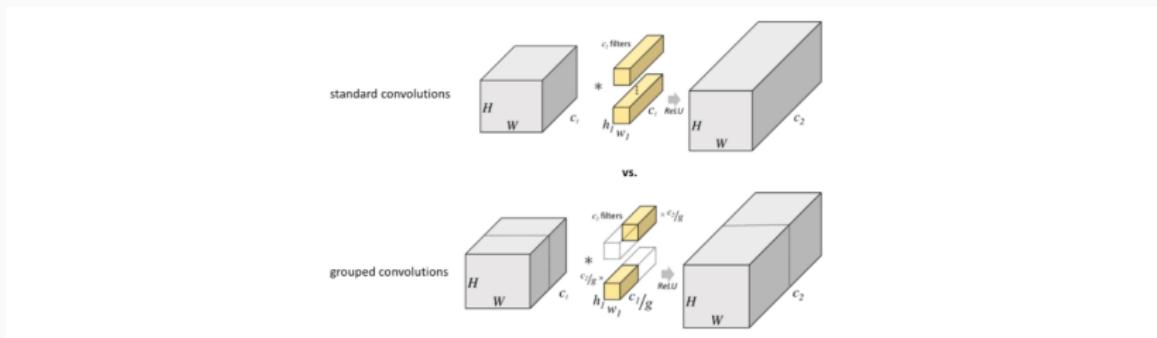
- ResNeXt (Xie et al., 2016): давайте заменим ResNet-блоки на "split-transform-merge" блоки, похожие на Inception.



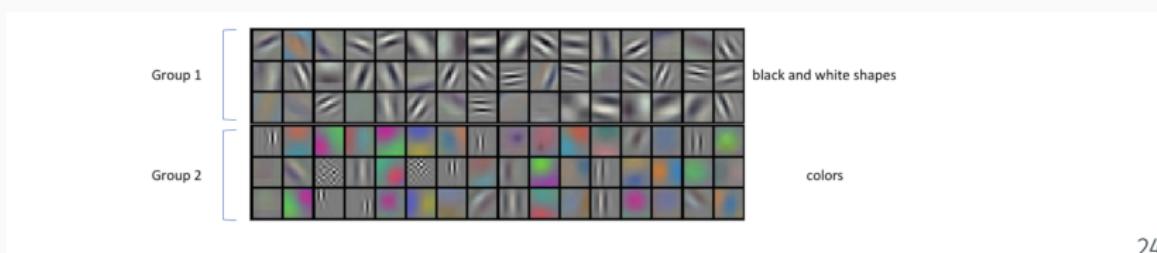
- Вход разбивается на блоки по каналам, к каждому блоку свои свёртки.

ResNeXt

- Идея похожа на group convolutions, которые были ещё в AlexNet для параллелизации:

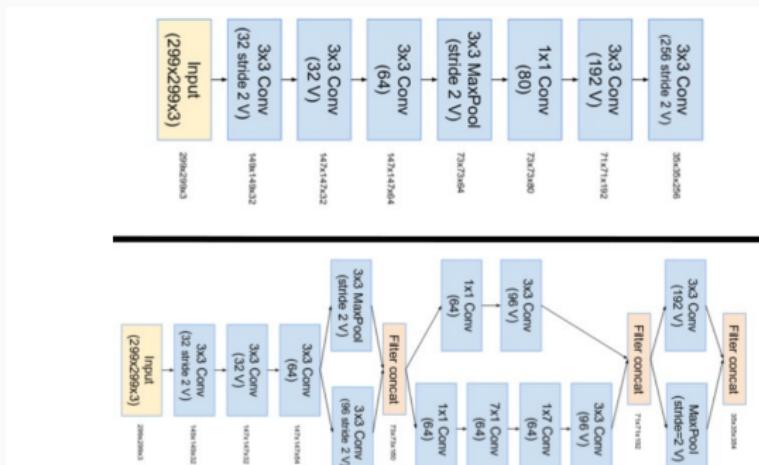


- И они дают специализацию в результатах:



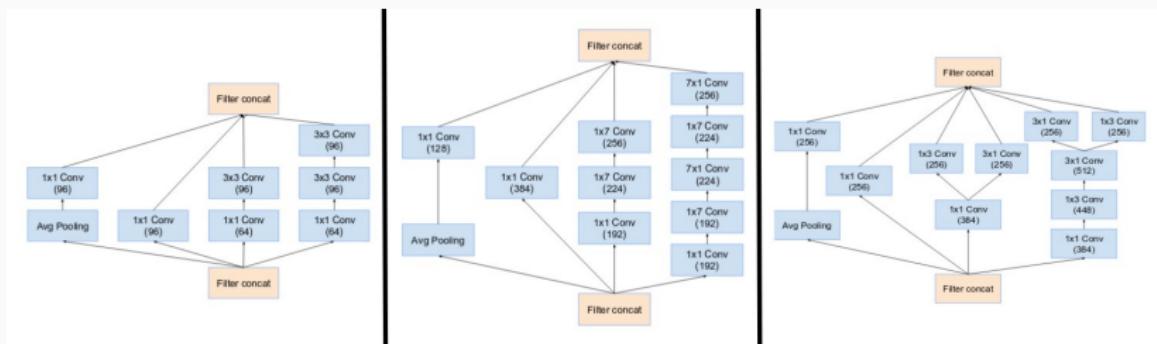
Inception v4 и Inception ResNet

- Ещё одна классическая статья (Szegedy et al., 2016): вводятся Inception v4 и Inception ResNet.
- Inception v4 – идея в том, чтобы всё стандартизировать, упростить модули. Во-первых, «членок»:



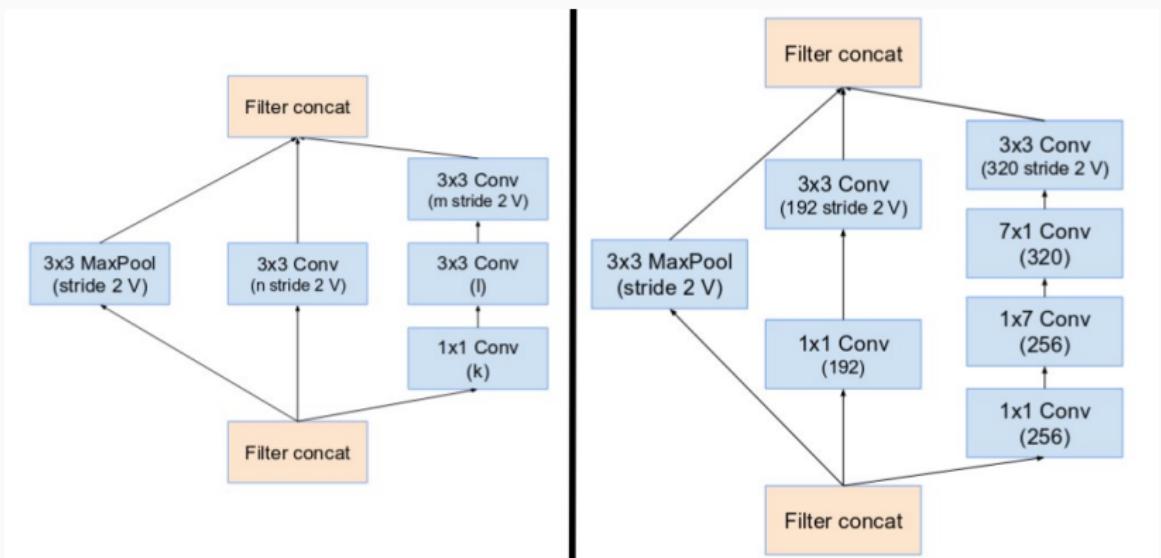
Inception v4 и Inception ResNet

- Во-вторых, теперь три базовых блока A, B, C:



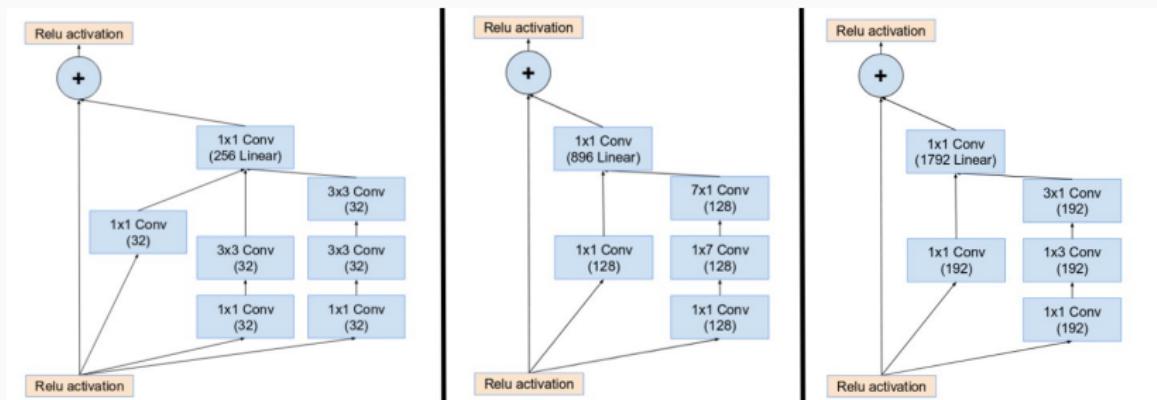
Inception v4 и Inception ResNet

- И специальные reduction blocks для сокращения размерности сетки:



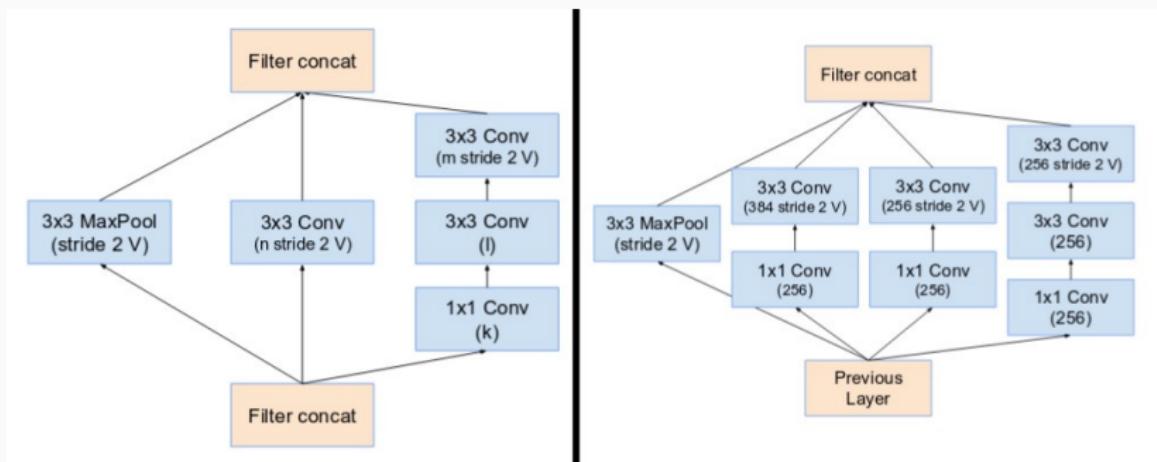
Inception v4 и Inception ResNet

- В Inception ResNet к тем же блокам добавляются остаточные связи:



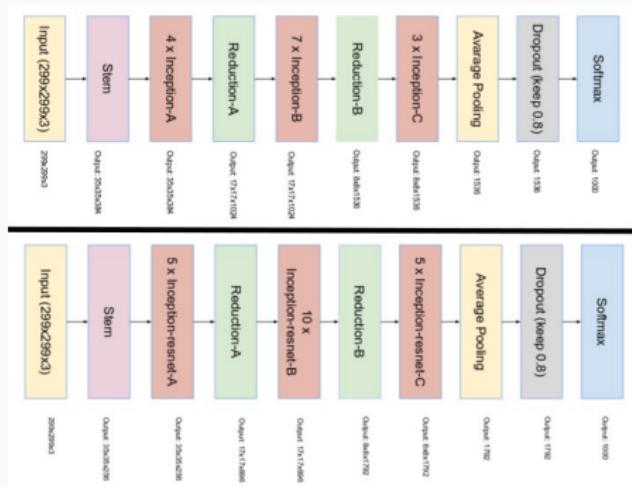
Inception v4 и Inception ResNet

- Субдискретизации теперь нет, но по-прежнему есть reduction blocks:



Inception v4 и Inception ResNet

- В итоге архитектура даже упростилась; сверху Inception v4, снизу Inception ResNet:



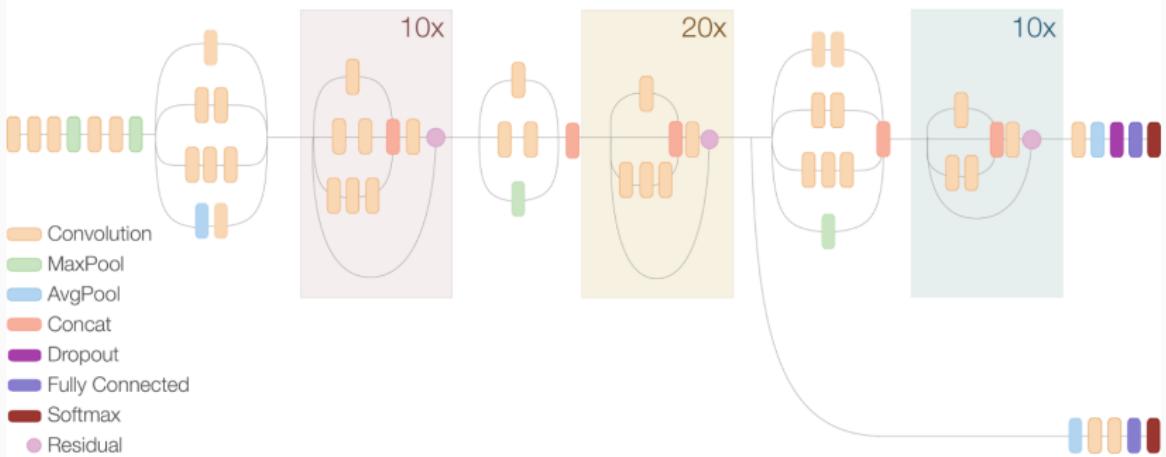
Inception v4 и Inception ResNet

- Inception ResNet v2:

Inception Resnet V2 Network



Compressed View



Inception v4 и Inception ResNet

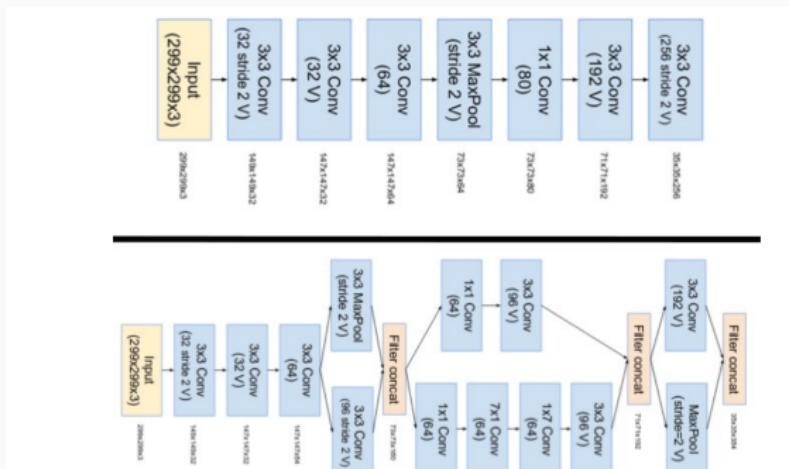
- И работает неплохо:

Network	Crops	Top-1 Error	Top-5 Error
ResNet-151 [5]	10	21.4%	5.7%
Inception-v3 [15]	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Network	Crops	Top-1 Error	Top-5 Error
ResNet-151 [5]	dense	19.4%	4.5%
Inception-v3 [15]	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

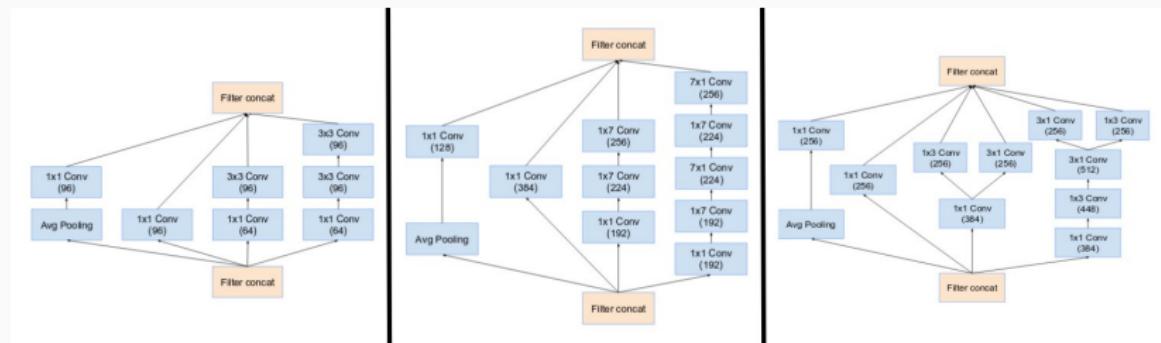
Inception v4 и Inception ResNet

- Ещё одна классическая статья (Szegedy et al., 2016): вводятся Inception v4 и Inception ResNet.
- Inception v4 – идея в том, чтобы всё стандартизировать, упростить модули. Во-первых, «членок»:



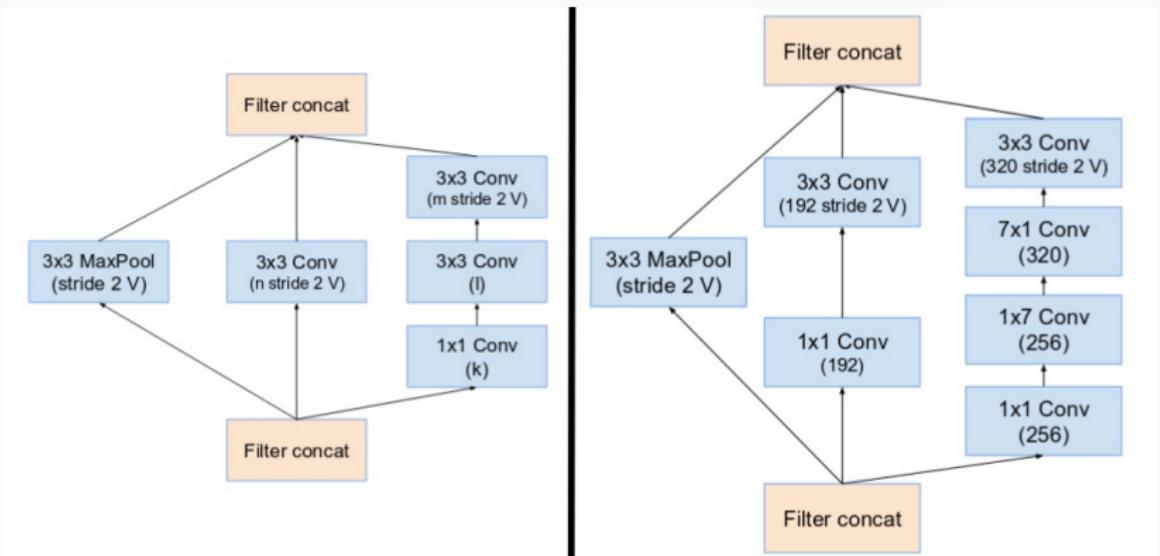
Inception v4 и Inception ResNet

- Во-вторых, теперь три базовых блока A, B, C:



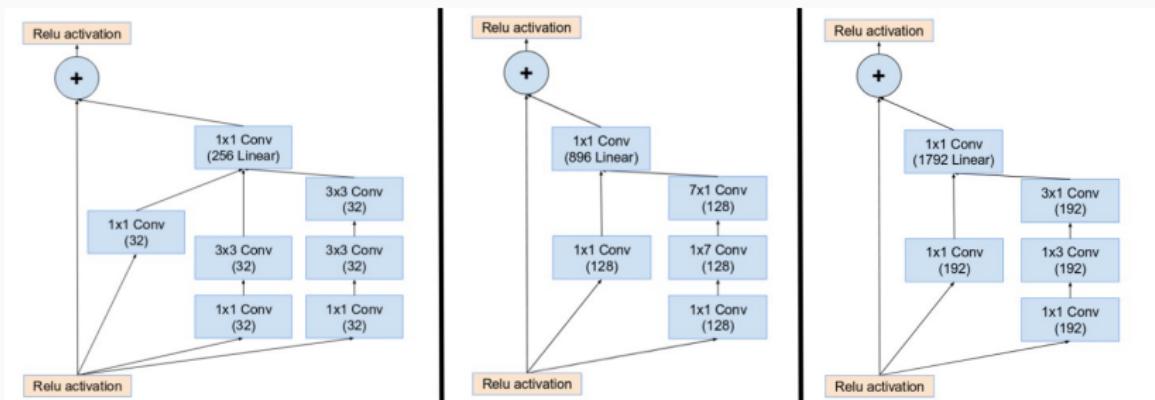
Inception v4 и Inception ResNet

- И специальные reduction blocks для сокращения размерности сетки:



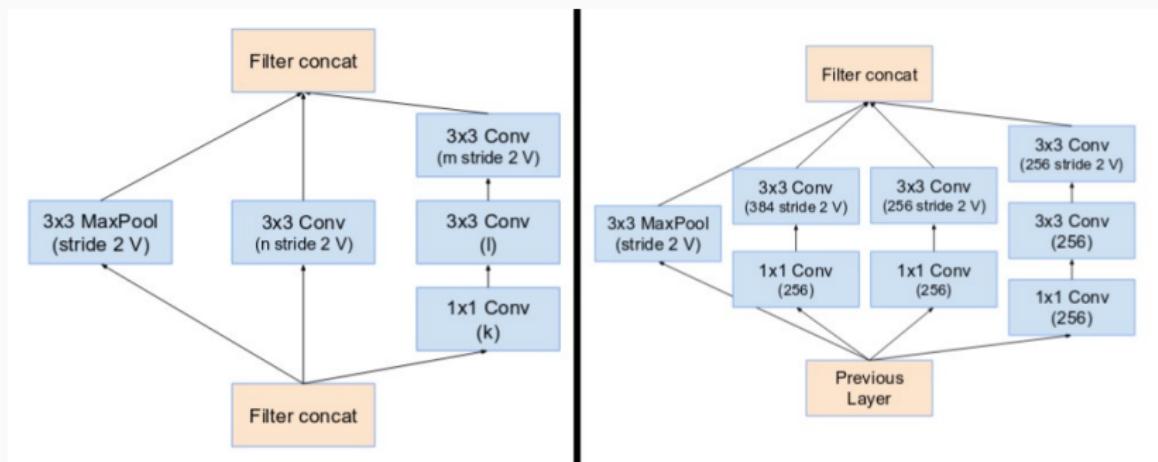
Inception v4 и Inception ResNet

- В Inception ResNet к тем же блокам добавляются остаточные связи:



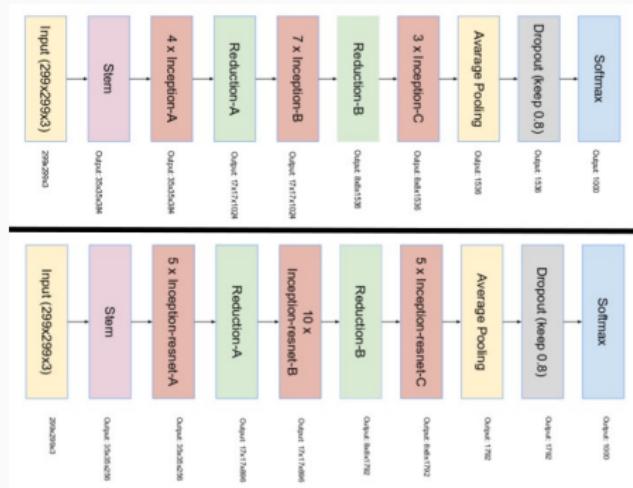
Inception v4 и Inception ResNet

- Субдискретизации теперь нет, но по-прежнему есть reduction blocks:



Inception v4 и Inception ResNet

- В итоге архитектура даже упростилась; сверху Inception v4, снизу Inception ResNet:



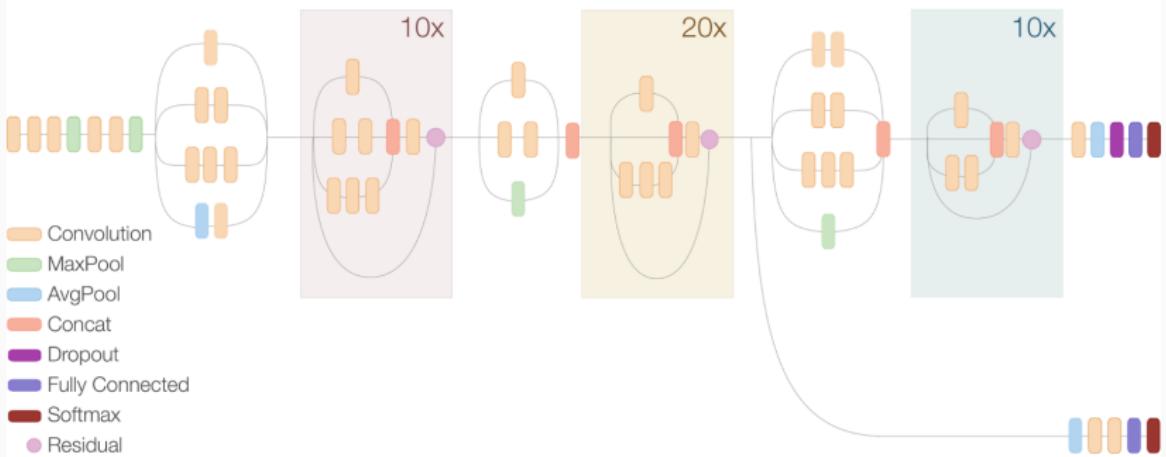
Inception v4 и Inception ResNet

- Inception ResNet v2:

Inception Resnet V2 Network



Compressed View



Inception v4 и Inception ResNet

- И работает неплохо:

Network	Crops	Top-1 Error	Top-5 Error
ResNet-151 [5]	10	21.4%	5.7%
Inception-v3 [15]	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Network	Crops	Top-1 Error	Top-5 Error
ResNet-151 [5]	dense	19.4%	4.5%
Inception-v3 [15]	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

Спасибо!

Спасибо за внимание!

