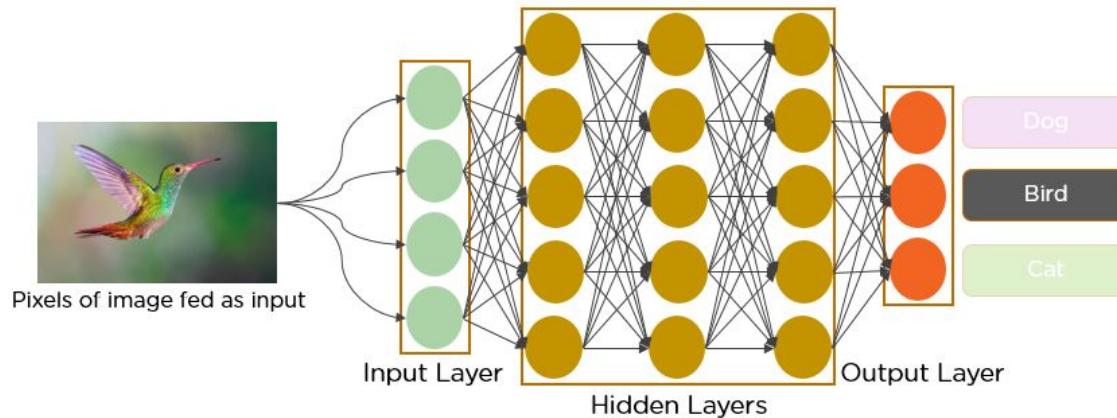


CNNs

FCNs

Базовое решение:

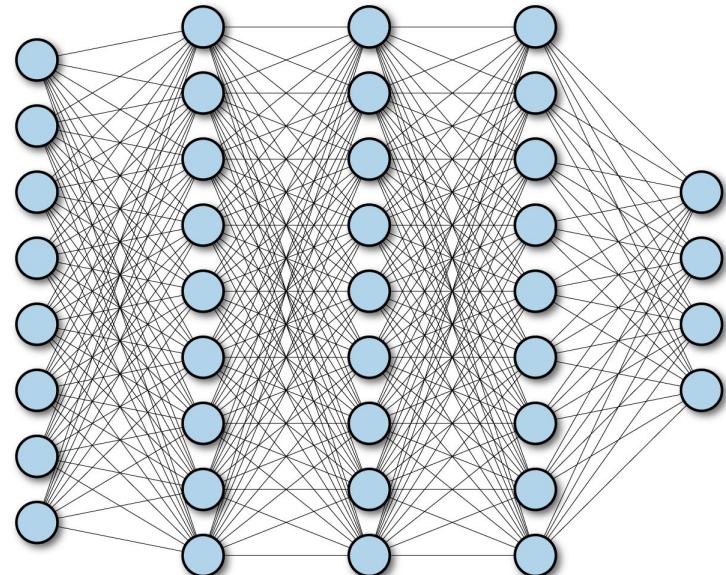
- изображение в вектор
- применить полносвязную нейронку (fully connected neural network, FCN)



source: <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>

FCNs

Проблема: очень много параметров



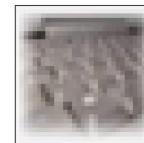
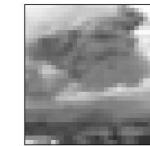
source: <https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html>

FCNs

Проблема: очень много параметров

- Cifar с разрешением 32x32

1 нейрон: $32 \times 32 \times 3 = 3072$ параметров



FCNs

Проблема: очень много параметров

- Cifar с разрешением 32x32

1 нейрон: $32 \times 32 \times 3 = 3072$ параметров

- ImageNet 256x256

1 нейрон: $2^8 \times 2^8 \times 3 \approx 200k$ параметров



n02097047 (196)



n01682714 (40)



n03134739 (522)



n04254777 (806)



n02859443 (449)



n02096177 (192)



n02107683 (239)



n01443537 (1)



n02264363 (318)

FCNs

Проблема: очень много параметров

- Cifar с разрешением 32x32

1 нейрон: $32 \times 32 \times 3 = 3072$ параметров

- ImageNet 256x256

1 нейрон: $2^8 \times 2^8 \times 3 \approx 200k$ параметров

А нужно >> 1 !



FCNs

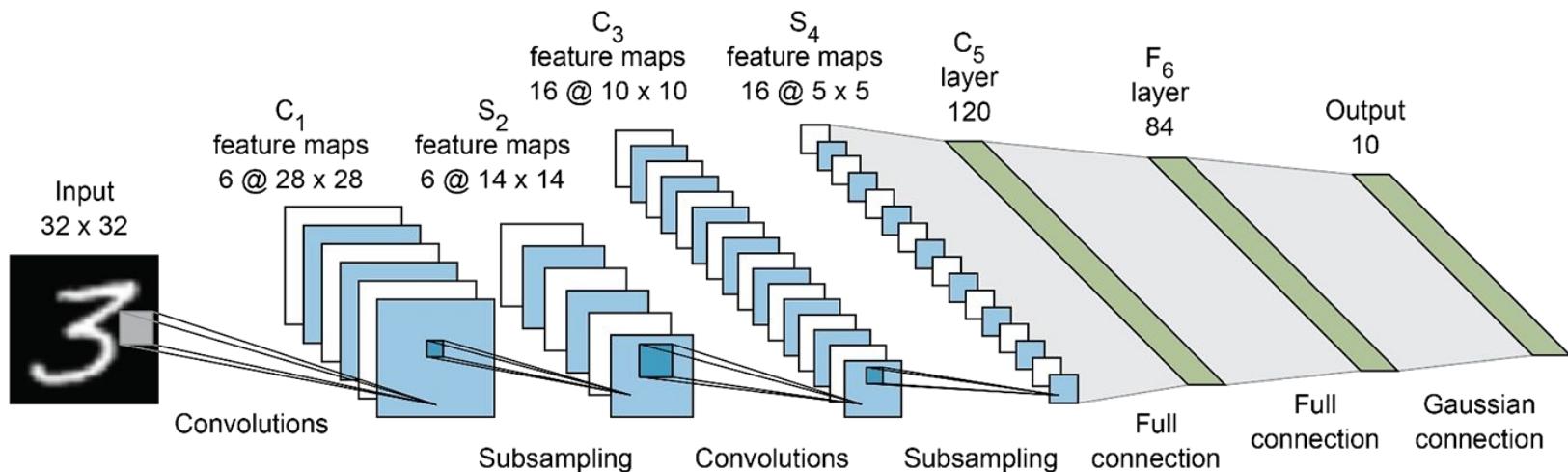
Проблема: очень много параметров

- много ресурсов (память, компьютер)
- оверфиттинг

CNNs

Решение проблемы: переиспользование параметров (parameter sharing).

Конкретнее: сверточные нейронные сети
(convolutional neural networks, CNNs).



Основные слои

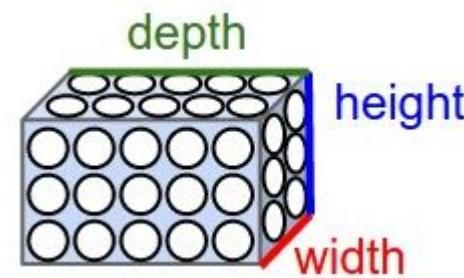
CNNs

Основные виды слоев:

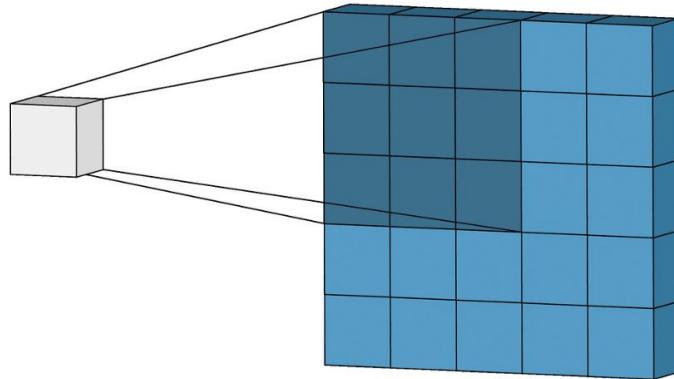
- сверточные слои (convolutional layer)
 - пулинговые слои (pooling layer)
 - полносвязные слои (fully-connected layer)
 - функции активации (activation function)
= нелинейность (nonlinearity)
-

Словарь

- тензор = массив
 - входное изображение
 - выход каждого слоя
- измерения тензора
 - пространственные: высота H , ширина W
 - каналы: глубина C
- карта признаков (feature map) - матрица значений определенного канала



2d свертка

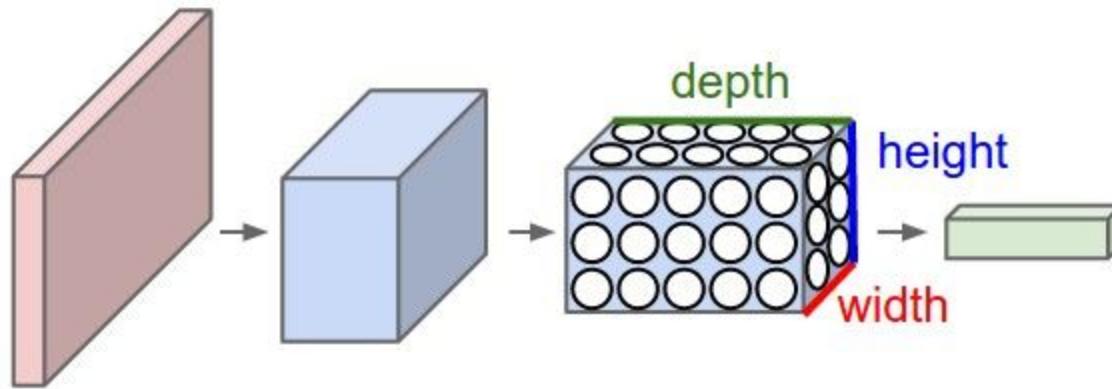


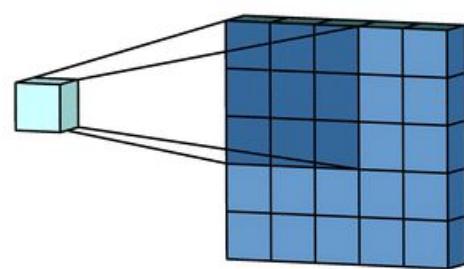
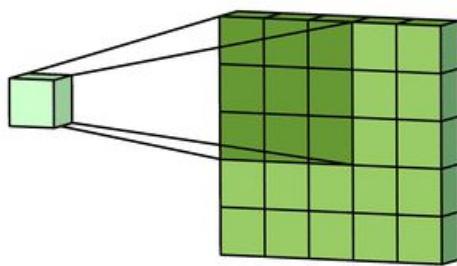
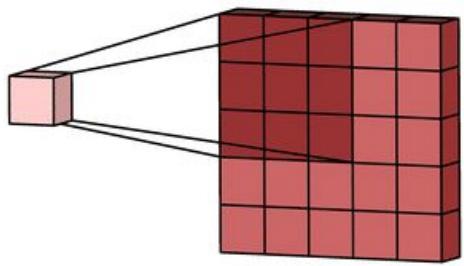
2d свертка

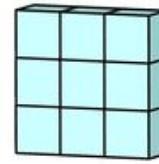
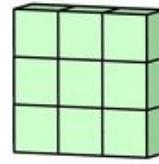
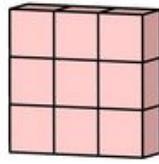
3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

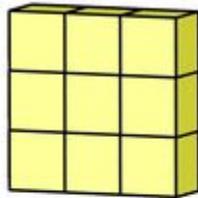
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

2d свертка







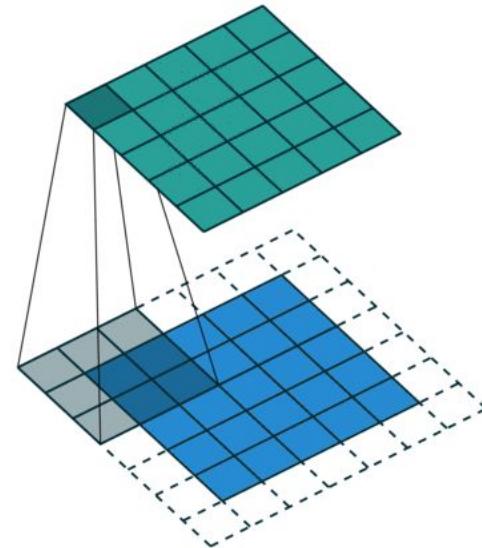


Гиперпараметры свертки

- размер ядра (kernel size)

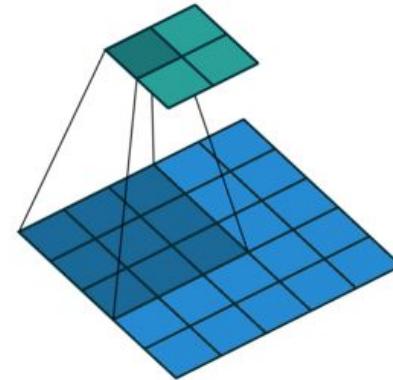
Гиперпараметры свертки

- размер ядра (kernel size)
- отступ (padding)



Гиперпараметры свертки

- размер ядра (kernel size)
- отступ (padding)
- шаг свертки (stride)



Формула

$$y_{i,j} = \sum_{1 \leq c \leq C} \left(\sum_{-k \leq a,b \leq k} W_{a,b}^c x_{i+a,j+b}^c \right)$$

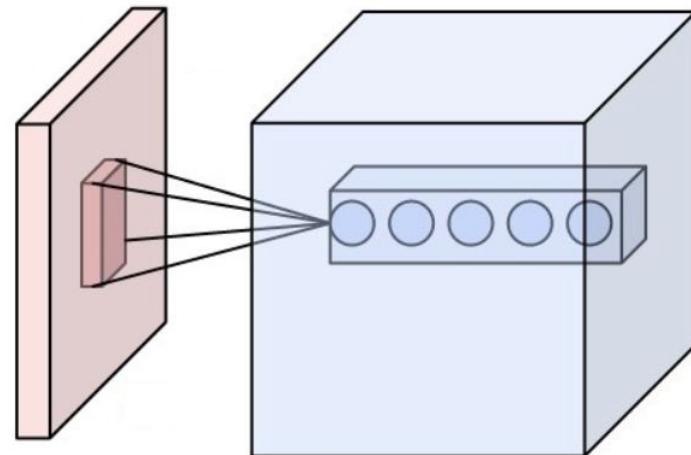
Receptive Field

Рецептивное поле (receptive field) нейрона

– нейроны предыдущего слоя, влияющие на его значение

Иерархическая структура

- увеличение рецептивного поля
- нижние слои - локальные признаки
верхние слои - глобальные признаки



Сверточный слой

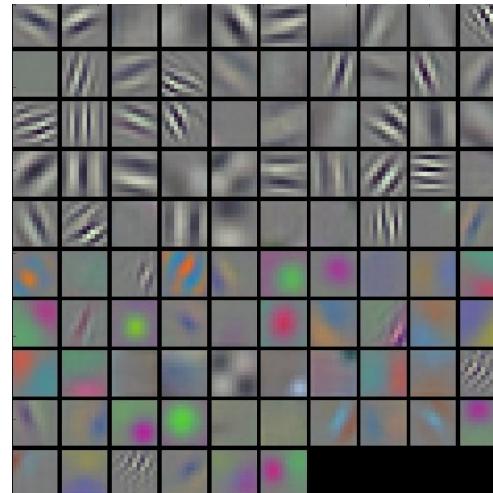
Сверточный слой (convolutional layer) - несколько сверток (фильтров)

Гиперпараметры:

- количество фильтров
- гиперпараметры свертки

Классические фильтры

- Операция свертки похожа на применение классических фильтров
- Только нейронка сама выучивает фильтр
- При обучении на больших данных на нижних слоях что-то похоже на разработанные учеными и инженерами фильтры



Преимущества свертки

- работа с данными переменного размера
- разреженные связи
 - ⇒ меньше параметров
 - ⇒ надо меньше ресурсов (памяти, компьютера)
- а также меньше параметров ⇒ меньше оверфиттинга
- переиспользование весов
 - ⇒ поиск универсальных признаков
 - ⇒ больше обобщаемость

Pooling

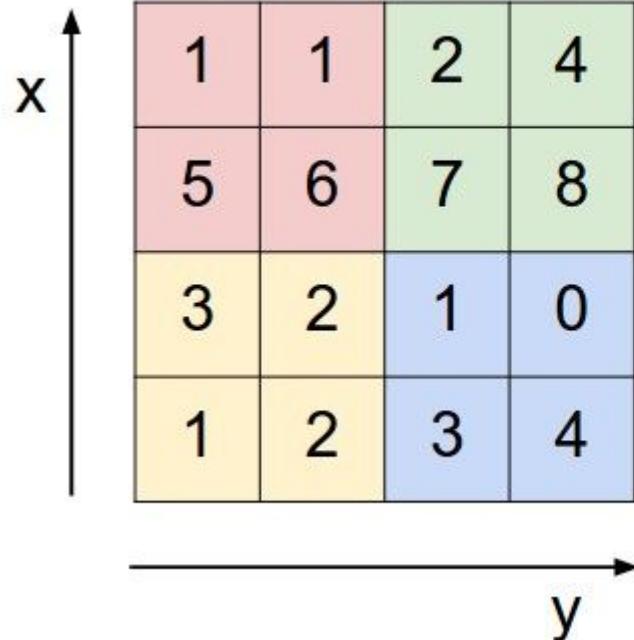
Пулинг (pooling) = субдискретизация (subsampling)

– побочная агрегация элементов тензора с помощью некоторой функции:

- функция максимума (max pooling)
- функция среднего (average pooling)
- функция взвешенного среднего (weighted average pooling)

Note: нет обучаемых параметров

Single depth slice



max pool with 2x2 filters
and stride 2

A 2x2 grid representing the output of a max pooling operation with 2x2 filters and stride 2. The values are:

6	8
3	4

Pooling

Гиперпараметры:

- размер ядра (kernel size) F
- шаг пулинга (stride) S

Обычно $F=2$, $S=2$

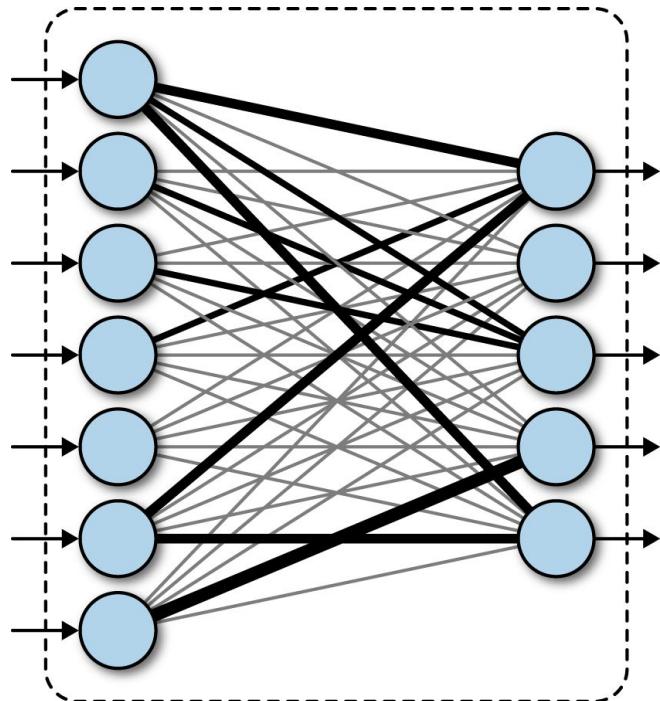
Pooling

Цели:

- уменьшение изображения для увеличения receptive поля последующих сверточ
- увеличение инвариантности выхода сети по отношению к малому переносу входа
- уменьшение изображения для экономии ресурсов (память, компьютер)
- (все это без добавления параметров)

FC

Полносвязный слой (fully-connected layer)

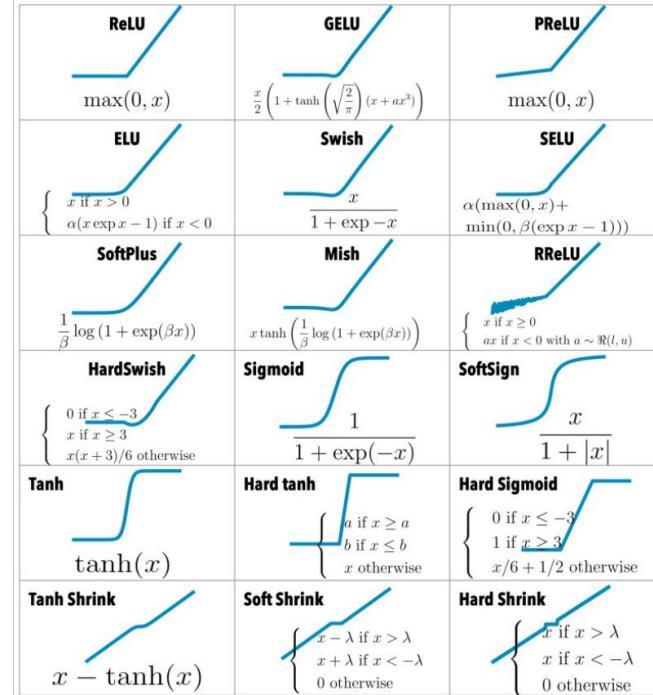


source: <https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html>

ФУНКЦИИ АКТИВАЦИИ

- ПОПУЛЯРНЫЕ
 - ReLU
 - LeakyReLU
- КЛАССИКА
 - tanh
 - sigmoid
- ПУБЛИКАЦИИ
 - GELU
 - SELU
 - Mish
 - Swish
 - ...

Neural Network Activation Functions: a small subset!

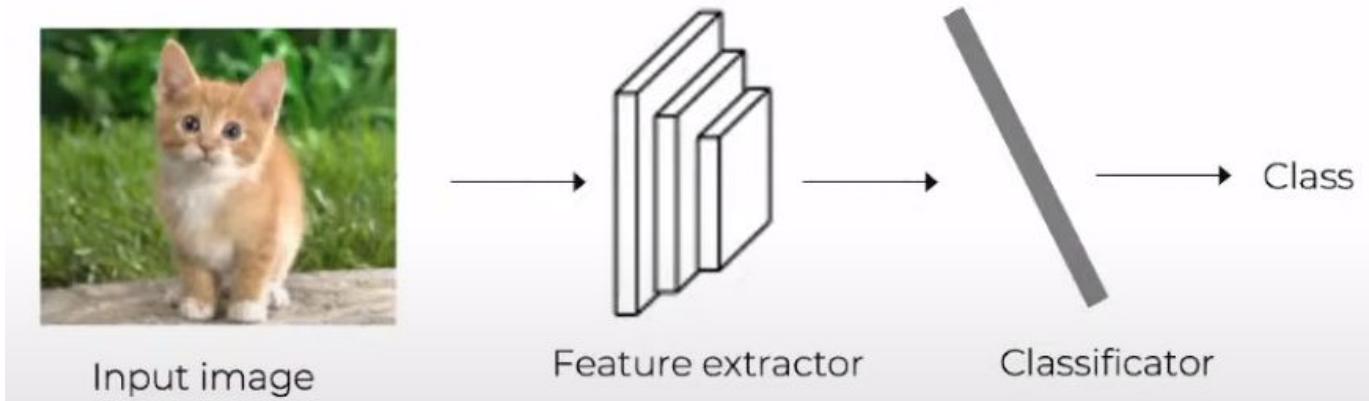


Сверточный блок

- conv → activation → pooling
- (conv → activation) → ... → (conv → activation) → pooling

Архитектура сверточных нейронных сетей

- Энкодер (encoder)
 - несколько сверточных блоков
- классификатор (classifier) = голова (head)
 - полносвязные слои
 - функции активации



Обучение

- Обучающая, валидационная, тестовая выборки данных
- Функция потерь
- Градиентный спуск
- Обратное распространение ошибки

Дополнительные слои

Вариации свертки

- 1x1 свертки
- depthwise separable convolutions
- dilated convolutions
- ...

1x1 convolution

- для изменения количества каналов

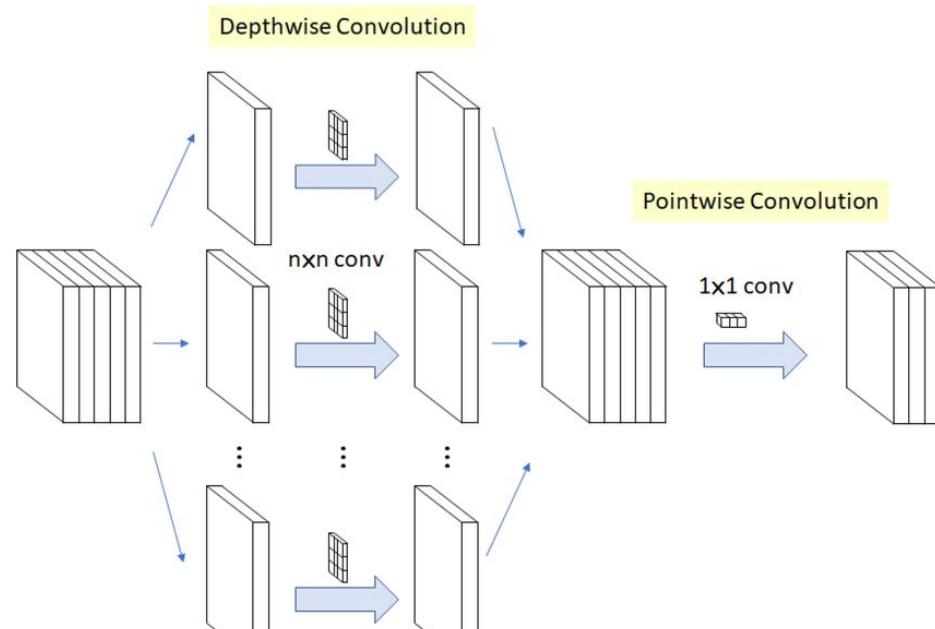
Depthwise Separable Convolution

Разбиваем свертку на 2 части:

- depthwise convolution:
по 1 фильтру на канал
- pointwise convolution:
 1×1 свертка

Зачем?

- меньше параметров
- разделение поиска
пространственных и
канальных зависимостей



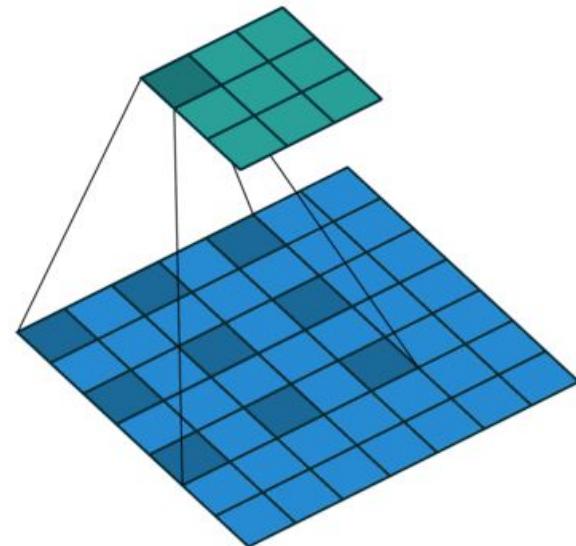
source: <https://www.analyticsvidhya.com/blog/2021/11/an-introduction-to-separable-convolutions/>

Dilated Convolutions

dilated convolutions = atrous convolutions

Зачем?

- больше receptive field
- то же количество параметров



Дополнительные слои

- Dropout
- Normalization layers

Dropout

Дропаут (Dropout layer)

Dropout

Дропаут (Dropout layer)

Что делает?

Зануляет каждый нейрон предыдущего слоя с вероятностью p

Dropout

Дропаут (Dropout layer)

Что делает?

Зануляет каждый нейрон предыдущего слоя с вероятностью p

Цель?

Регуляризация

Dropout

Дропаут (Dropout layer)

Что делает?

Зануляет каждый нейрон предыдущего слоя с вероятностью p

Цель?

Регуляризация

Почему работает?

Нейроны следующего слоя не могут положиться ни на какую конкретную фичу => более равномерное распределение внимания (весов)

Как работает Dropout

Во время обучения:

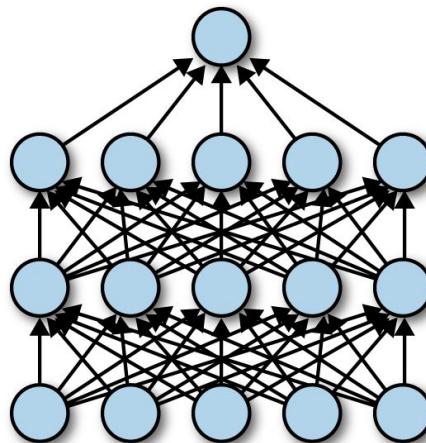
- зануляет выход каждого из нейронов предыдущего слоя с вероятностью p
- умножает все выходы на $1/(1-p)$

Зачем умножение:

чтобы сохранить матожидание

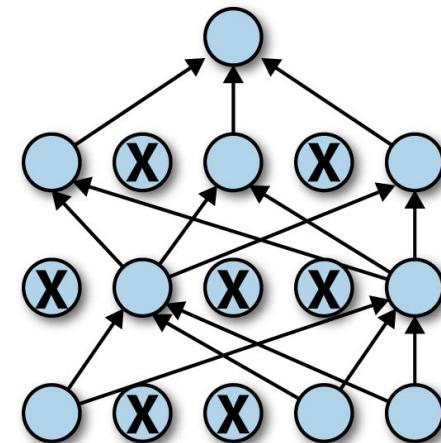
Note:

градиенты



(a) Standard Neural Net

source: <https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html>



(b) After applying dropout

Как работает Dropout

Во время инференса: действует как тождественное преобразование

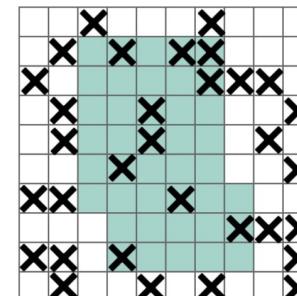
Structured Dropout

Структурированные модификации дропаута:

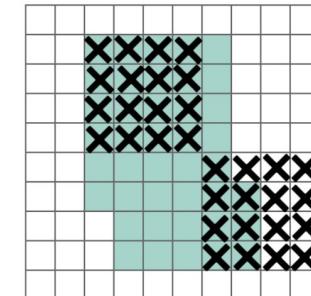
- DropBlock [paper](#)
- DropChannel [paper](#)
- DropPath [paper](#)



(a)



(b)



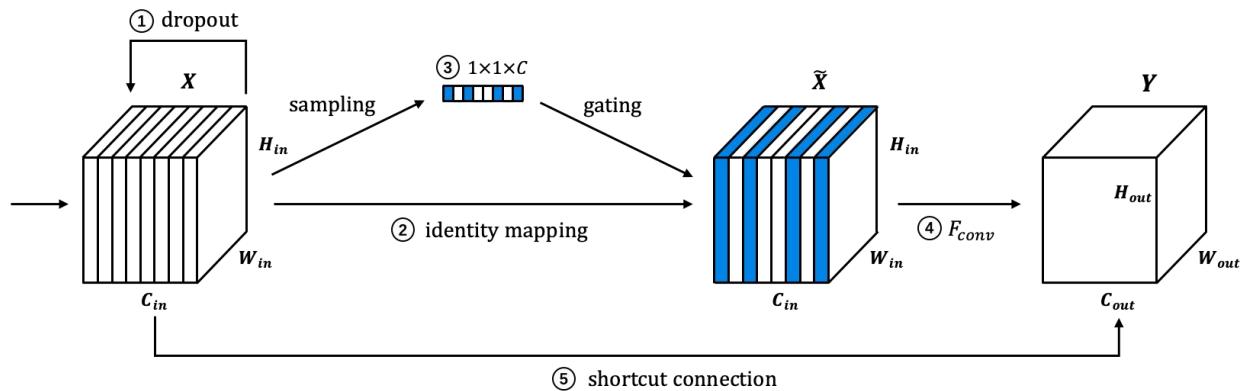
(c)

Figure 1: (a) input image to a convolutional neural network. The green regions in (b) and (c) include the activation units which contain semantic information in the input image. Dropping out activations at random is not effective in removing semantic information because nearby activations contain closely related information. Instead, dropping continuous regions can remove certain semantic information (e.g., head or feet) and consequently enforcing remaining units to learn features for classifying input image.

Structured Dropout

Структурированные модификации дропаута:

- DropBlock [paper](#)
- DropChannel [paper](#)
- DropPath [paper](#)



source: <https://github.com/oilbc88/dropout>

Structured Dropout

Структурированные модификации дропаута:

- DropBlock [paper](#)
- DropChannel [paper](#)
- DropPath [paper](#)

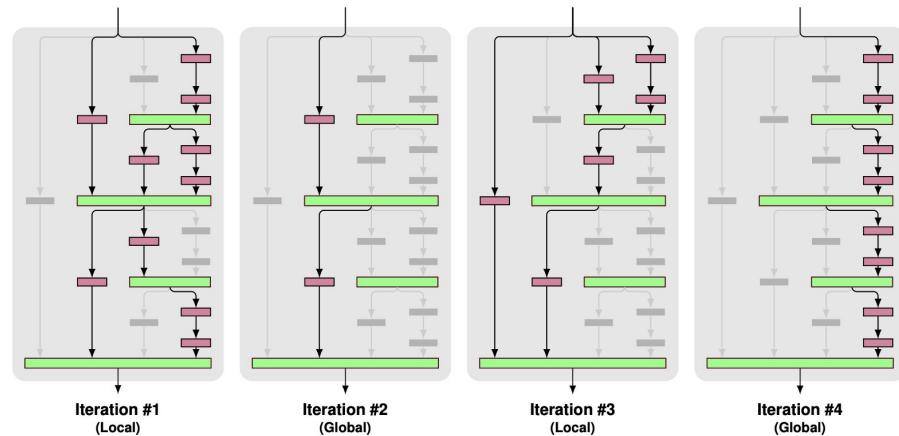


Figure 2: **Drop-path.** A fractal network block functions with some connections between layers disabled, provided some path from input to output is still available. Drop-path guarantees at least one such path, while sampling a subnetwork with many other paths disabled. During training, presenting a different active subnetwork to each mini-batch prevents co-adaptation of parallel paths. A global sampling strategy returns a single column as a subnetwork. Alternating it with local sampling encourages the development of individual columns as performant stand-alone subnetworks.

source: <https://paperswithcode.com/method/droppath>

Normalization layers

- BatchNorm
- LayerNorm
- InstanceNorm
- GroupNorm

BatchNorm

Батчнорм (BatchNorm layer)

Что делает:

Приводит статистики выходов слоя (матожидание, дисперсия) к желаемым значениям

Зачем нужен:

Внутренний ковариантный сдвиг (internal covariate shift)

BatchNorm

Эффекты батчнорма:

- каждый слой обучается более независимо от других
- разные фичи имеют примерно одинаковый масштаб
- масштаб настраиваемый

Плюсы батчнорма:

- более быстрая сходимость обучения
- можно использовать более высокие lr

Незначительно:

- добавляет шум => регуляризационный эффект

BatchNorm

2 части

- нормализация
- scale & shift

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

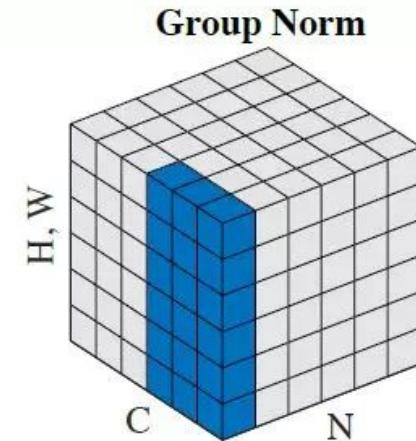
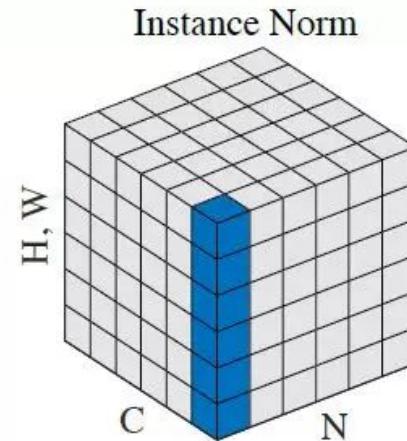
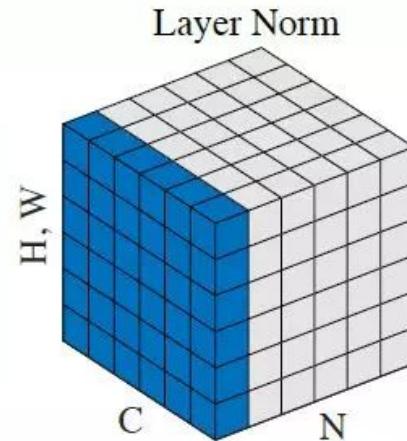
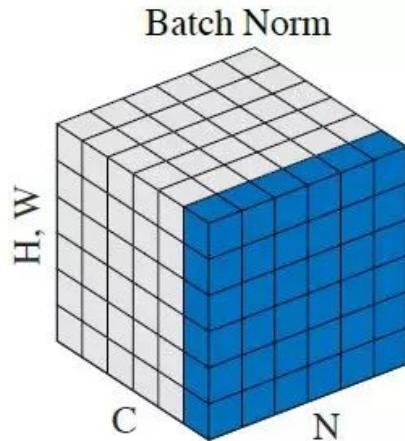
BatchNorm

Статистики для нормализации

- обучение: статистики текущего батча
- инференс: усредненное значение статистик с обучения (исп. moving average)

Normalization layers

- BatchNorm
- LayerNorm
- InstanceNorm
- GroupNorm



Важные архитектуры

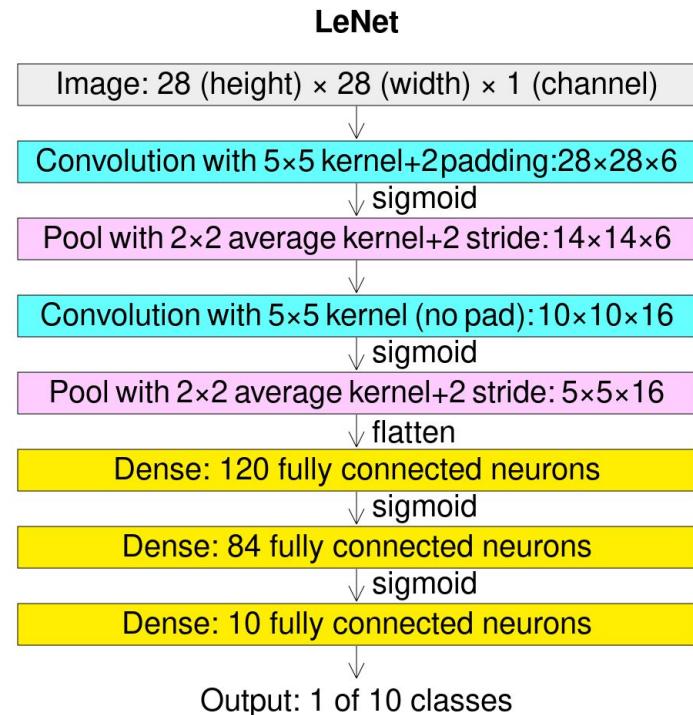
Timeline

- 1989 LeNet
- 2012 AlexNet
- 2014 VGG
- 2014 Inception (GoogleNet)
- 2015 ResNet
- 2018 NASNet
- ...

LeNet (1989)

Yann LeCun et al.

- MNIST: рукописные цифры
70k штук, grayscale, 28x28
- демонстрация успешности CNNs
- базовые компоненты: conv, pool, FC
- на самом деле, целое семейство
 - 1989 LeNet-1
 - ...
 - 1994 LeNet-4
 - 1995 LeNet-5

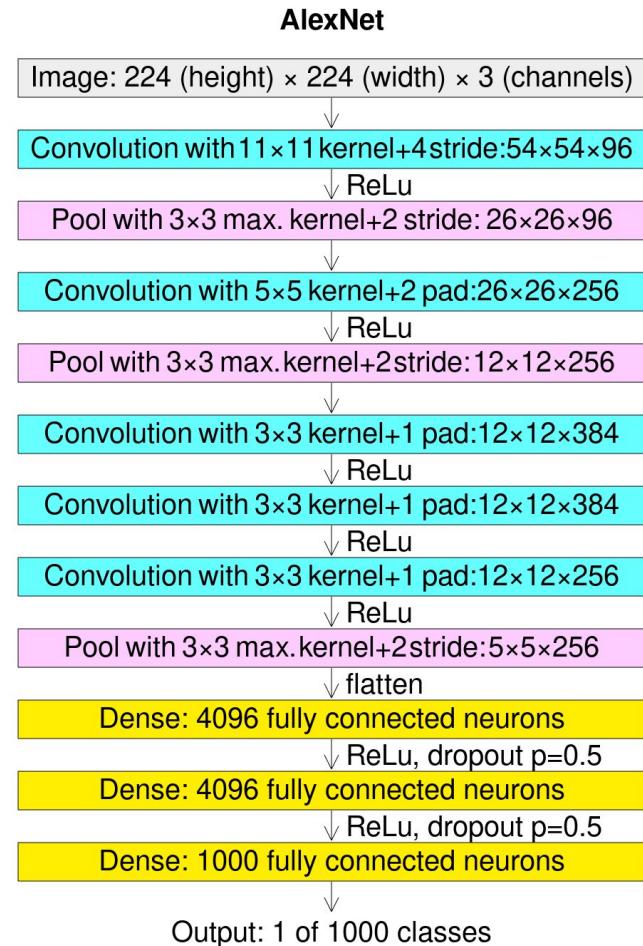


source: <https://en.wikipedia.org/wiki/LeNet>

AlexNet (2012)

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton [paper](#)

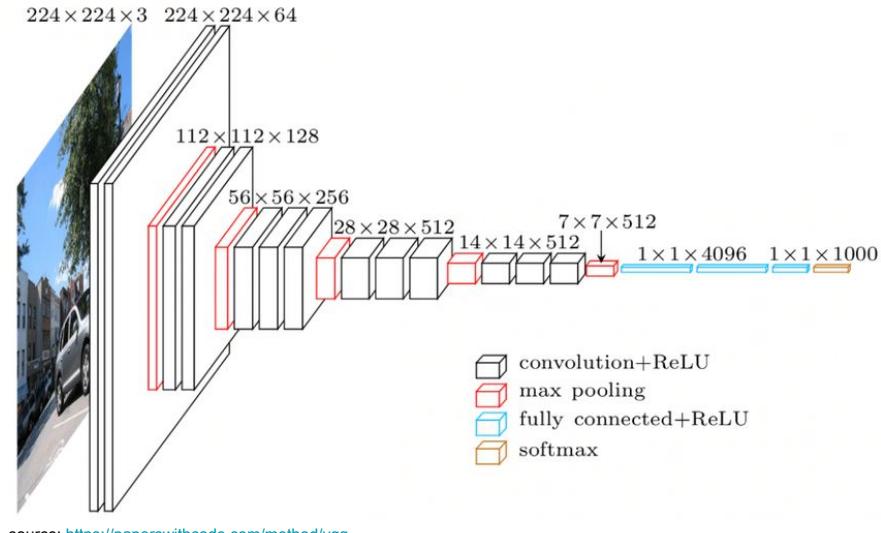
- ImageNet: естественные изображения
1M штук, rgb, 224x224
- более глубокая
- базовые компоненты: conv, pool, FC, **ReLU**, **dropout**
- *быстрое обучение на GPU*



VGG (2014)

Karen Simonyan, Andrew Zisserman [paper](#)

- блочная архитектура



VGG (2014)

Karen Simonyan, Andrew Zisserman [paper](#)

- блочная архитектура
- уменьшили ядра всех сверток до 3x3

	7x7	3 по 3x3
receptive field	7x7	7x7
#parameters	49	27
#layers	1	3

VGG (2014)

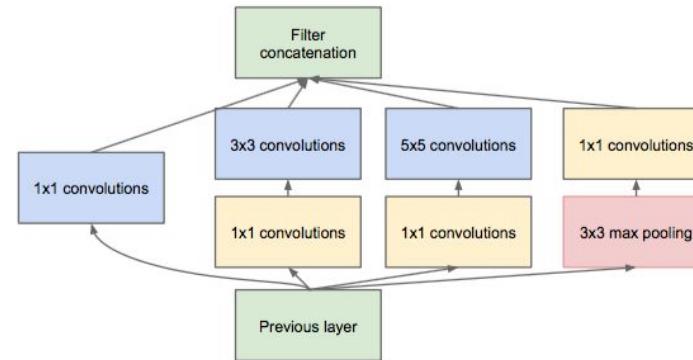
Karen Simonyan, Andrew Zisserman [paper](#)

- блочная архитектура
- уменьшили ядра всех сверток до 3x3
 - глубже
 - меньше параметров

Inception-v1 (2014)

Christian Szegedy et al [paper](#)

- не столько более глубокая, сколько более широкая
- Inception block
 - multiple paths (branches)
 - 1x1 convs

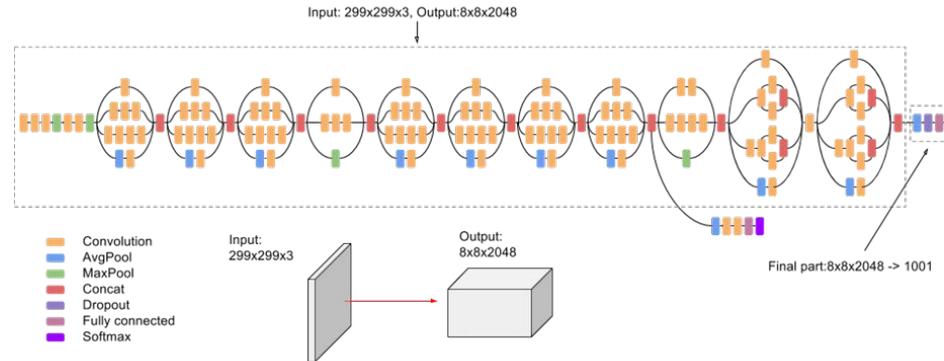


(b) Inception module with dimension reductions

Inception-v1 (2014)

Christian Szegedy et al [paper](#)

- не столько более глубокая, сколько более широкая
- Inception block
 - multiple paths (branches)
 - 1x1 convs
- доп. классификаторы



Inception-v1 (2014)

Christian Szegedy et al [paper](#)

- не столько более глубокая, сколько более широкая
- Inception block
 - multiple paths (branches)
 - 1x1 convs
- дополнительные классификаторы
- BatchNorm

ResNet (2015)

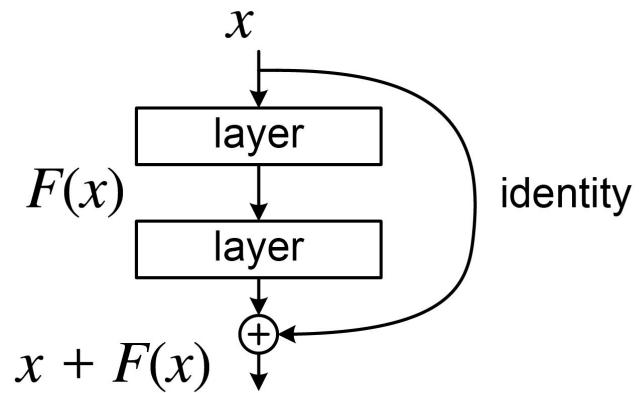
Kaiming He et al. [paper](#)

- остаточные сети (residual networks)
- можно обучать очень глубокие сети
- превзошли человека на ImageNet

ResNet (2015)

остаточный блок (residual block)

- skip connection
- residual path



ResNet (2015)

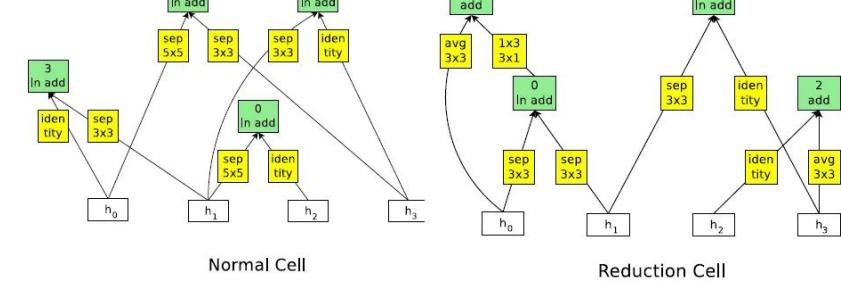
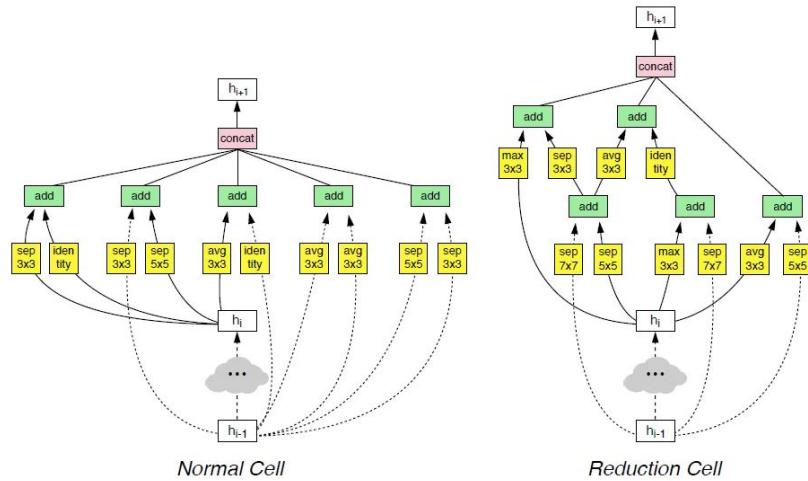
Почему работает?

1. Предотвращает затухание градиентов в глубоких сетях
2. Легко выучивает тождественную функцию => нахождение удачных [под] сетей меньшего размера

NASNet (2018)

Barret Zoph et al. [paper](#)

NAS = Neural Architecture Search:
использование AutoML для выбора структуры сети



Практическое

Железо

Для обучения и инференса используйте GPU или TPU (а не CPU)

Transfer Learning

Варианты обучения: обучение с нуля

1. Обучить модель на своем небольшом наборе данных

Transfer Learning

Варианты обучения: перенос обучения (transfer learning)

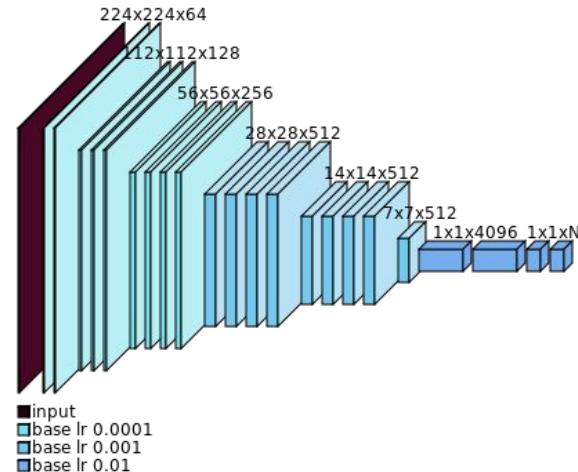
1. Обучить на большом наборе посторонних данных (напр., ImageNet)
2. Потьюнить (finetune) на своем небольшом наборе данных

Варианты тьюнинга

- A. заморозить энкодер, обучать свою голову
- B. обучать все целиком с константным learning rate
- C. использовать разные значения learning rate на разных группах слоев (на нижних - меньше, на верхних - больше)

Качество: A < B < C

Ресурсы: A << B, C



Аугментации

Аугментация данных - добавление в (обучающую) выборку
модифицированных данных

Зачем?

Снижает риск оверфиттинга

Базовые аугментации

- Цветовые
- Геометрические
- Шум



Микс

- [mixup](#)
- [CutMix](#)



Автоматический подбор аугментаций

Использование AutoML для подбора оптимальных аугментаций

- [AutoAugment](#)
- [RandAugment](#)
- [AugMix](#)
- [TrivialAugment](#)

Сравнение by Sebastian Raschka: [post](#)

Спасибо за внимание!