

CNNs

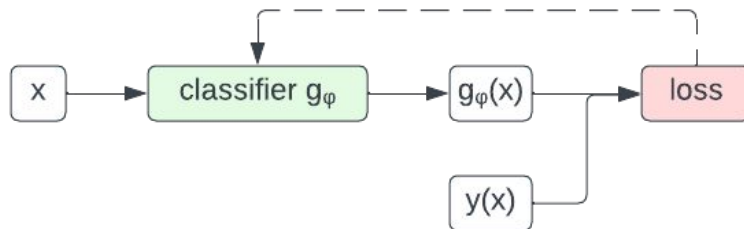
ML pipeline

Компоненты задачи распознавания в классическом ML:

- пространство объектов X (картинки какого-то домена)
- пространство ответов Y (классы, сегментационные маски, ...)
- $y: X \rightarrow Y$ - истинная зависимость
- набор данных $D = \{x^m, y^m\}_{m=1..M}$, где $y^m = y(x^m)$

Классическое решение задачи:

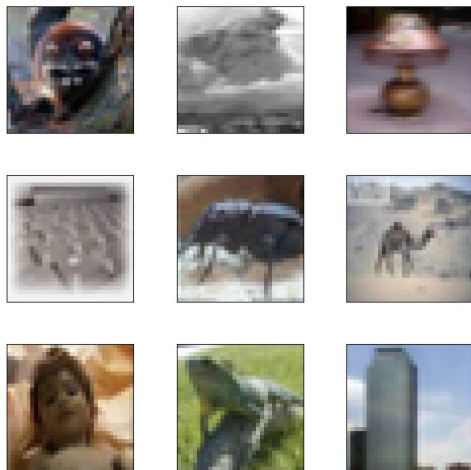
- берем значения пикселей картинки в качестве признакового описания картинки
- применяем модель ML
- обучаем с помощью функции потерь и SGD



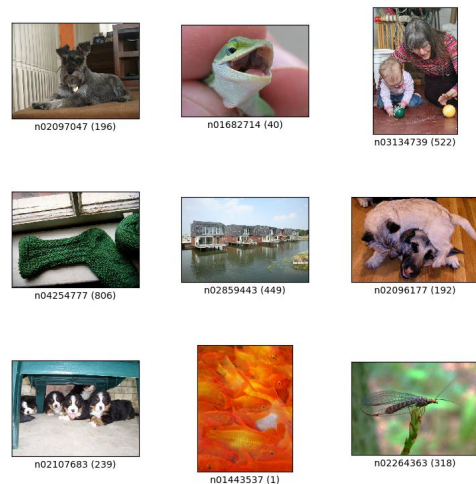
Размерность картинок

Картинки - матрицы высокой размерности: $H \times W \times C$

Cifar-10: $32 \times 32 \times 3 = 3072$



ImageNet $256 \times 256 \times 3 \approx 200k$

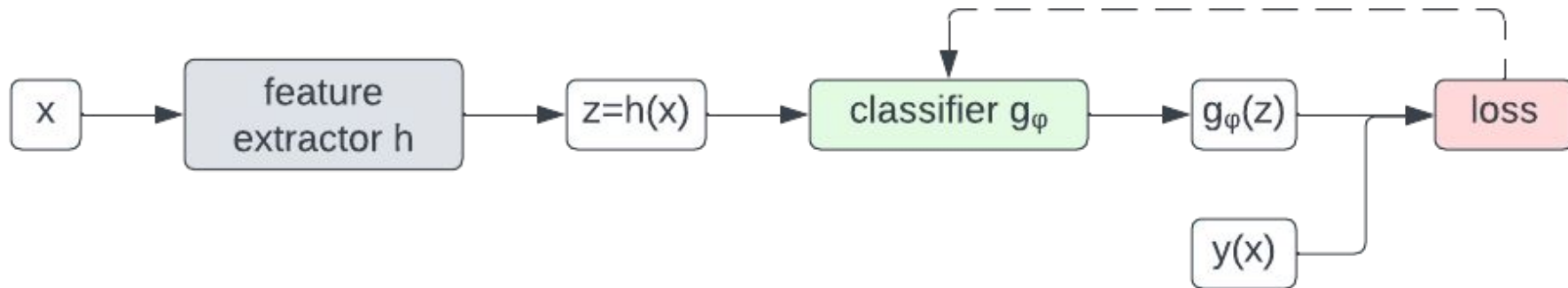


ML pipeline: высокая размерность

Проблемы ML на пикселях:

- высокая размерность -> требуется много вычислительных ресурсов
- соседние пиксели не независимы -> вредит многим моделям

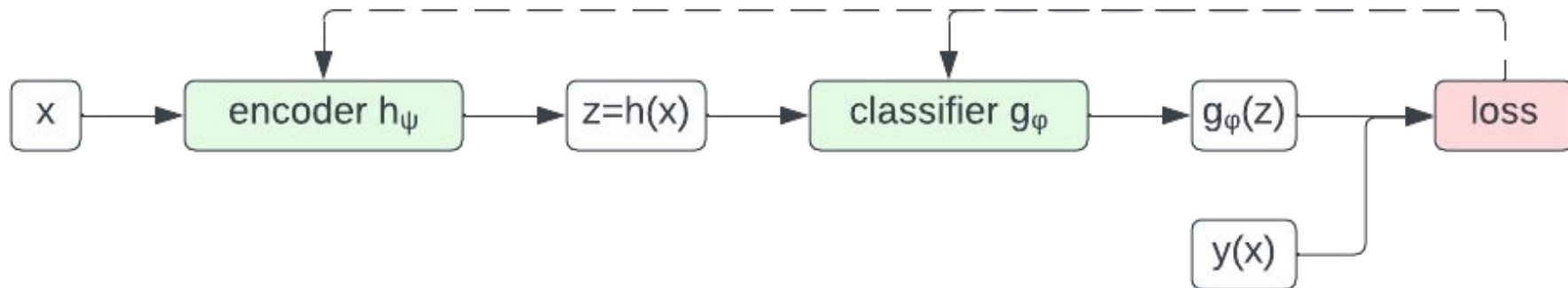
Решение: делать извлечение признаков



ML pipeline: обучаемое извлечение признаков

Новые проблемы: строить feature extractor'ы дорого & долго

Решение - сделать извлечение признаков обучаемым: $h \rightarrow h_\psi$



Устройство энкодера

Требования к h_ψ :

- дифференцируемость - иначе необучаемо
- нелинейность - иначе невыразительные признаки

Стандартный подход: чередование линейных и нелинейных преобразований (функций активации)

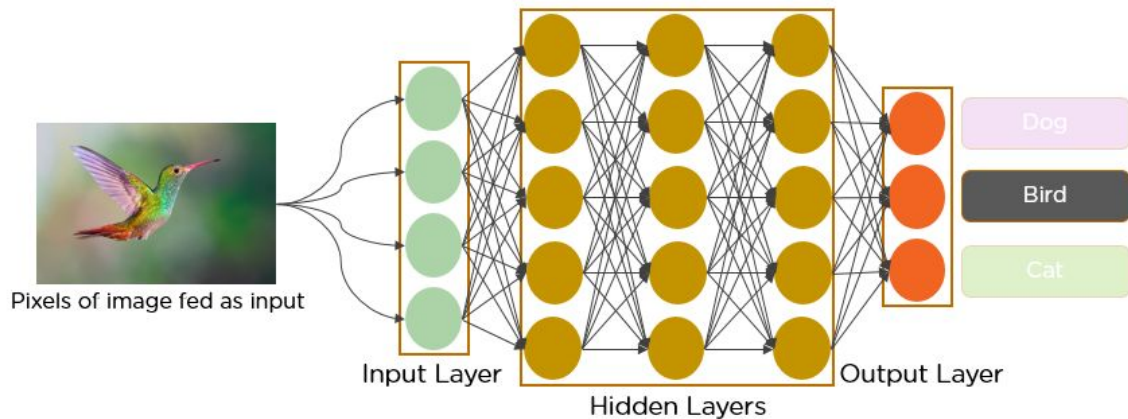
- линейные быстро обрабатываются GPU и CPU
- нелинейные расширяют класс реализуемых функций

Note: h_ψ и g_ϕ можно объединить в единую обучаемую функцию $f_\theta = g_\phi(h_\psi(x))$

FCNs

Базовое DL решение:

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



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

Обучение

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

FCNs: проблемы для CV

FCNs плохо подходят для картинок:

- высокая размерность картинок -> полносвязные слои будут содержать очень много параметров
 - требует много ресурсов (память, компьютер)
 - оверфиттинг
- не используют пространственную информацию

Решение: заменить FC слои на сверточные слои

Сверточный слой

2d свертка

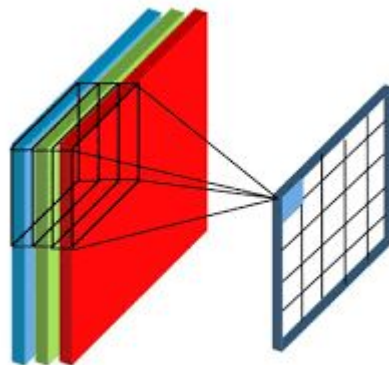
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

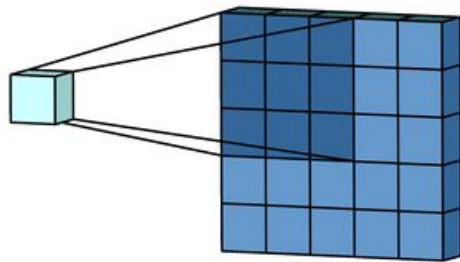
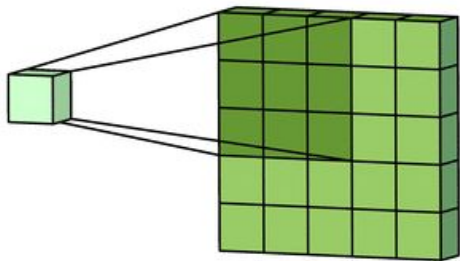
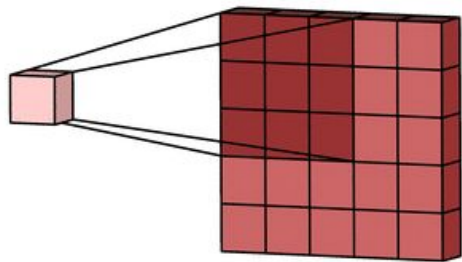
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

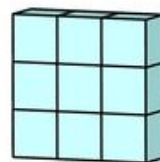
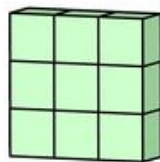
3d свертка

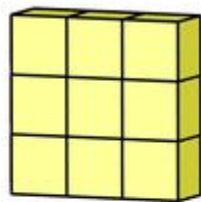
Изображение: $H \times W \times C$

Ядро 3d свертки: $h \times w \times C$







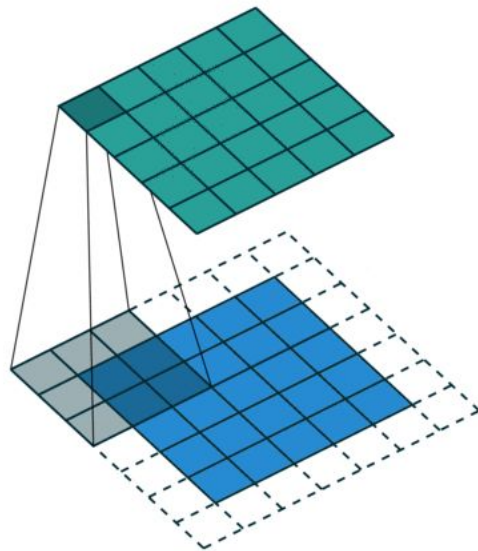


Гиперпараметры свертки

- размер ядра (kernel size)

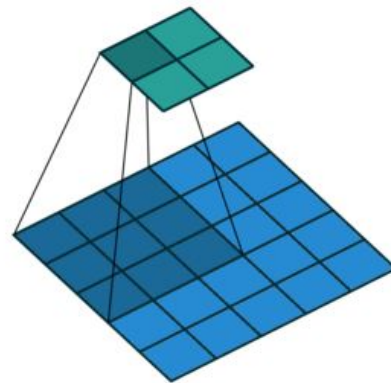
Гиперпараметры свертки

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



Гиперпараметры свертки

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



Сверточный слой

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

Гиперпараметры:

- количество фильтров
- гиперпараметры свертки

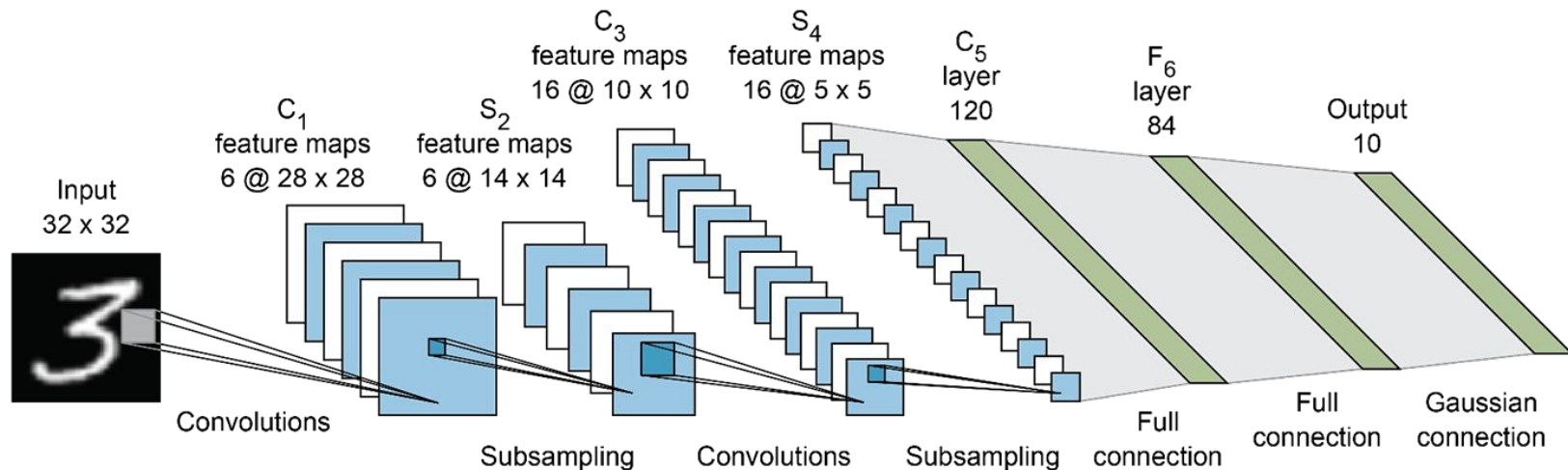
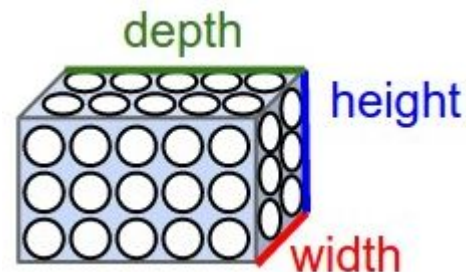
Опр. Карта признаков (feature map) - матрица значений определенного канала.

Словарь

Выход свёрточного слоя - 3d массив $H^* \times W^* \times C^*$

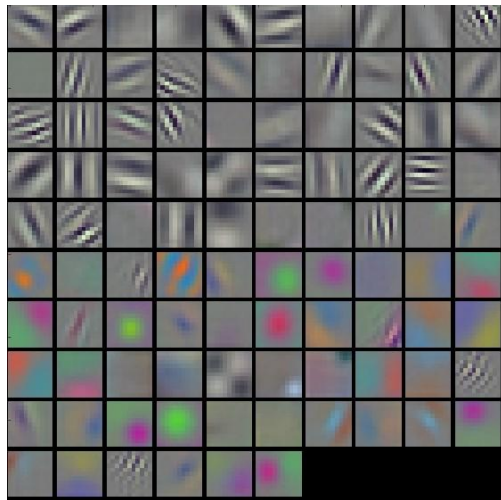
=> к нему тоже можно применять 3d свёртку

=> можно делать композицию сверточных слоев



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

Преимущества conv слоя сравнению с FC:

- работа с данными переменного размера
- разреженные связи
 - ⇒ меньше параметров
 - ⇒ надо меньше ресурсов (памяти, компьютера)
- меньше параметров ⇒ меньше оверфиттинга
- переиспользование весов
 - ⇒ извлечение локальных признаков, универсальных относительно трансляции
 - ⇒ больше обобщаемость
- использует информацию о пространственной структуре

Итого: conv слои ресурсо-эффективнее и лучше обобщают

Недостатки:

- сверточный слой не способен найти связи между далекими пикселями

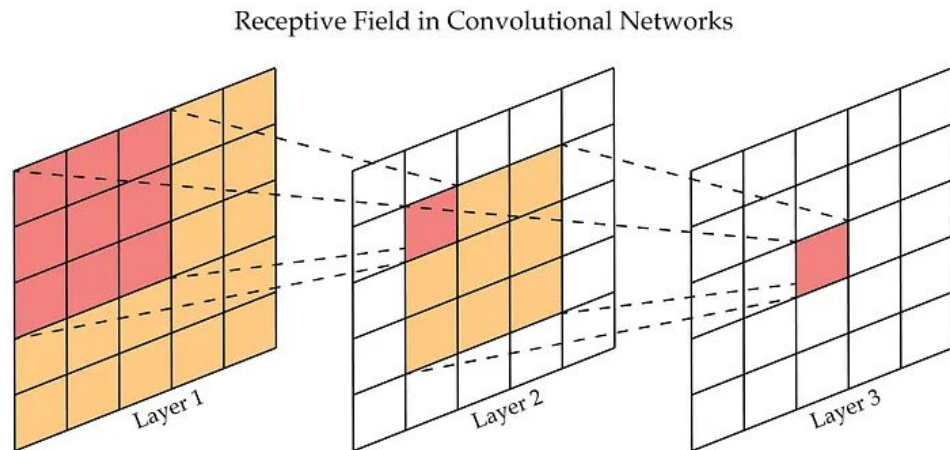
Receptive Field

Рецептивное поле (receptive field) нейрона

– это область входного изображения, которая влияет на активацию нейрона (данного нейрона в данном слое сети).

Иерархическая структура:

- увеличение рецептивного поля от слоя к слою
- нижние слои - локальные признаки
верхние слои - глобальные признаки



source: <https://medium.com/@rekalantar/receptive-fields-in-deep-convolutional-networks-43871d2ef2e9>

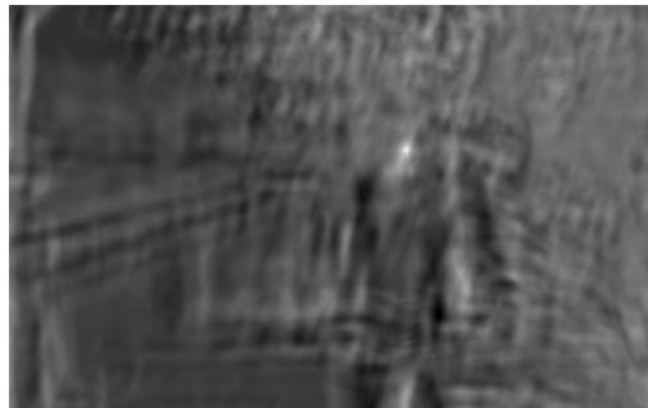
Свертка как template matching



source: https://www.freepik.com/free-photo/enjoying-cycling_5534982.htm



conv



CNN

CNNs

Основные виды слоев сверточной сети (convolutional neural network, CNN):

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

Pooling

Пулинг (pooling) = субдискретизация (subsampling)

– поблочная агрегация элементов тензора с помощью некоторой необучаемой функции.

Note:

- фильтрация (линейная/нелинейная)
- нет обучаемых параметров
- каждый канал обрабатывается независимо

Стандартные функции агрегации:

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

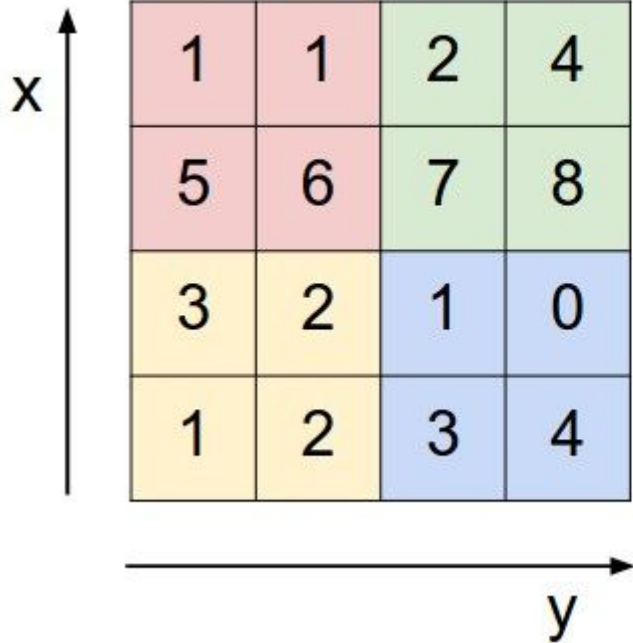
Pooling

Гиперпараметры:

- функция агрегации
- размер ядра (kernel size) K
- шаг пулинга (stride) S

Обычно: $K=2$, $S=2$

Single depth slice



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters
and stride 2



6	8
3	4

Pooling

Польза пулинга:

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

Функции активации










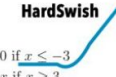



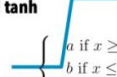
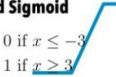

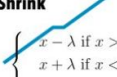
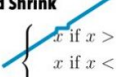
Функции активации (activation function) = нелинейность (nonlinearity):

- популярные: ReLU, LeakyReLU
- классика: tanh, sigmoid
- публикации: GELU, SELU, Mish, Swish, ...

Требования:

- обязательно
 - нелинейность
 - дифференцируемость почти везде
- зачастую
 - неубывание

Neural Network Activation Functions: a small subset!

ReLU  $\max(0, x)$	GELU  $\frac{x}{2} \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + ax^3) \right) \right)$	PReLU  $\max(0, x)$
ELU  $\begin{cases} x & \text{if } x > 0 \\ \alpha(x \exp x - 1) & \text{if } x < 0 \end{cases}$	Swish  $\frac{x}{1 + \exp -x}$	SELU  $\alpha(\max(0, x) + \min(0, \beta(\exp x - 1)))$
SoftPlus  $\frac{1}{\beta} \log(1 + \exp(\beta x))$	Mish  $x \tanh \left(\frac{1}{\beta} \log(1 + \exp(\beta x)) \right)$	RRReLU  $\begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \text{ with } a \sim \mathcal{R}(l, u) \end{cases}$
HardSwish  $\begin{cases} 0 & \text{if } x \leq -3 \\ x & \text{if } x \geq 3 \\ x(x+3)/6 & \text{otherwise} \end{cases}$	Sigmoid  $\frac{1}{1 + \exp(-x)}$	SoftSign  $\frac{x}{1 + x }$
Tanh  $\tanh(x)$	Hard tanh  $\begin{cases} a & \text{if } x \geq a \\ b & \text{if } x \leq b \\ x & \text{otherwise} \end{cases}$	Hard Sigmoid  $\begin{cases} 0 & \text{if } x \leq -3 \\ 1 & \text{if } x \geq 3 \\ x/6 + 1/2 & \text{otherwise} \end{cases}$
Tanh Shrink  $x - \tanh(x)$	Soft Shrink  $\begin{cases} x - \lambda & \text{if } x > \lambda \\ x + \lambda & \text{if } x < -\lambda \\ 0 & \text{otherwise} \end{cases}$	Hard Shrink  $\begin{cases} x & \text{if } x > \lambda \\ x & \text{if } x < -\lambda \\ 0 & \text{otherwise} \end{cases}$

CNNs

Основные виды слоев сверточной сети (convolutional neural network, CNN):

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

В каком порядке их использовать?

Сверточный блок:

- conv \rightarrow activation \rightarrow pooling
- (conv \rightarrow activation) \rightarrow ... \rightarrow (conv \rightarrow activation) \rightarrow pooling

Модификации свертки

Модификации свертки

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

1x1 convolution

- для изменения количества каналов

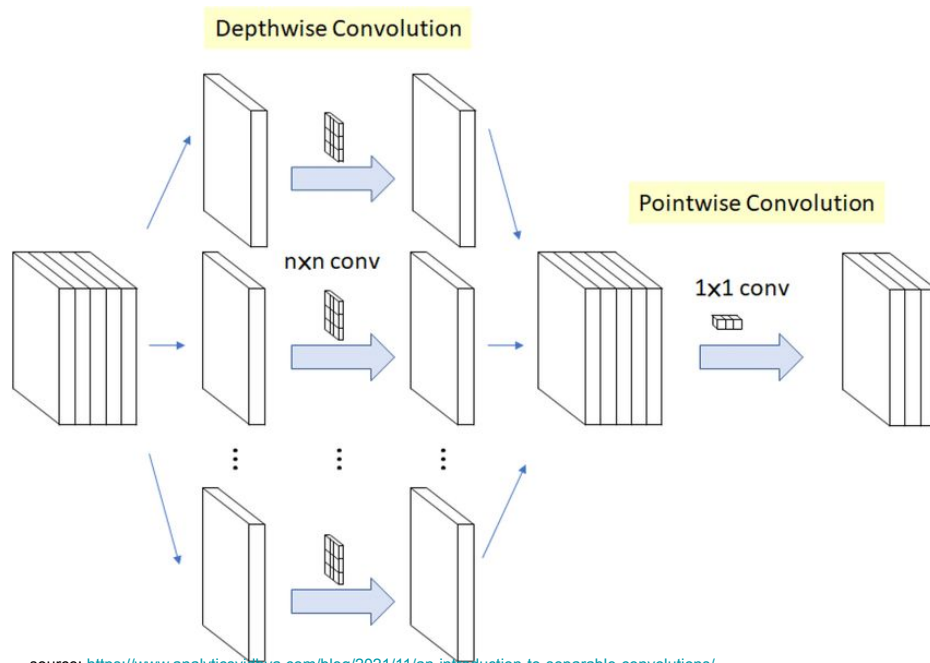
Depthwise Separable Convolution

Разбиваем свертку на 2 части:

- depthwise convolution:
по 1 фильтру на канал
- pointwise convolution:
1x1 свертка

Зачем?

- меньше параметров
- разделение поиска пространственных и канальных зависимостей



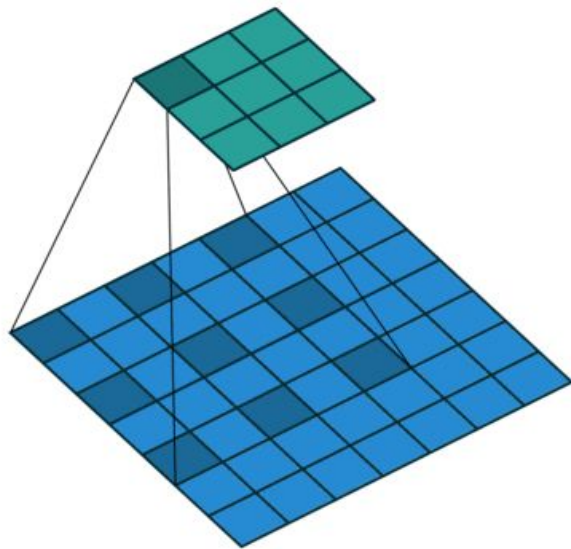
source: <https://www.analyticsvidhya.com/blog/2021/11/an-introduction-to-separable-convolutions/>

Dilated Convolutions

dilated convolutions = atrous convolutions - свертки с пропусками

Зачем?

- больше receptive field
- то же количество параметров



Дополнительные слои

Дополнительные слои

- Dropout
- слои нормализации (Normalization layers)

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:

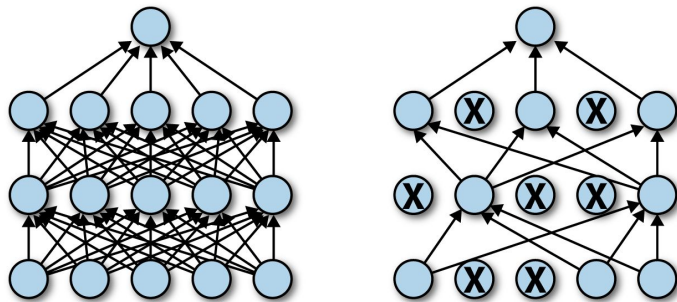
- зануляет выход каждого из нейронов предыдущего слоя с вероятностью p
- умножает все выходы на $1/(1-p)$

Зачем умножение:

чтобы сохранить матожидание

Note:

градиенты



(a) Standard Neural Net
source: <https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html>

Во время инференса: действует как тождественное преобразование

Structured Dropout

Структурированные модификации дропаута:

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

DropBlock

DropBlock [paper](#)

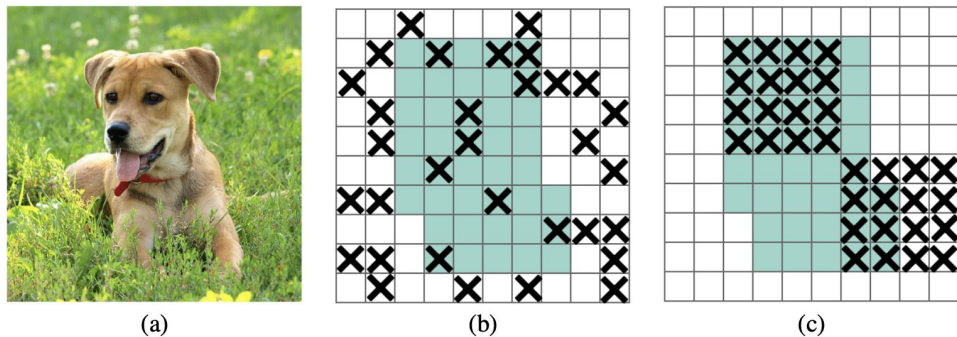
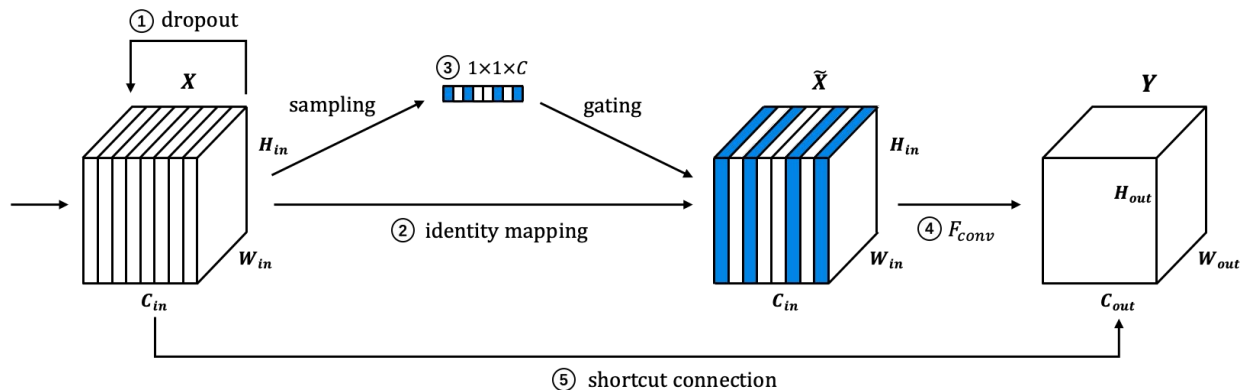


Figure 1: (a) input image to a convolutional neural network. The green regions in (b) and (c) include the activation units which contain semantic information in the input image. Dropping out activations at random is not effective in removing semantic information because nearby activations contain closely 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.

DropChannel

DropChannel [paper](#)
(Dropout2d в PyTorch)



source: <https://github.com/ooibc88/dropout>

DropPath

DropPath [paper](#)

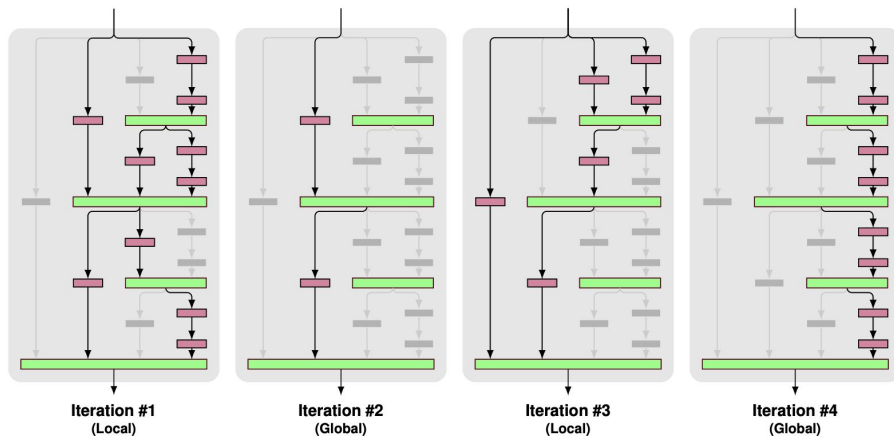


Figure 2: **Drop-path.** A fractal network block functions with some connections between layers disabled, provided some path from input to output is still available. Drop-path guarantees at least one such path, while sampling a subnetwork with many other paths disabled. During training, presenting a 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)

Зачем нужен:

Обработка внутреннего ковариатного сдвига (internal covariate shift) при обучении

Что делает:

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

BatchNorm

2 части

- нормализация статистиками по батчу
- scale & shift обучаемыми параметрами

На инференсе: используются накопленные усредненные по батчам статистики

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

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

Эффекты батчнорма:

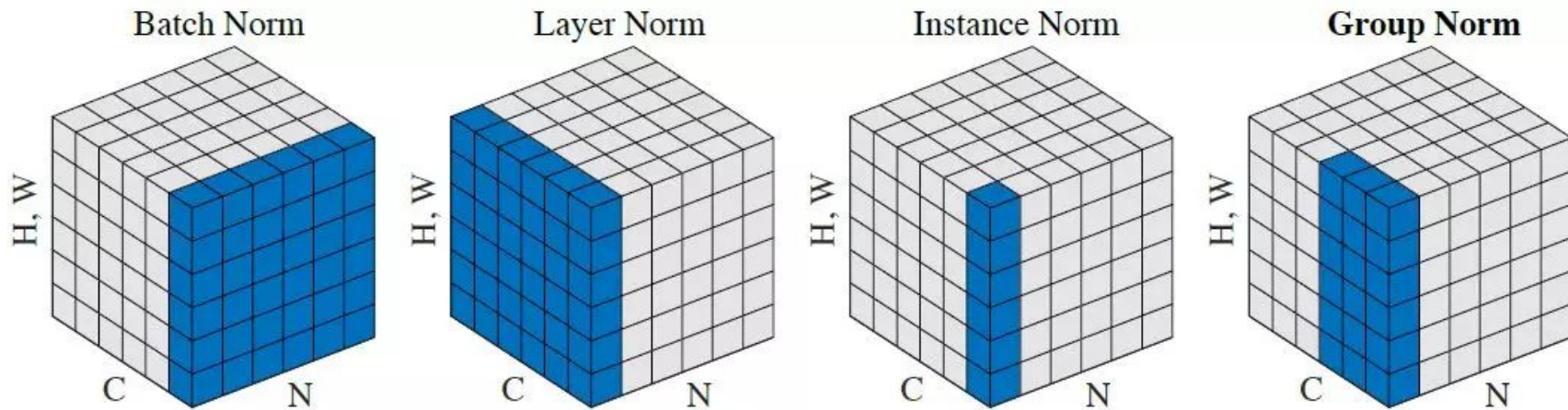
- каждый слой обучается более независимо от других
- разные фичи имеют примерно одинаковый масштаб
- масштаб настраиваемый

Плюсы батчнорма:

- более быстрая сходимость обучения
- можно использовать более высокие lr
- небольшой регуляризационный эффект за счет добавления шума
- меньше чувствительность к начальной инициализации весов

Normalization layers

- BatchNorm
- LayerNorm
- InstanceNorm
- GroupNorm



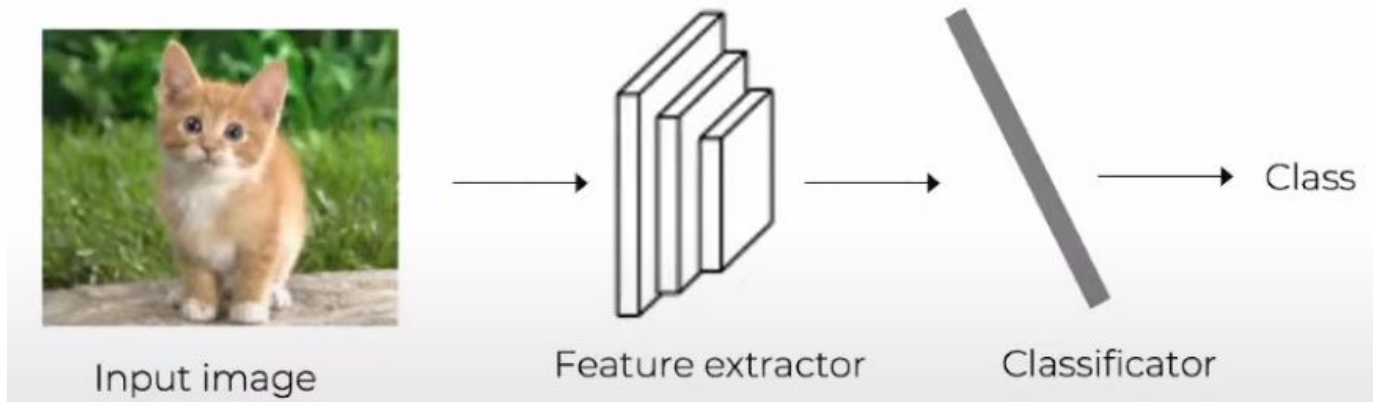
Важные архитектуры

Timeline

- 1989 LeNet
- 2012 AlexNet
- 2014 VGG
- 2014 Inception (GoogleNet)
- 2015 ResNet
- 2016 RexNeXt
- 2018 NASNet
- ...

Архитектура сверточных нейронных сетей

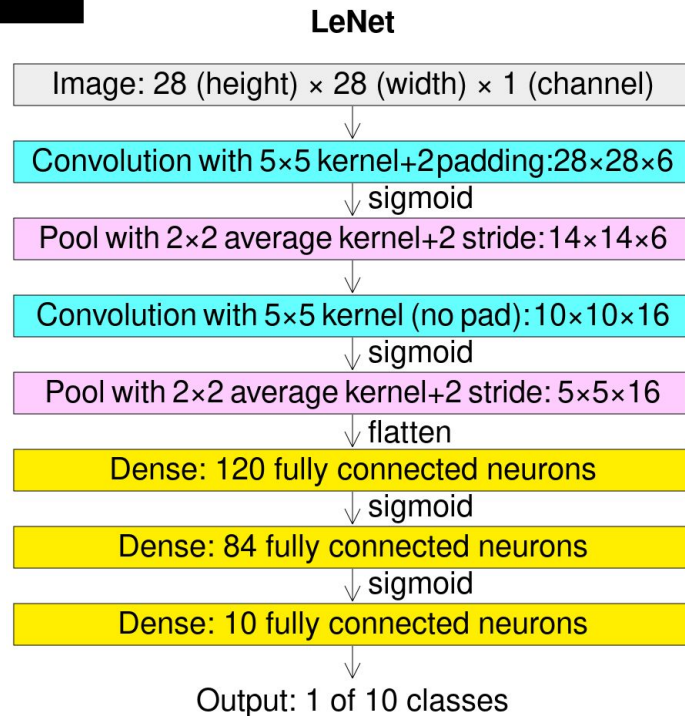
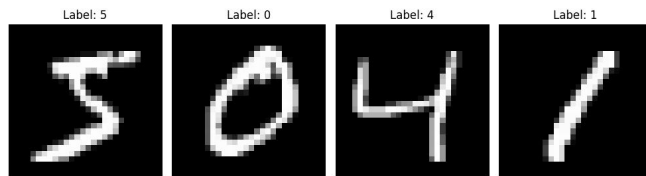
- Энкодер (encoder)
 - несколько сверточных блоков
- классификатор (classifier) = голова (head)
 - полносвязные слои
 - функции активации



LeNet (1989)

Yann LeCun et al.

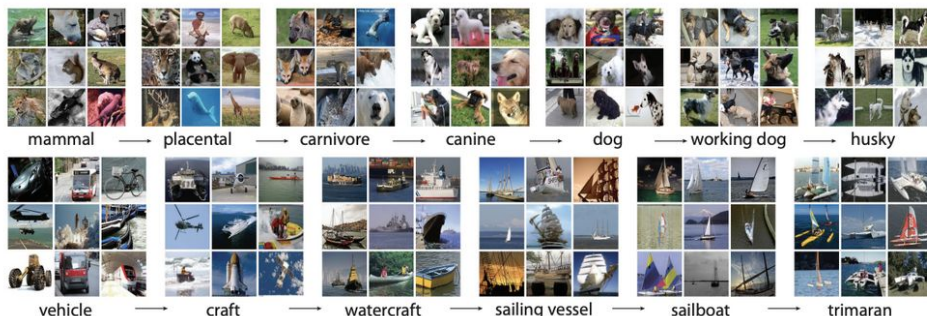
- MNIST: рукописные цифры
70k картинок, grayscale, 28x28
- демонстрация успешности CNNs
- базовые компоненты: conv, pool, FC
- на самом деле, целое семейство
 - 1989 LeNet-1
 - ...
 - 1994 LeNet-4
 - 1995 LeNet-5



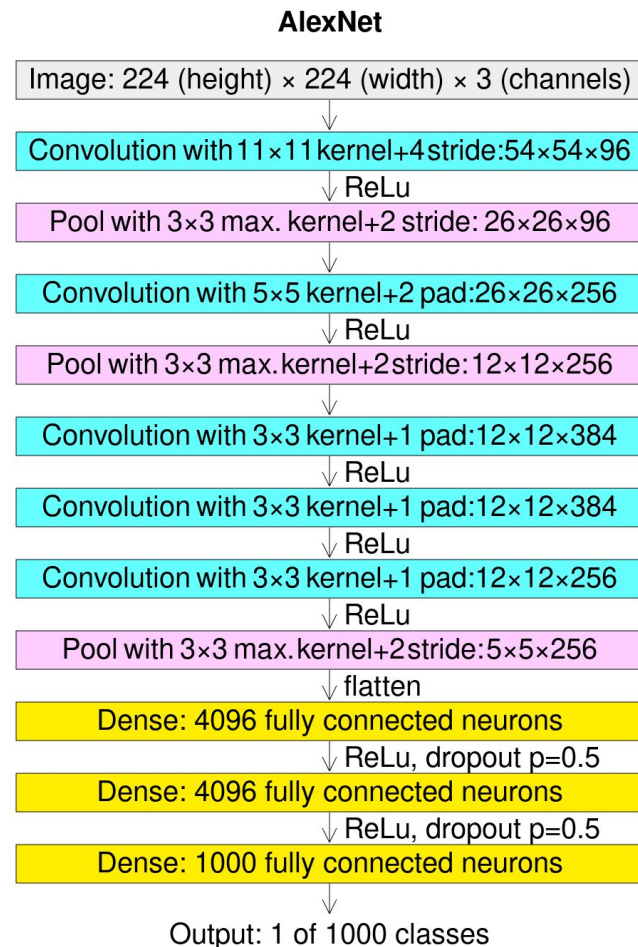
AlexNet (2012)

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

- ImageNet: естественные изображения
~1М штук, 1000 классов, цветные, 224x224
- более глубокая
- базовые компоненты: conv, pool, FC, **ReLU**, **dropout**
- *быстрое* обучение на GPU



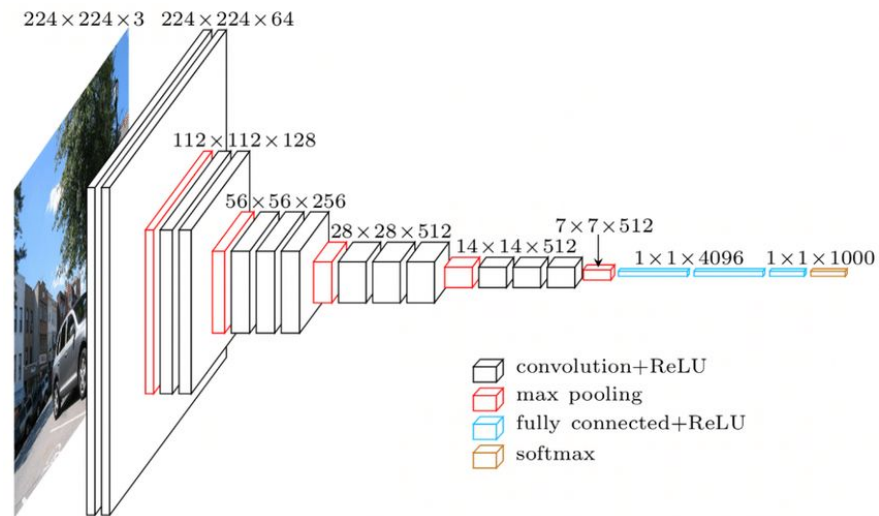
source: https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf?ref=blog.roboflow.com



VGG (2014)

Karen Simonyan, Andrew Zisserman [paper](#)

- блочная архитектура



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

VGG (2014)

Karen Simonyan, Andrew Zisserman [paper](#)

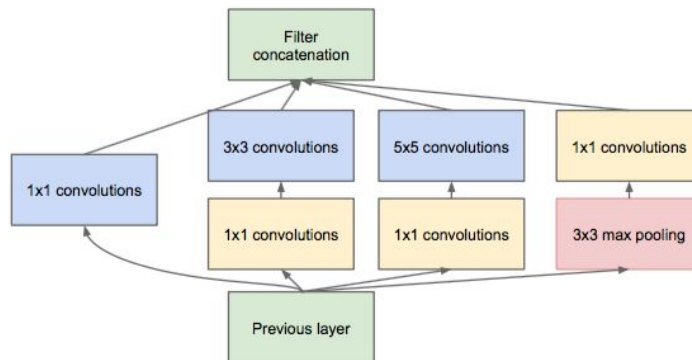
- блочная архитектура
- уменьшили ядра всех сверток до 3x3
 - глубже
 - при этом меньше параметров

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

Inception-v1 (2014)

Christian Szegedy et al [paper](#)

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

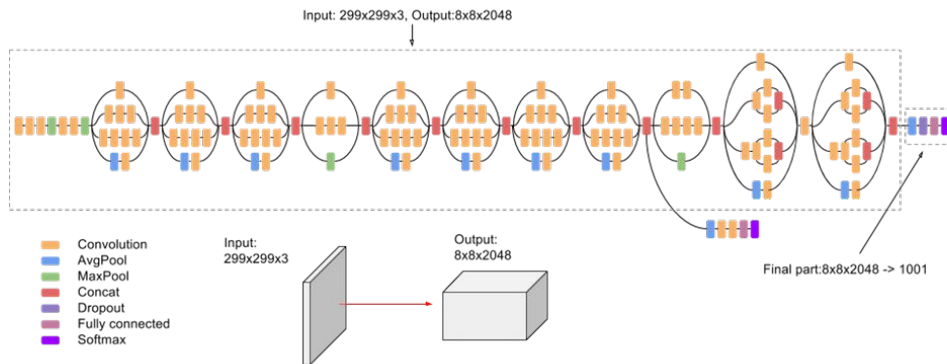


(b) Inception module with dimension reductions

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)

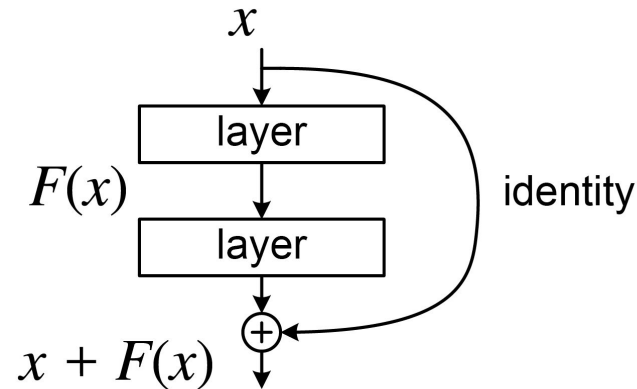
Kaiming He et al. [paper](#)

- 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)

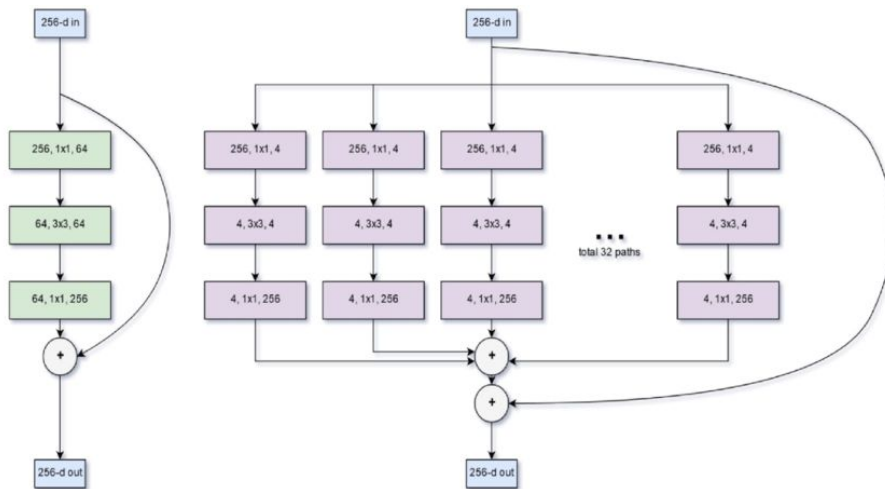
Почему работает?

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

ResNeXt

S. Xie et al. 2016 [paper](#)

- Модификация ResNet
- cardinality: количество параллельных остаточных путей

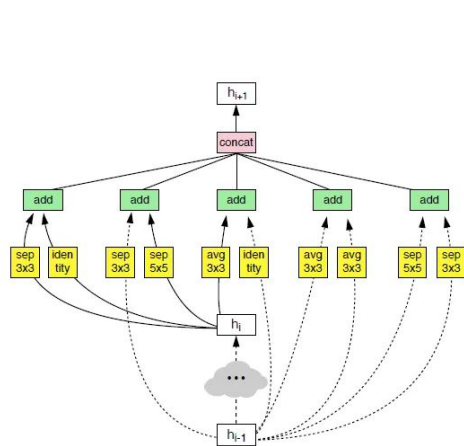


NASNet (2018)

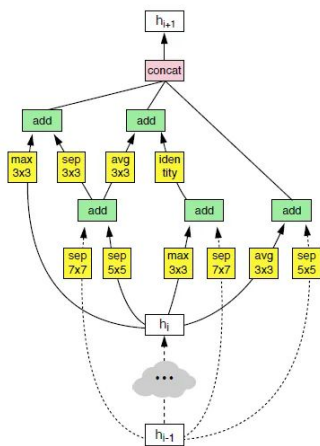
Barret Zoph et al. [paper](#)

NAS = Neural Architecture Search:

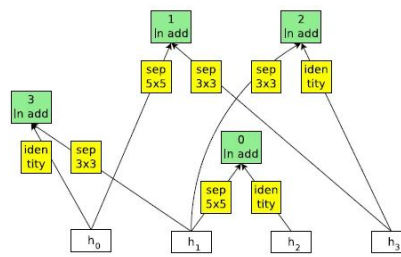
использование AutoML для выбора структуры сетки



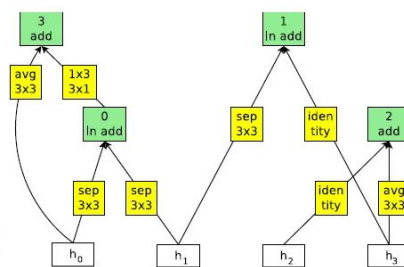
Normal Cell



Reduction Cell



Normal Cell



Reduction Cell

Спасибо за внимание!