

# Цифровая обработка изображения

5. Сверточные нейронные сети

# План занятия

- архитектура сверточных нейронных сетей
- реализация операций свертки
- визуализация слоев, Deep Dream и Style Transfer

# ImageNet Challenge

			
<b>mite</b> black widow cockroach tick starfish	<b>container ship</b> lifeboat amphibian fireboat drilling platform	<b>motor scooter</b> go-kart moped bumper car golfcart	<b>leopard</b> jaguar cheetah snow leopard Egyptian cat
			
<b>grille</b> convertible grille pickup beach wagon fire engine	<b>mushroom</b> agaric mushroom jelly fungus gill fungus dead-man's-fingers	<b>cherry</b> dalmatian grape elderberry ffordshire bullterrier currant	<b>Madagascar cat</b> squirrel monkey spider monkey titi indri howler monkey

# Deep Dream



# Style Transfer



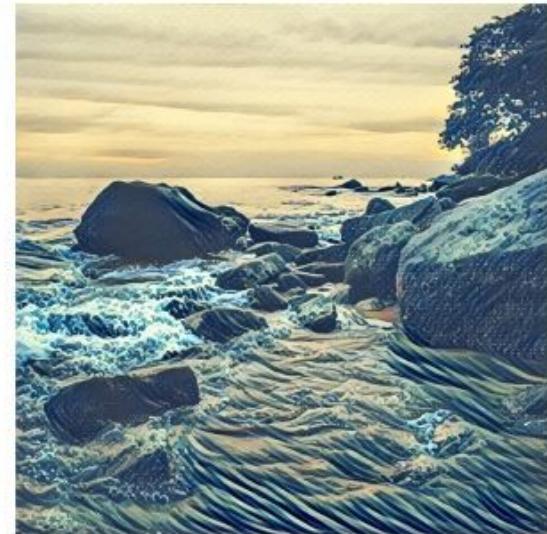
01. Take a picture



02. Pick a style



03. Enjoy the result



# Сверточные нейронные сети

# Мотивация

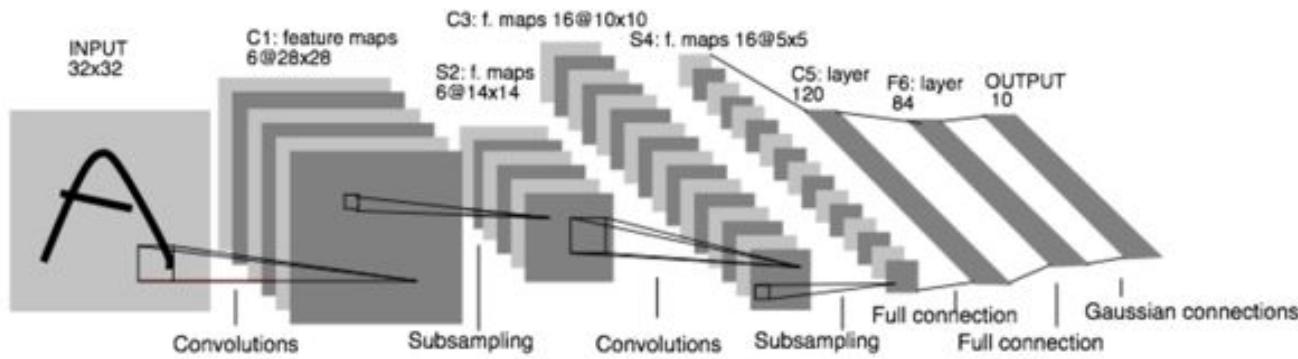
- изображения представляют собой объекты с большим количеством признаков
- изображение в формате RGB размера 640x480 будет иметь ~1млн признаков
- число параметров полносвязной сети с внутренним слоем из 10 нейронов равно ~10млн
- большое число параметров модели существенно затрудняет процесс обучения

# Сверточные нейронные сети

- в нейронных сетях для обработки изображений полносвязные слои заменяют на сверточные
- полносвязные слои могут использоваться в выходном слое
- в сверточных сетях также используются слои понижения размера (Pooling), слои нормализации (BatchNorm) и регуляризации (Dropout)

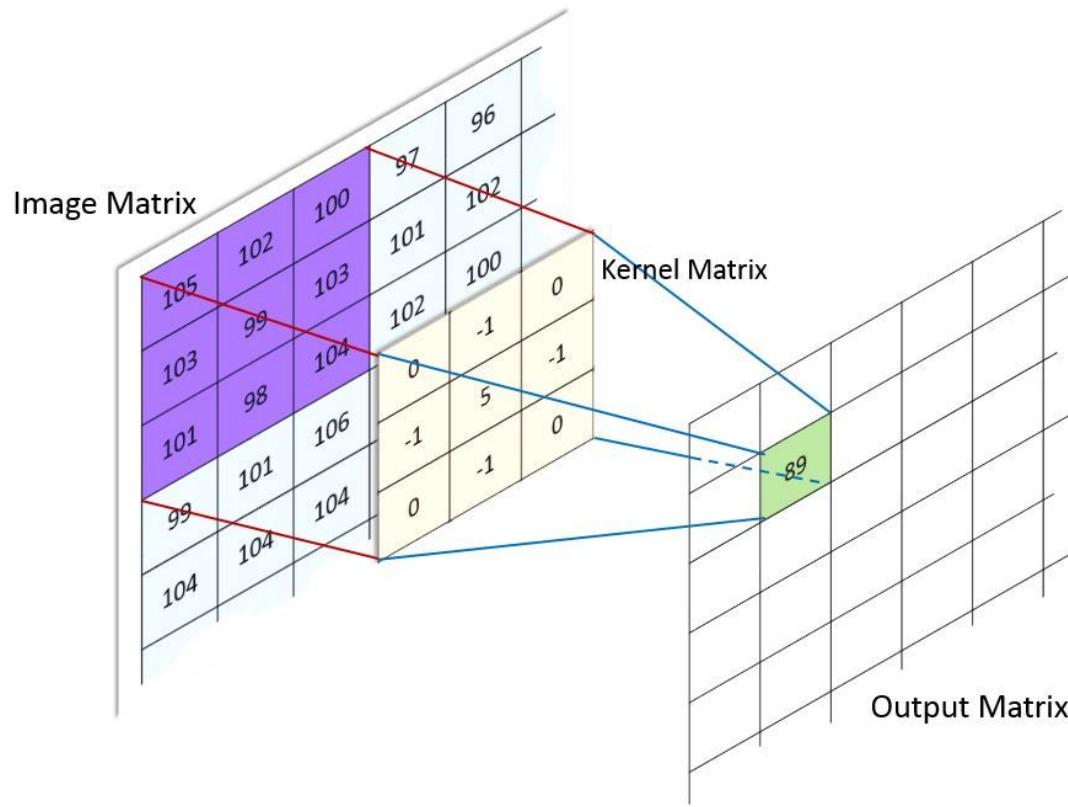
# Архитектура сверточной сети LeNet

## Convolutional Networks: 1989



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ LeNet ]

# Операция свертки



# Операция свертки

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v]F[i - u, j - v]$$

F - исходно изображение

H - фильтр размера k x k

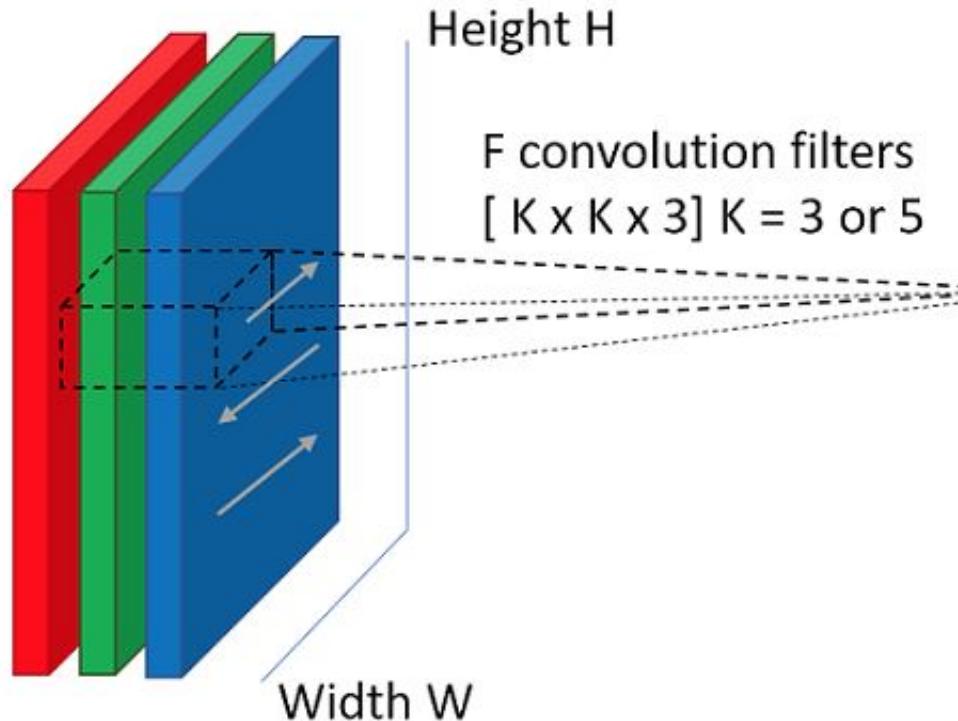
G - результат на выходе свертки

i,j - координаты пикселя в области которого применяется операция свертки

# Сверточный слой (Convolution Layer)

- в отличие от полносвязной сети, каждый нейрон сверточного слоя принимает на вход только часть исходного изображения
- размер входа нейрона (фильтра) называется областью видимости фильтра (kernel size) и является параметром слоя
- при этом глубина фильтра совпадает с числом каналов входного изображения

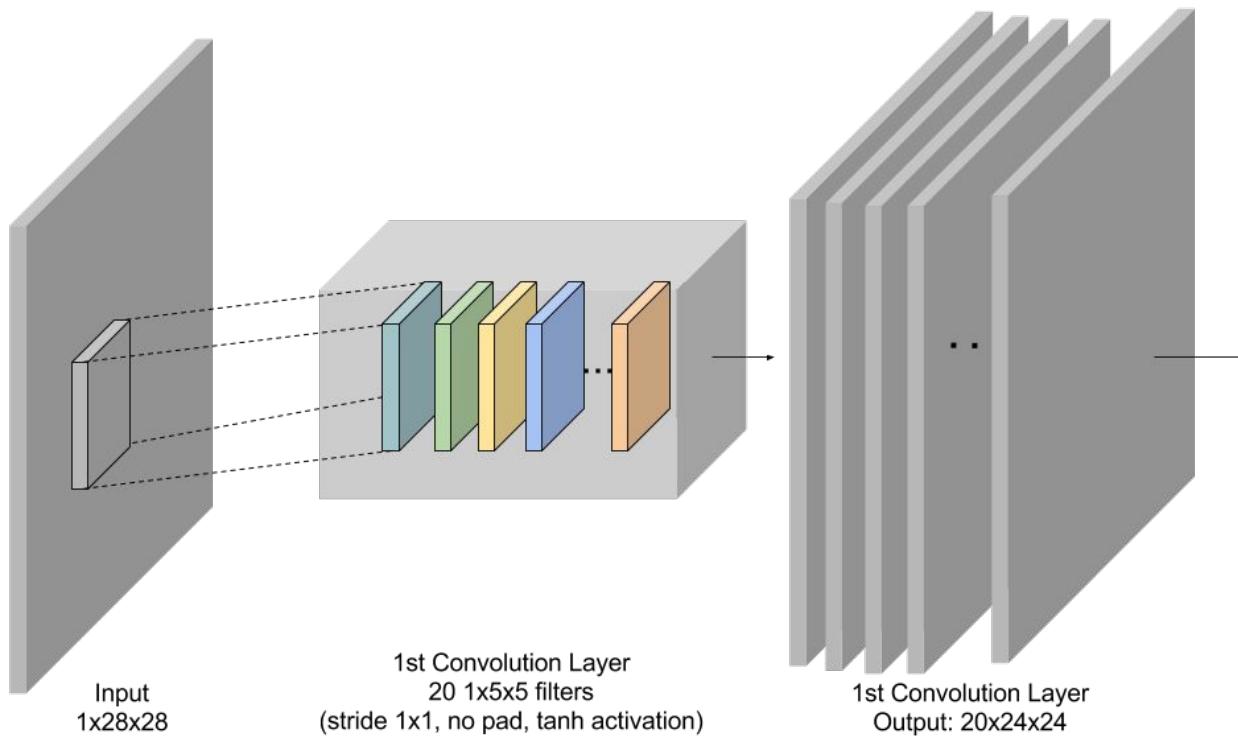
# Сверточный слой (Convolution Layer)



# Сверточный слой (Convolution Layer)

- сверточный слой содержит один или несколько различных фильтров
- размеры фильтров внутри сверточного слоя совпадают
- число фильтров определяет число каналов на выходе сверточного слоя

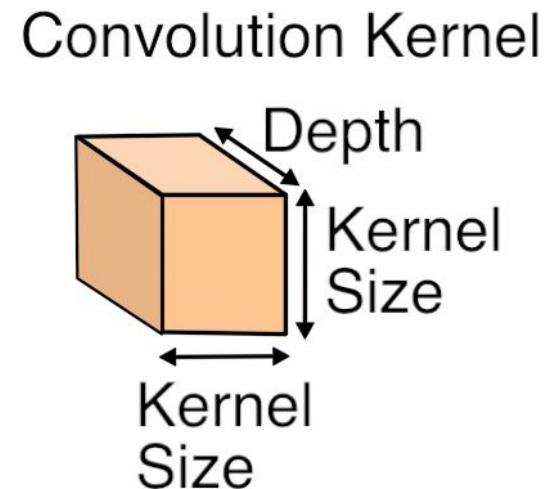
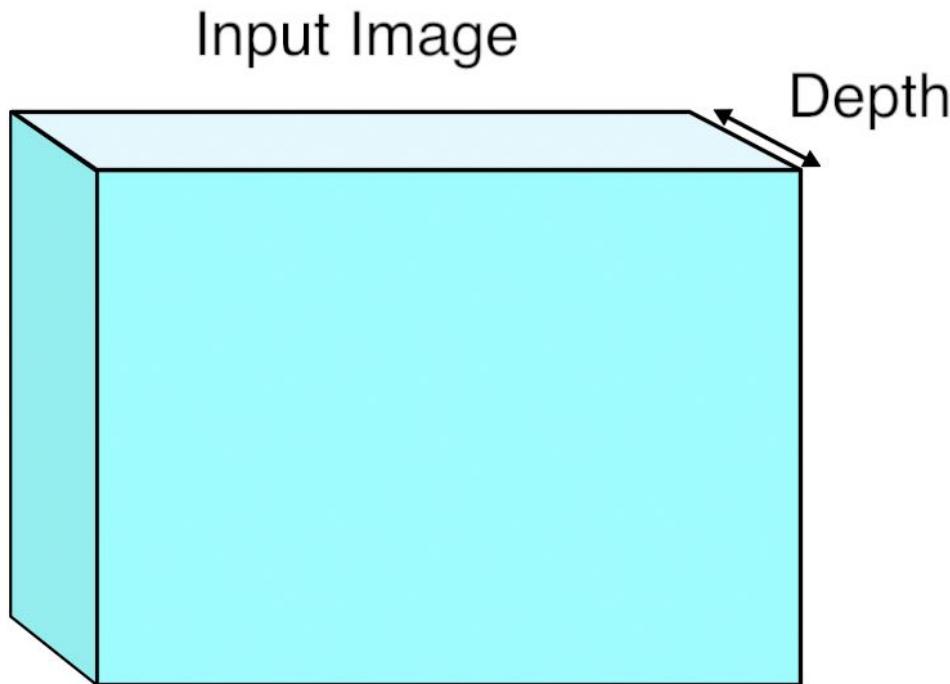
# Сверточный слой (Convolution Layer)



# Параметры свертки

- размер фильтра (kernel size, receptive field)
- отступы (padding: same/valid)
- шаг (stride)
- расширение (dilation)

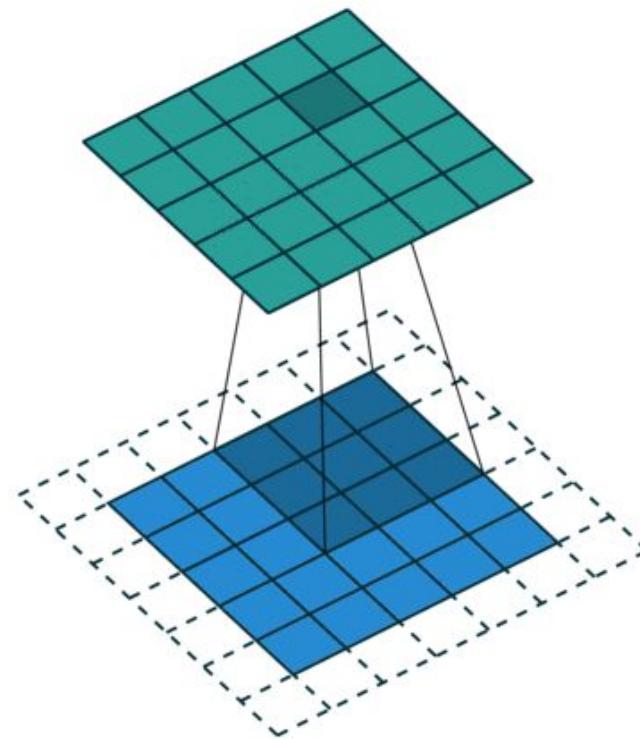
# Размер фильтра (kernel size)



# Размер фильтра (kernel size)

- размер фильтра определяет область анализируемой части изображения
- типичные размеры сверточного фильтра 3 - 7 пикселя, как правило нечетное значение
- глубина сверточного фильтра совпадает с числом каналов исходного изображения

# Отступы (padding)



# Отступы (padding)

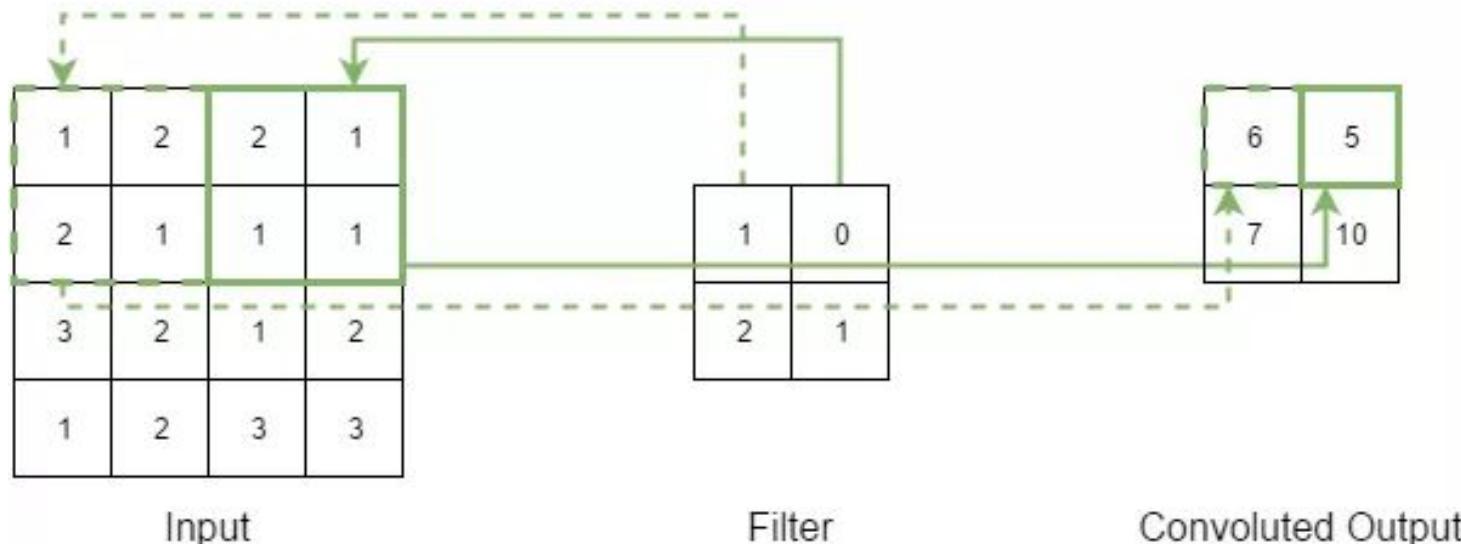
- при последовательном применении фильтров размер выхода будет уменьшаться с каждым шагом на размер фильтра
- при этом теряется информация о краях изображения

# Отступы (padding)

- чтобы размер на выходе совпадал с размером на входе, размер исходного изображения увеличивают, заполняя пространство по периметру нулями (zero padding)
- как правило параметр padding принимает два значения:
  - same - сохраняет размер исходного изображения
  - valid - отступы не добавляются, размер выхода меньше входа

# Шаг (stride)

Stride (2,2) Convolution



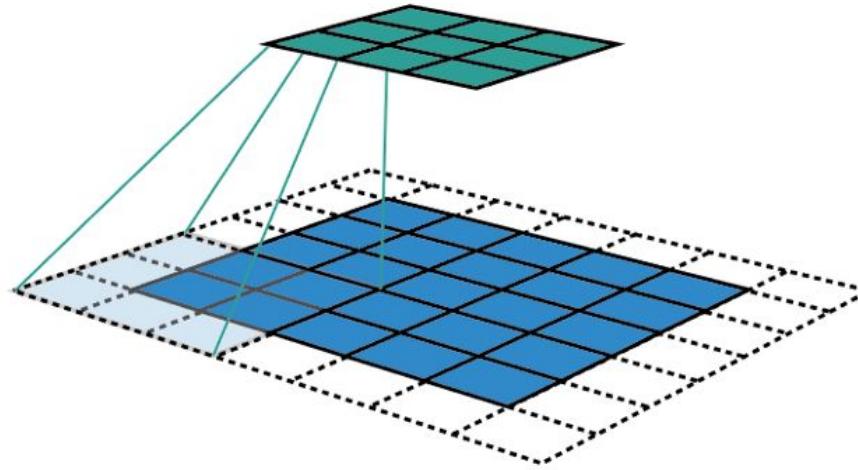
# Шаг (stride)

- шаг может быть разным по x и у
- определяет смещение фильтра на очередной итерации
- при увеличении шага уменьшается размер выхода

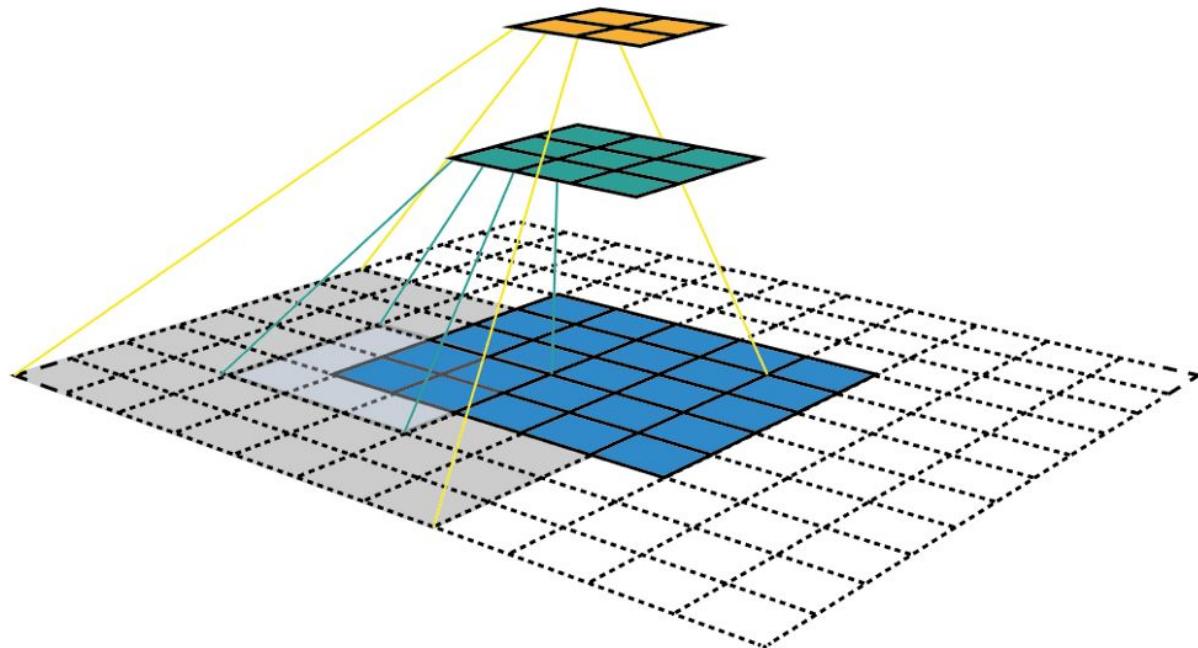
# Область видимости фильтра (receptive field)

- область видимости - область исходного изображения от которой зависит значение пикселя на выходе сверточного слоя
- область видимости отдельного пикселя на выходе сверточного слоя возрастает с увеличением глубины сети

# Область видимости фильтра (receptive field)



# Область видимости фильтра (receptive field)

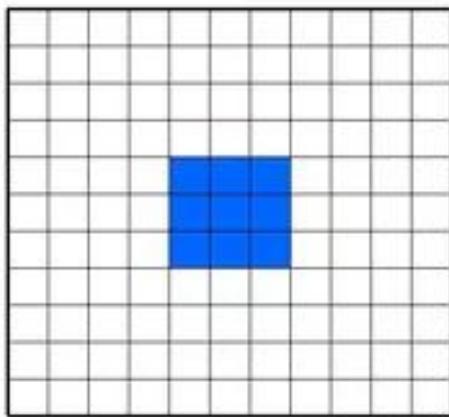


# Расширение (dilation)

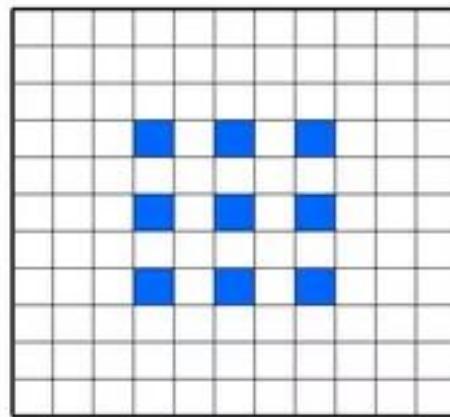
- увеличение размера фильтра (области видимости) без увеличения числа параметров (весов)
- веса фильтра “раздвигаются” в пространстве
- фильтр имеет разреженную структуру
- свободные позиции фильтра заполняются нулями

# Расширение (dilation)

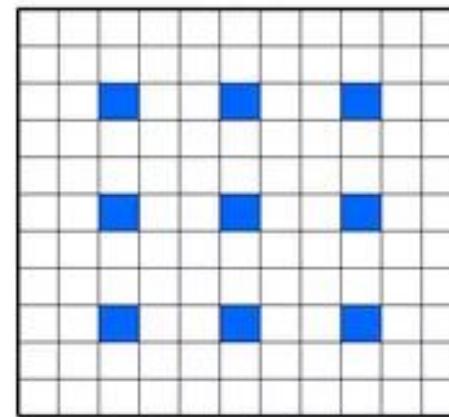
D = 1



D = 2



D = 3

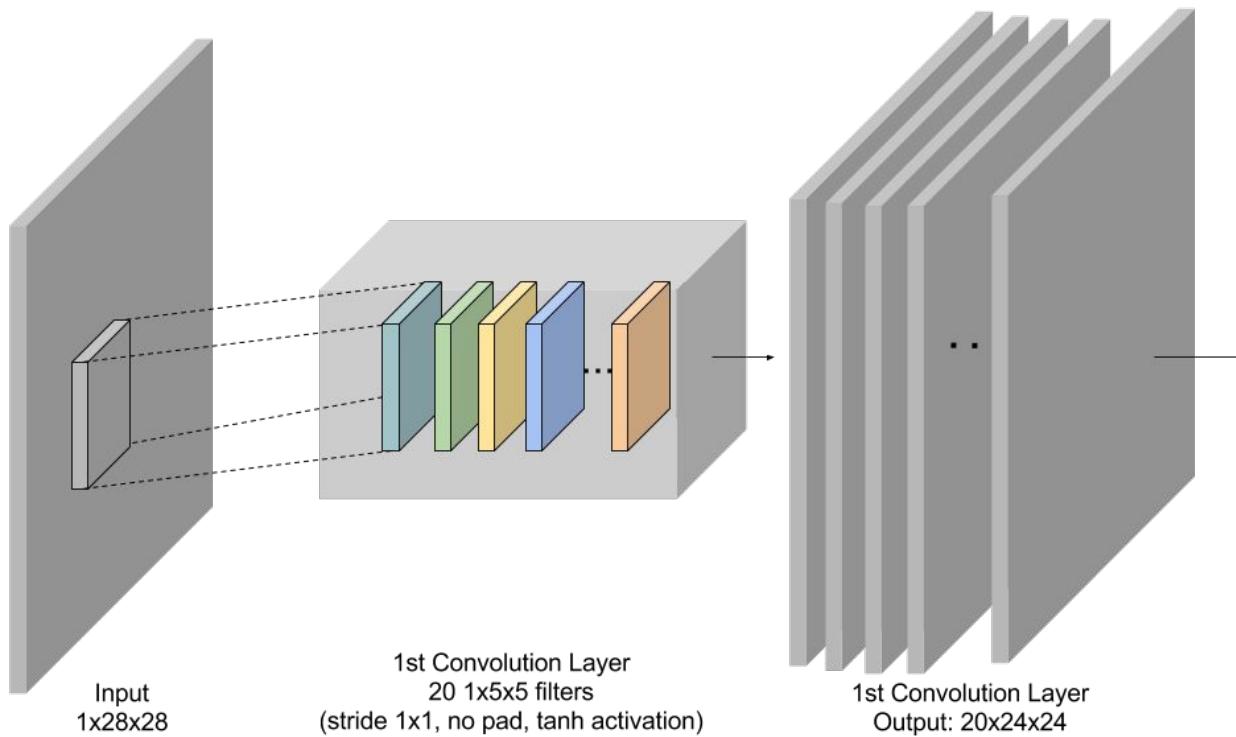


# Сверточный слой

# Сверточный слой (Convolution Layer)

- содержит набор фильтров одинакового размера
- число параметров в слое определяется числом фильтров, их размером и глубиной входа
- глубина на выходе сверточного слоя определяется числом фильтров сверточного слоя
- к выходу сверточного слоя применяется нелинейная функция активации, например, ReLU

# Сверточный слой (Convolution Layer)



# Параметры сверточного слоя в Keras

```
keras.layers.convolutional.Conv2D(filters, kernel_size,  
                                  strides=(1, 1), padding='valid',  
                                  dilation_rate=(1, 1), activation=None)
```

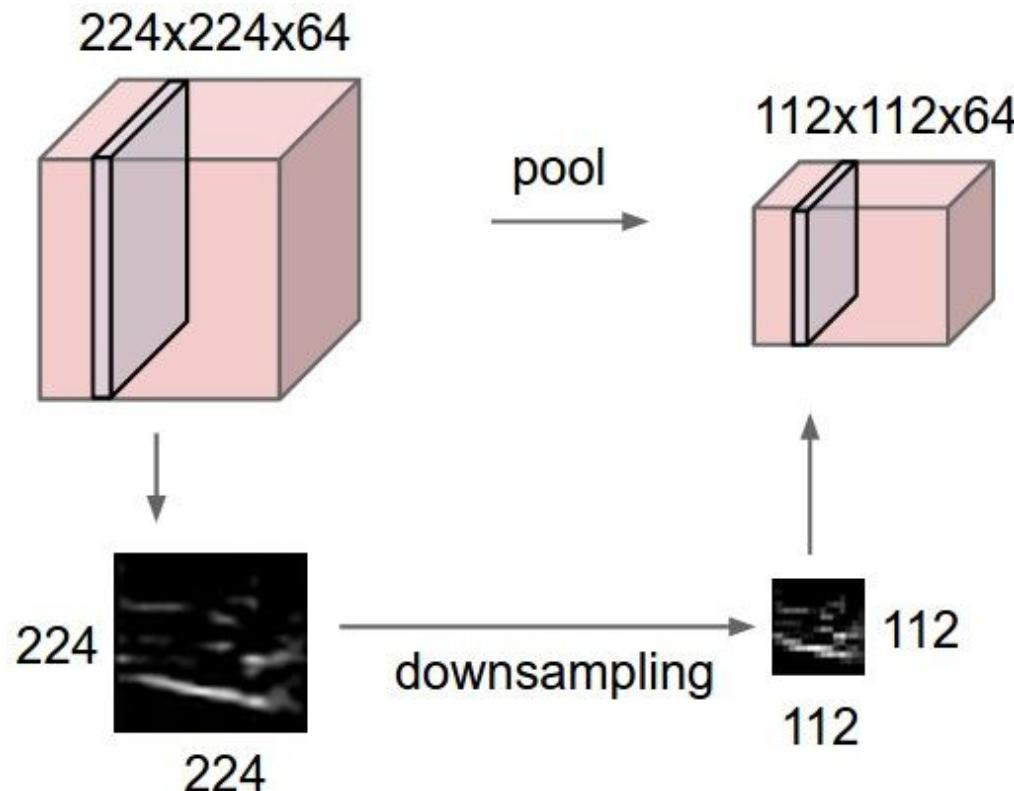
- **filters**: число фильтров в сверточном слое
- **kernel\_size**: размер фильтра
- **strides**: шаг фильтра
- **padding**: отступы "valid" или "same"
- **dilation\_rate**: расширение
- **activation**: функция активации, по умолчанию линейная

# Pooling Layer

# Pooling Layer

- pooling слой вставляется после активаций сверточного слоя
- уменьшает пространственный размер входных данных
- как результат, уменьшается число параметров сети

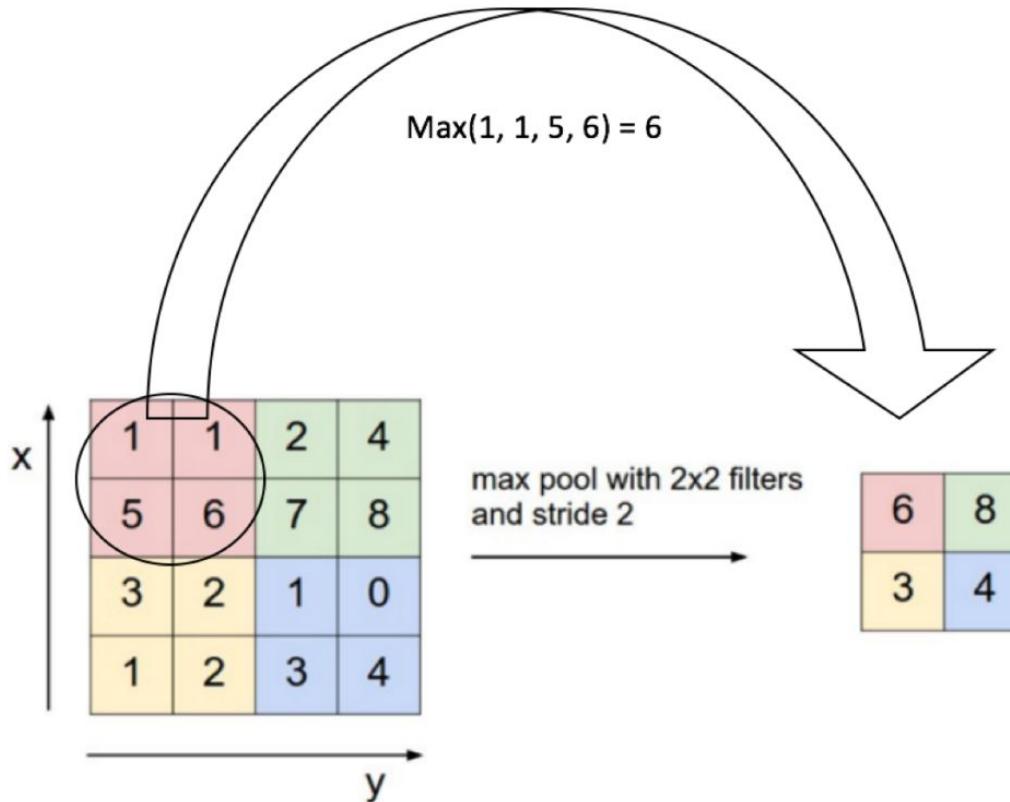
# Pooling Layer



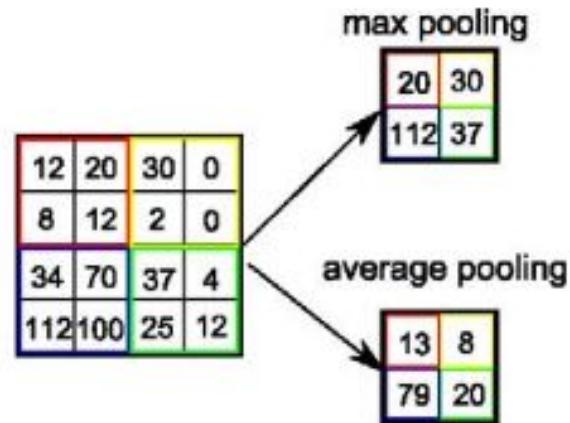
# Pooling Layer

- типичный размер окна pooling слоя 2
- шаг фильтра совпадает с размером окна
- вместо свертки применяется операция MAX или MEAN (среднее)
- глубина данных на выходе, как правило, совпадает с глубиной входных данных

# Pooling Layer



# Pooling Layer



Max



Average



# Pooling Layer

```
keras.layers.pooling.MaxPooling2D(pool_size=(2, 2), strides=None,  
padding='valid', data_format=None)
```

- **pool\_size**: размер окна
- **strides**: шаг окна, по умолчанию совпадает с pool\_size
- **padding**: отступ "valid" или "same"
- **data\_format**: задает измерение в котором хранятся номера каналов

# Pooling Layer

```
keras.layers.pooling.AveragePooling2D(pool_size=(2, 2),  
                                      strides=None, padding='valid', data_format=None)
```

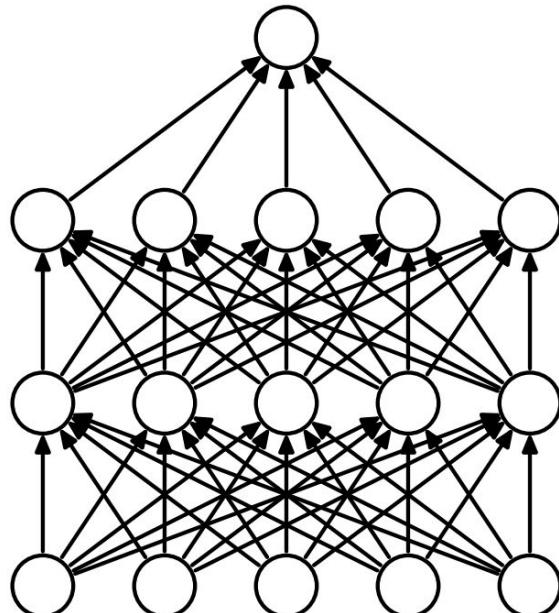
- **pool\_size**: размер окна
- **strides**: шаг окна, по умолчанию совпадает с pool\_size
- **padding**: отступ "valid" или "same"
- **data\_format**: задает измерение в котором хранятся номера каналов

# Dropout

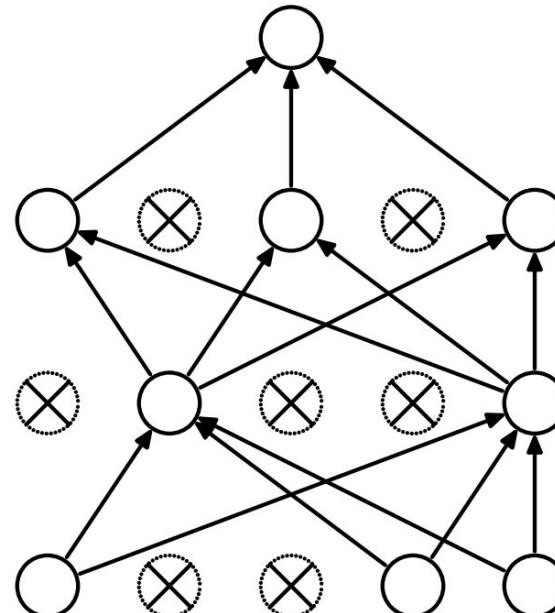
# Dropout

- в процессе обучения случайным образом выбранная часть входов слоя зануляется
- доля зануляемых входов задается параметром слоя
- на этапе предсказания значения входов перевзвешиваются пропорционально доли зануленных входово
- такой подход позволяет модели быть более устойчивой к переобучению

# Dropout



(a) Standard Neural Net



(b) After applying dropout.

[A Simple Way to Prevent Neural Networks from Overfitting](#)

# Dropout

```
keras.layers.core.Dropout(rate, noise_shape=None, seed=None)
```

- **rate**: диапазон 0..1
- **noise\_shape**: позволяет задать одинаковую маску по каналам
- **seed**: инициализация генератора случайных чисел

# Batch Normalization

# Batch Normalization

- приводит значения активаций на выходе слоя к нулевому среднему и единичной дисперсии
- во время обучения среднее и дисперсия оцениваются для каждого батча
- такая нормировка дает прирост как в качестве, так и в скорости сходимости процесса обучения
- повышается обобщающая способность сети

# Batch Normalization

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

Parameters to be learned:  $\gamma, \beta$

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

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

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

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

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

# Batch Normalization

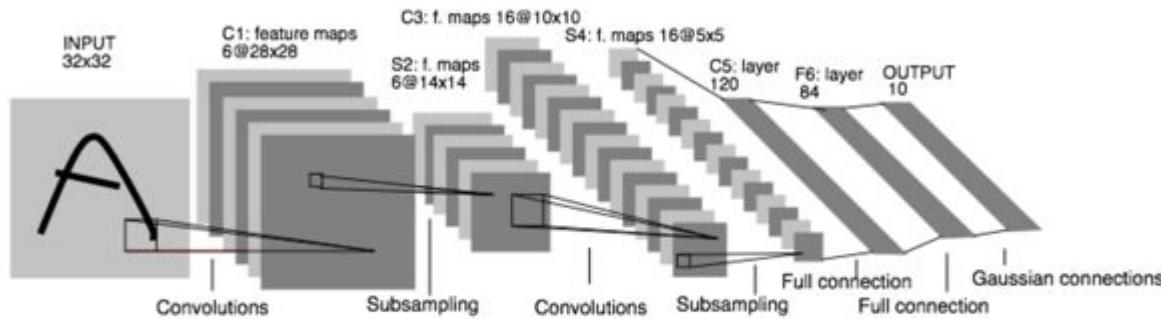
```
keras.layers.normalization.BatchNormalization(axis=-1,  
                                              momentum=0.99, epsilon=0.001, center=True, scale=True,  
                                              beta_regularizer=None, gamma_regularizer=None)
```

- **axis**: размерность для которой выполняется нормализация
- **momentum**: момент для оценки скользящего среднего
- **epsilon**: константа для численной устойчивости
- **center, scale**: флаги центрирования и нормировки (beta, gamma)

# Архитектура сверточной сети LeNet

# Архитектура сверточной сети LeNet

Convolutional Networks: 1989



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ LeNet ]

Gradient-based Learning Applied to Document Recognition

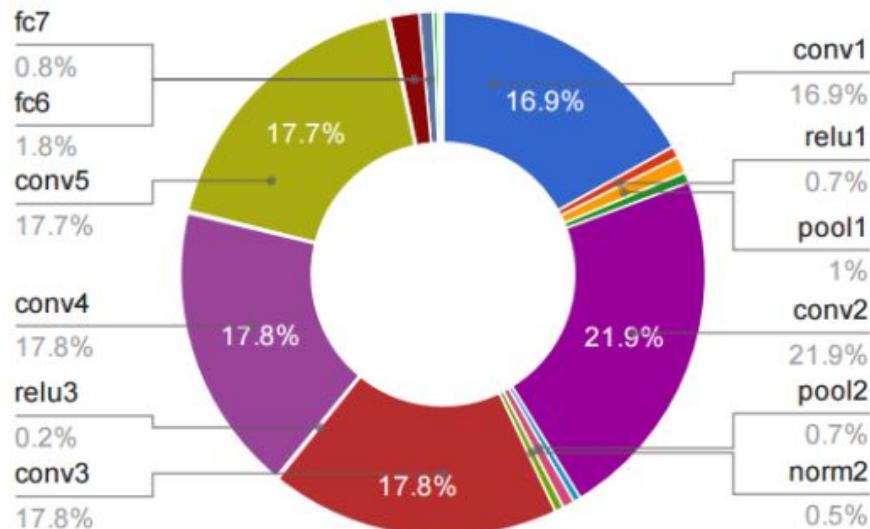
# LeNet

- размер входного изображения фиксирован
- после двух сверточных слоев с семплированием идут 2 полно связных слоя
- выходным слоем сети является полно связный слой с 10 нейронами

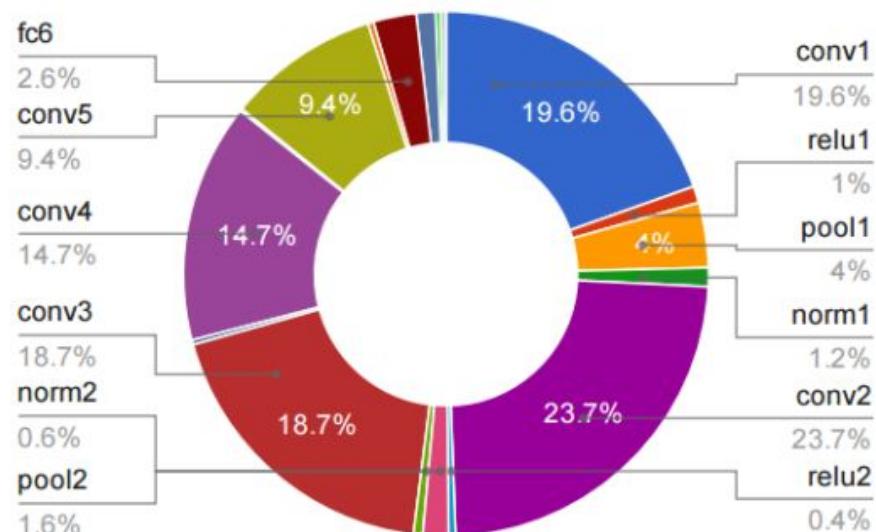
# Реализация операции свертки

# Распределение времени операций

**GPU Forward Time Distribution**



**CPU Forward Time Distribution**

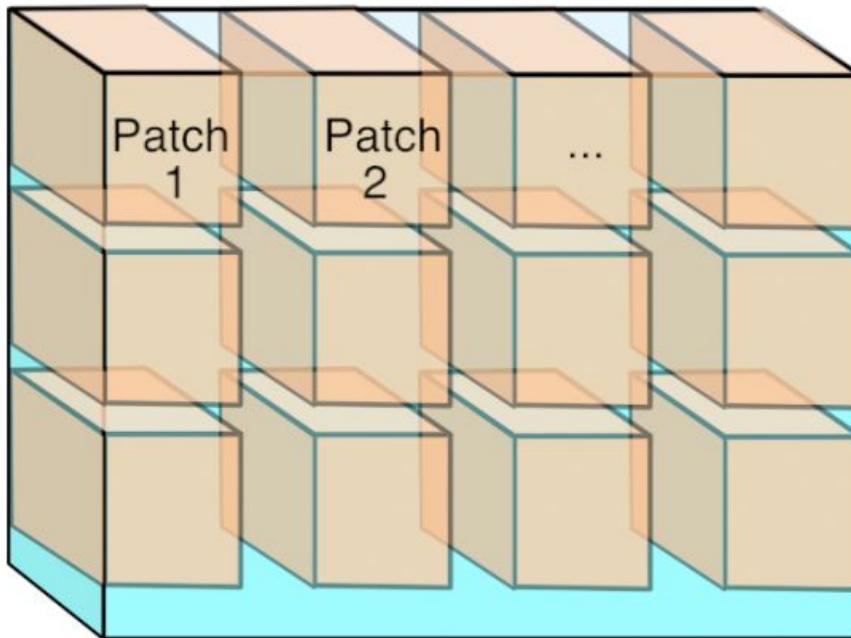


# Реализация операции свертки

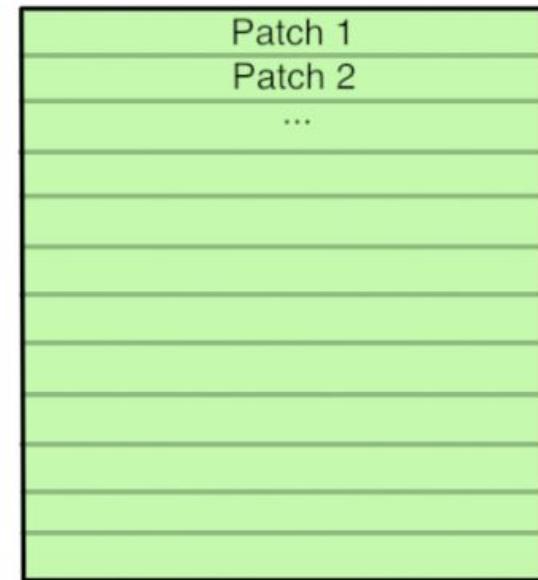
- операция свертки требовательна к вычислительным ресурсам
- для свертки RGB изображения размера 256x256 фильтром 5x5 необходимо выполнить ~5млн операций умножения
- на практике операцию свертки сводят к перемножению матриц
- для вычисления произведения матриц используют специальные библиотеки с эффективной реализацией

# Реализация операции свертки - img2col

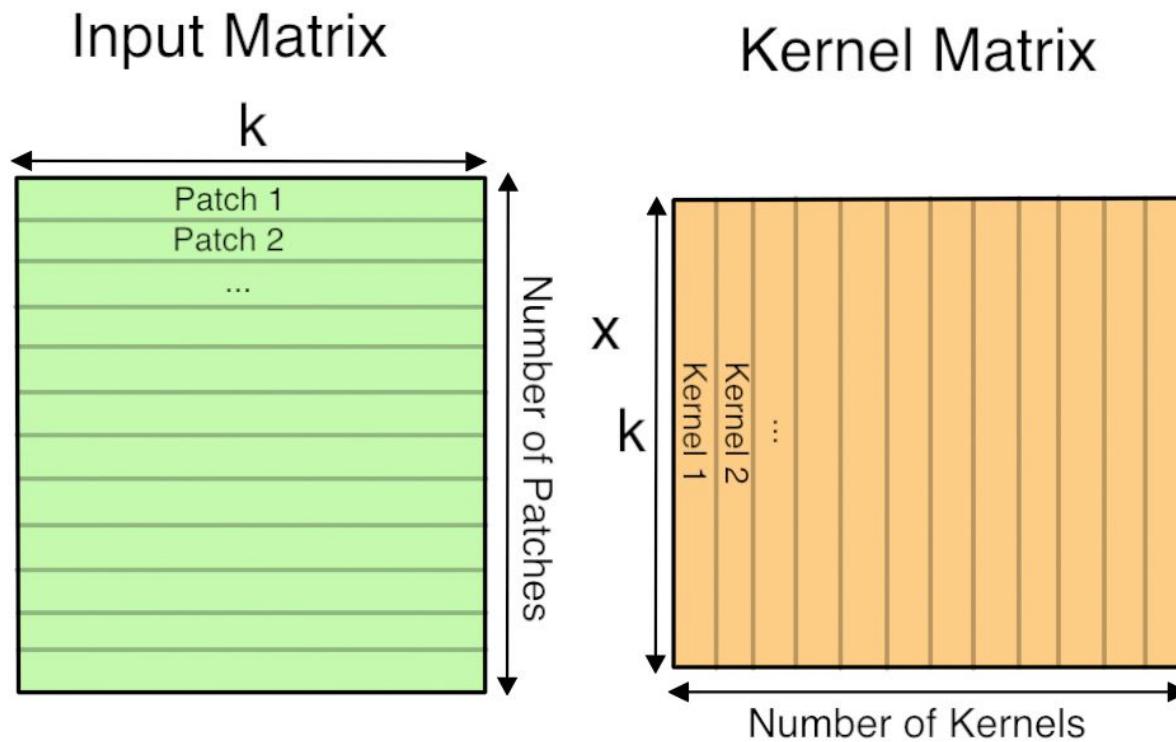
Input Image



im2col  
=>



# Реализация операции свертки



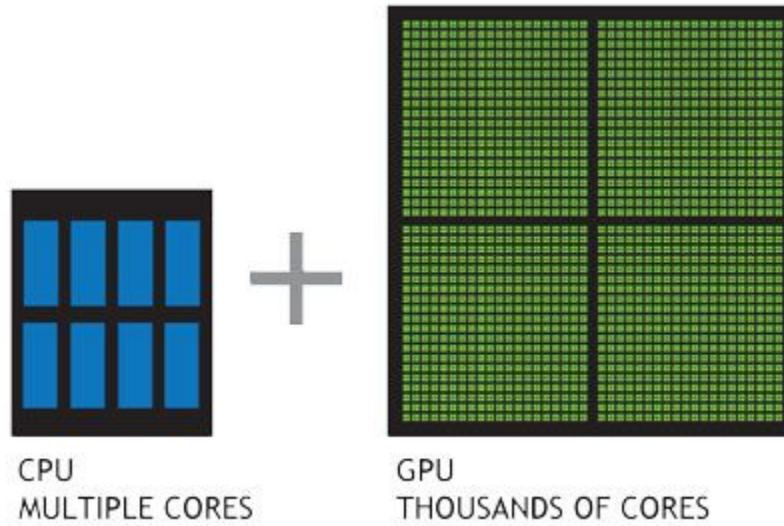
# Реализации матричных операций

- [BLAS - Basic Linear Algebra Subprograms](#)
- [ATLAS - Automatically Tuned Linear Algebra Software](#)
- [cuBLAS - NVIDIA cuBLAS](#)

# CPU vs GPU



# Вычисления на CPU vs GPU

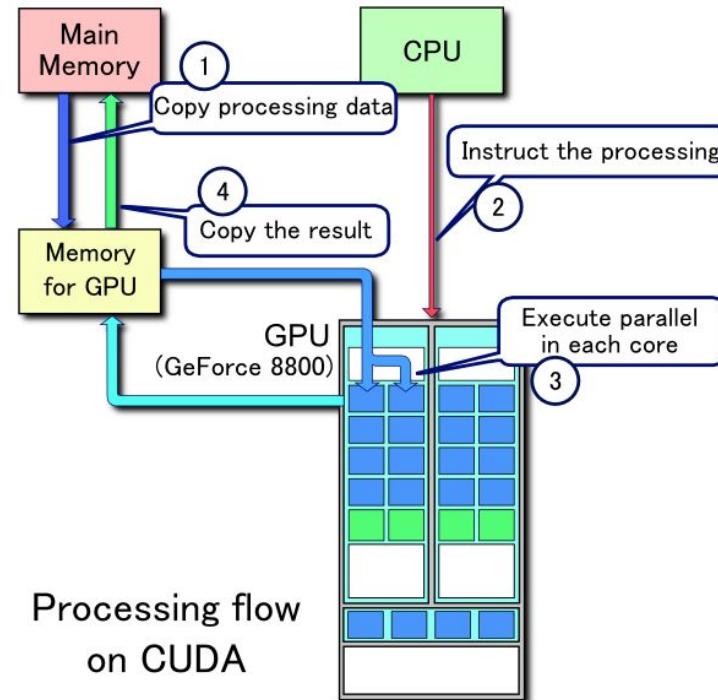


<https://www.youtube.com/watch?v=-P28LKWTzrl>

# Сравнение производительности



# Вычисления на GPU + CPU

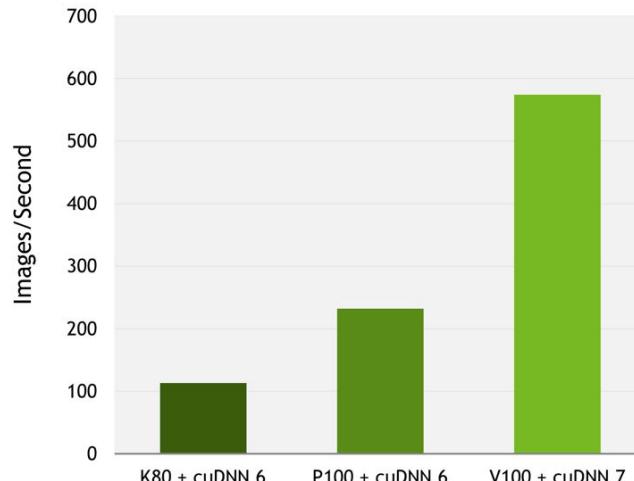


# Nvidia cuDNN - Deep Neural Network Library

- Оптимизация нейросетевых операций под Nvidia CUDA
- Реализованы основные слои (forward и backward pass)
- Оптимизированы операции над тензорами

# Nvidia cuDNN - Deep Neural Network Library

2.5x Faster Training of CNNs



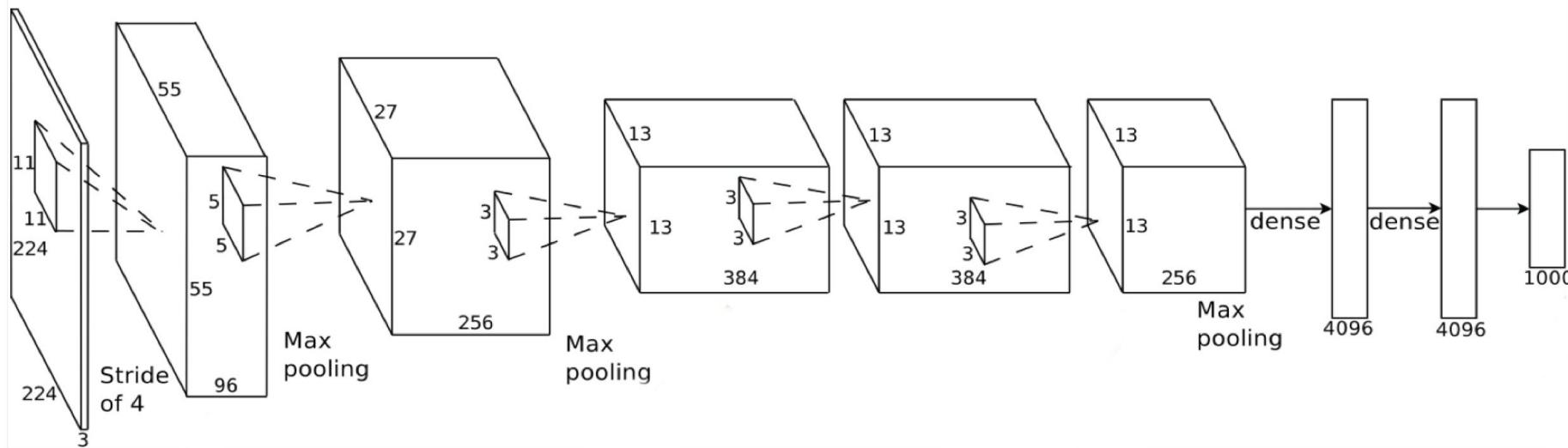
Caffe2 performance (images/sec), Tesla K80 + cuDNN 6  
(FP32), Tesla P100 + cuDNN 6 (FP32), Tesla V100 + cuDNN  
7 (FP16, pre-release H/W and S/W). ResNet50, Batch size: 64

# Визуализация сверточных сетей

# Визуализация сверточных сетей

- визуализация областей изображения с максимальной активацией
- кластеризация изображений по признакам полносвязного слоя ([t-SNE](#))
- визуализация сверточных фильтров
- image optimization
- deep dream
- style transfer

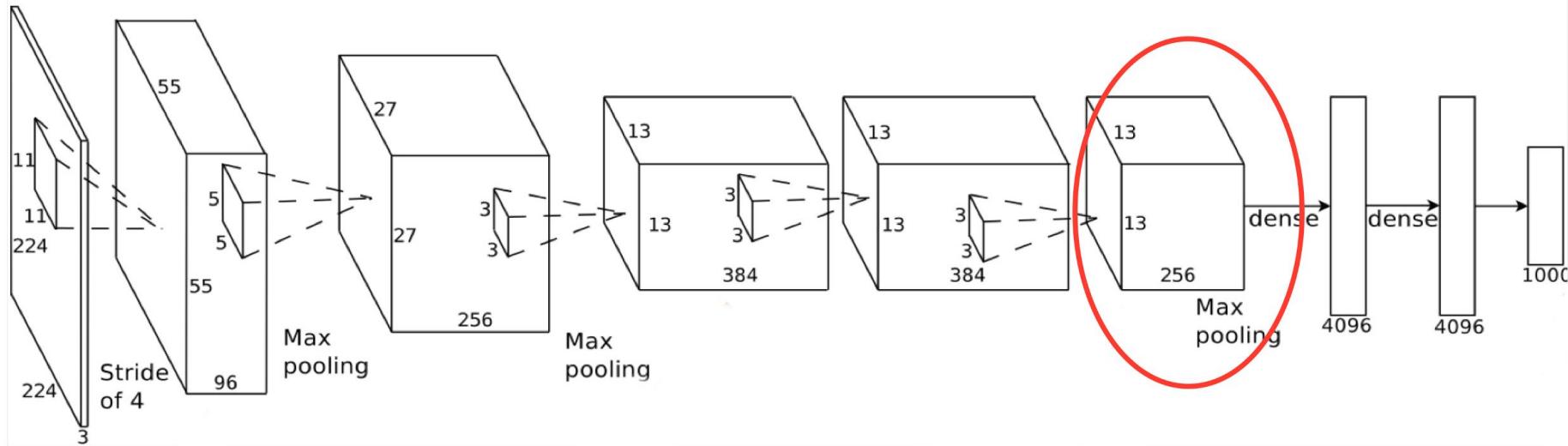
# Визуализация сверточных сетей



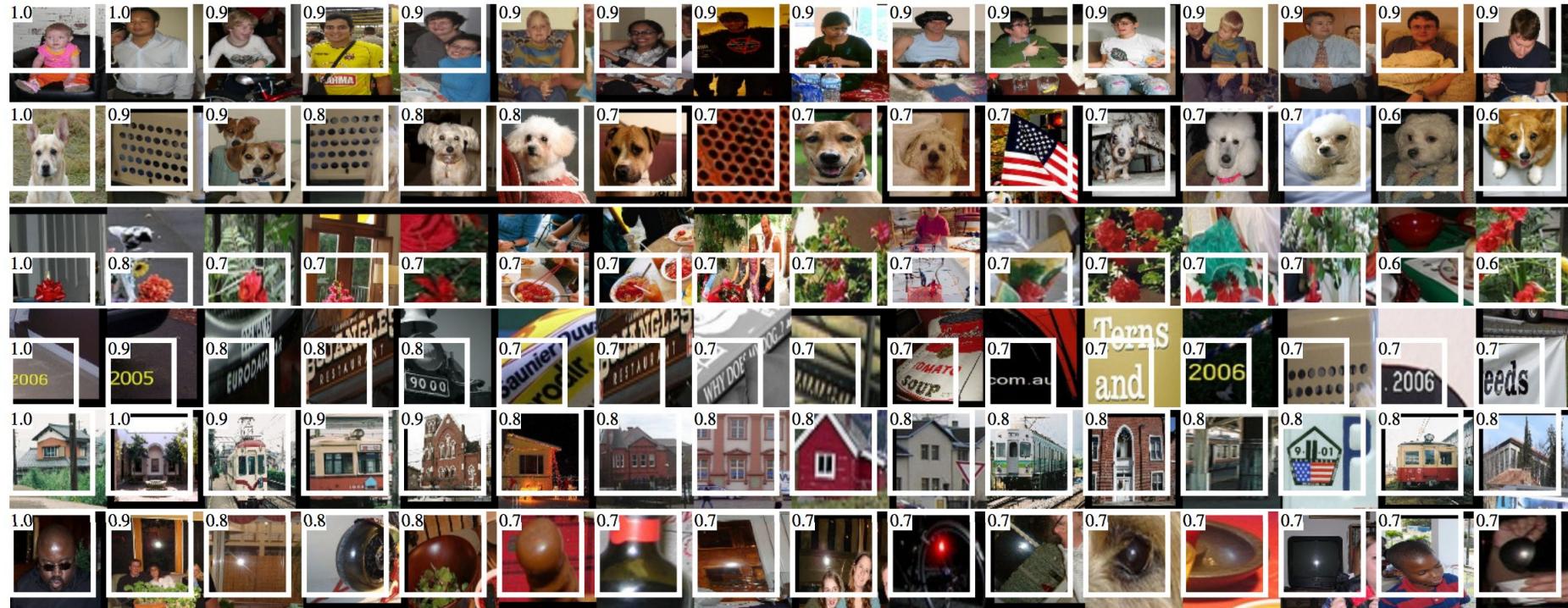
# Визуализация активаций R-CNN

- анализируем предобученную нейронную сеть
- рассматриваем активации на выходе последнего Pooling слоя
- визуализируем области на изображении соответствующие максимальными значениями активаций

# Визуализация активаций R-CNN



# Визуализация активаций R-CNN

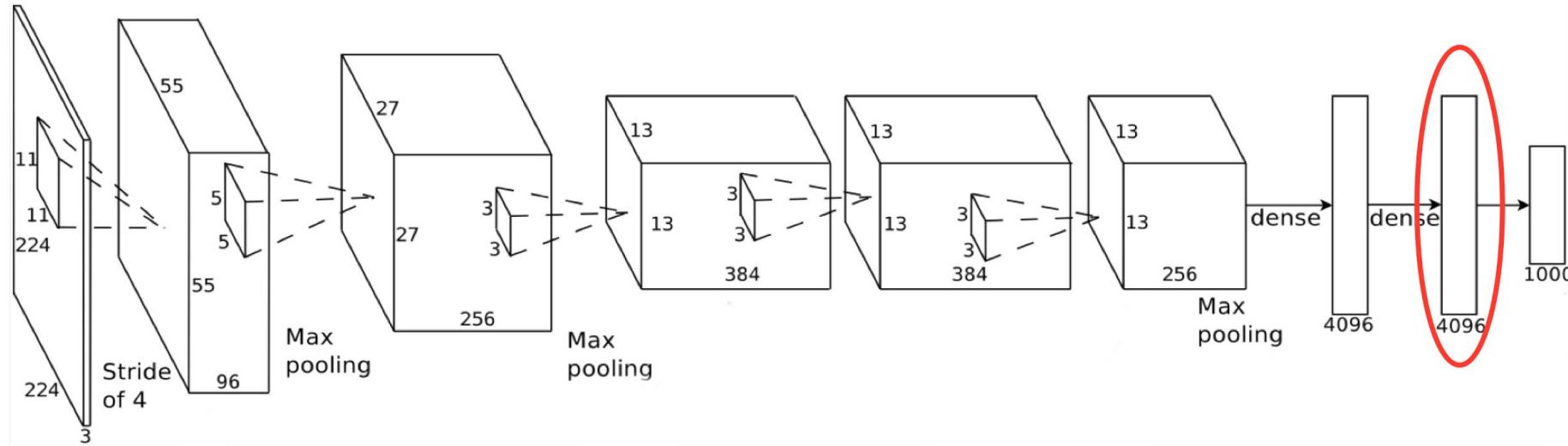


Rich feature hierarchies for accurate object detection and semantic segmentation

# Кластеризация по признакам FC слоя

- анализируем векторное представление изображение на выходе последнего полносвязного слоя
- понижаем размерность векторного представления до 2 алгоритмом t-SNE
- кластеризуем изображения

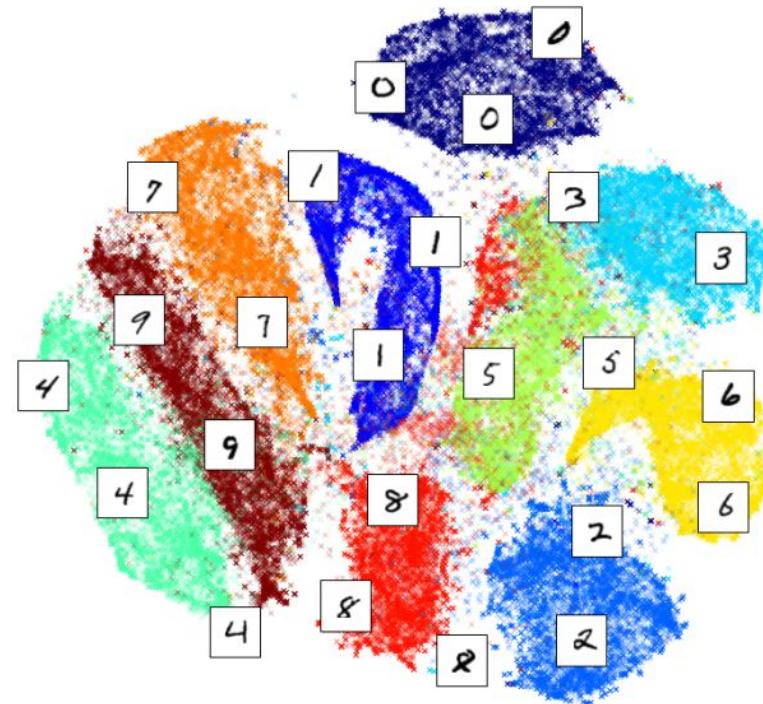
# Кластеризация по признакам FC слоя



# Кластеризация MNIST (FC + t-SNE)

MNIST dataset

Two-dimensional embedding of 70,000 handwritten digits with t-SNE



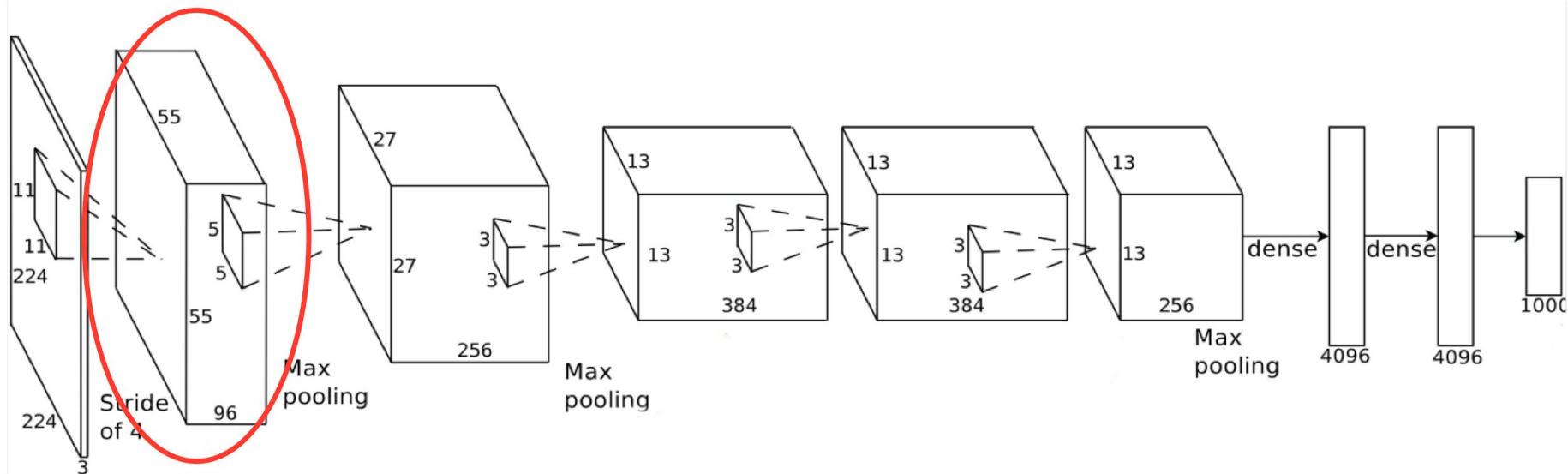
# Кластеризация ImageNET (FC + t-SNE)



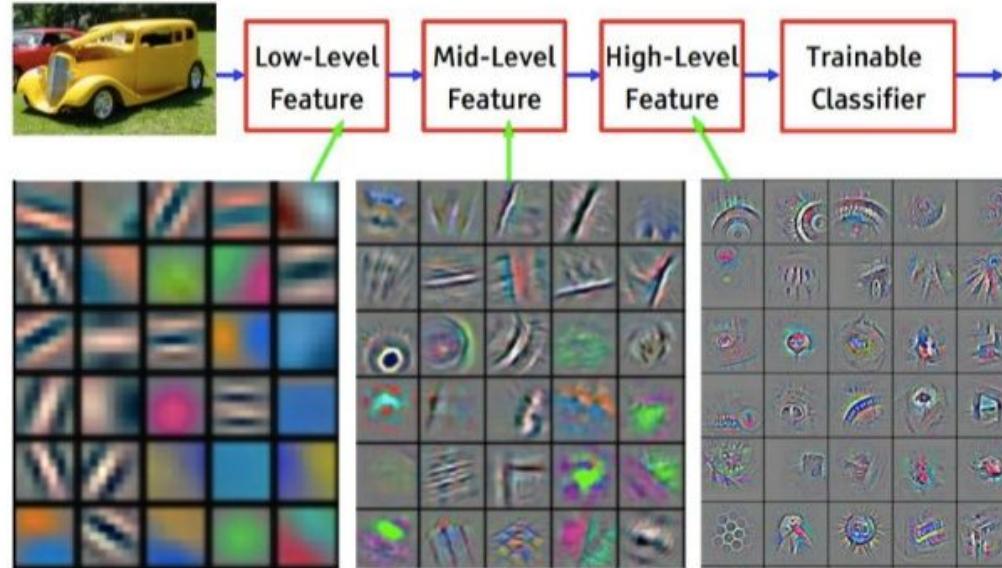
# Визуализация сверточных фильтров

- веса фильтров визуализируем как интенсивность
- рассматриваем фильтры на различных слоях
- на первом слое фильтры имеют простые геометрические формы
- с увеличением глубины, фильтры выучивают более сложные паттерны и визуально становятся похожи на части объектов

# Визуализация сверточных фильтров



# Визуализация сверточных фильтров



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Image optimization

- выбираем предобученную сеть
- выбираем класс и фиксируем его на выходном слое
- решаем обратную задачу - подбираем такое изображение, которое максимизирует вес выбранного класса

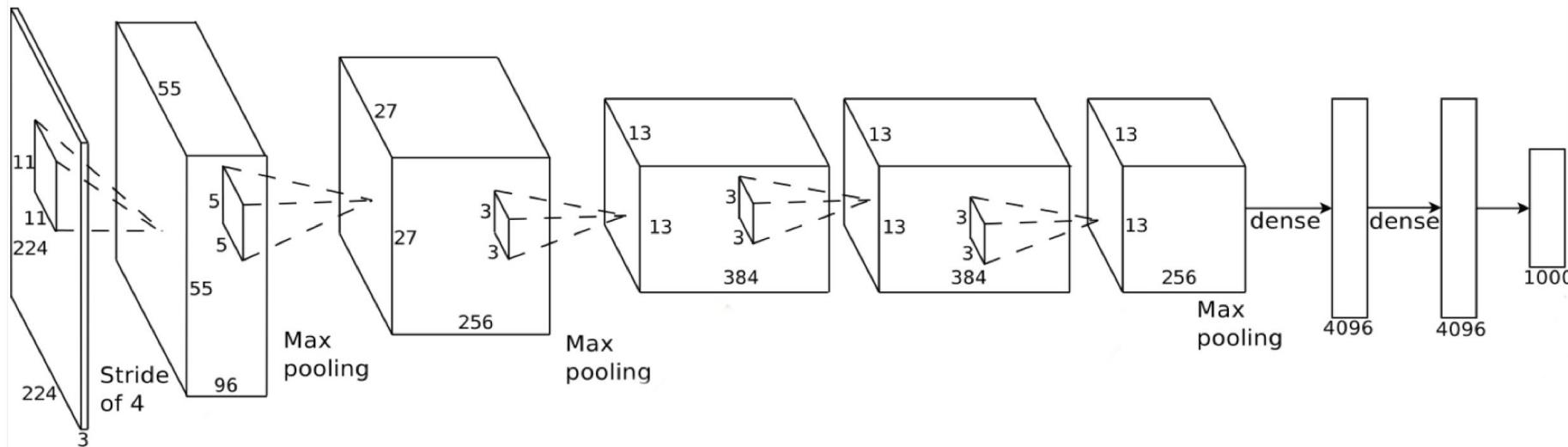
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2,$$

I - изображение

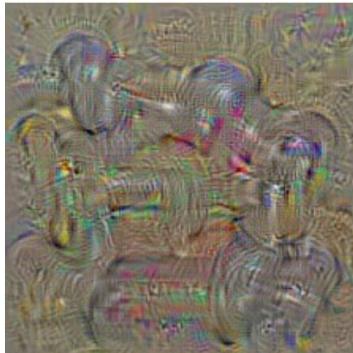
Sc - вес выбранного класса

lambda - параметр регуляризации

# Image Optimization



# Image optimization



dumbbell



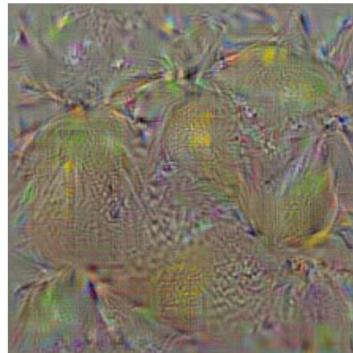
cup



dalmatian



bell pepper



lemon

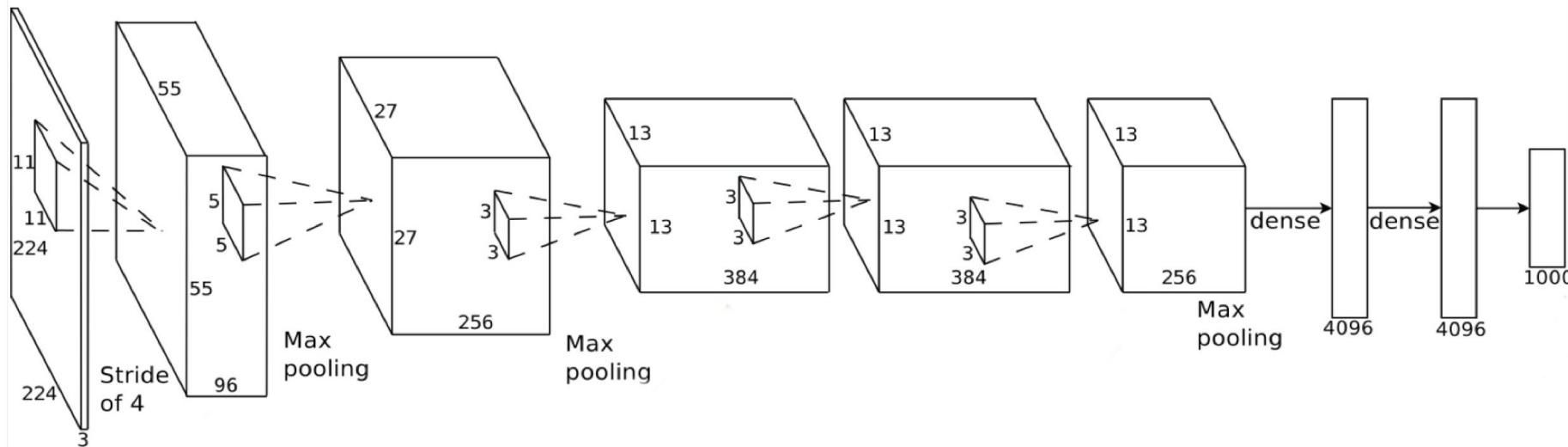


husky

# Deep Dream

- выбираем предобученную сеть
- фиксируем интересующий нас сверточный слой
- вычисляем активации на выбранном слое
- распространяем обратно значения активаций как значения градиента
- обновляем изображение с учетом полученных градиентов и повторяем итерацию

# Deep Dream



# Deep Dream

- результат будет зависеть от того, на каких данных обучалась нейронная сеть
- детали рисунка будут зависеть от глубины слоя с которого распространяются активации

# Deep Dream



Horizon



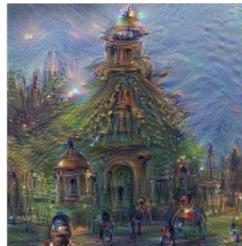
Trees



Leaves



Towers & Pagodas



Buildings



Birds & Insects

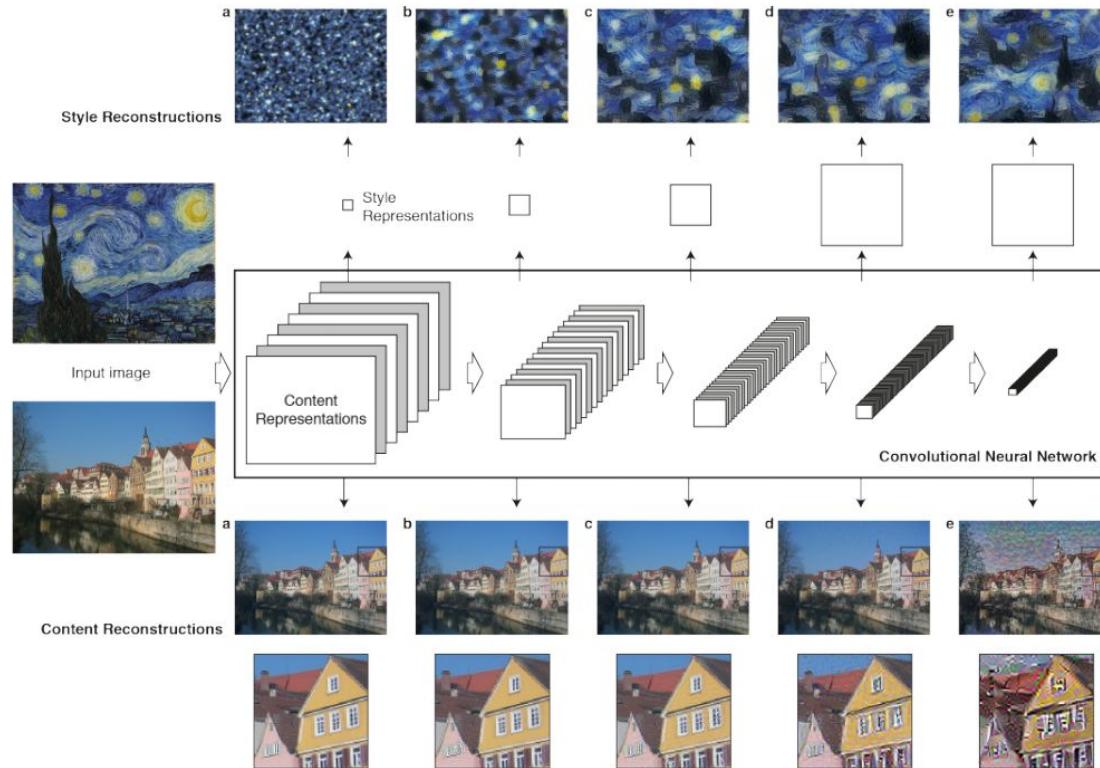
<https://github.com/google/deepdream>

<https://www.youtube.com/watch?v=DgPaCWJL7XI>

# Style Transfer

- сгенерируем такое изображение, которое максимально похоже на исходное
- при этом, стиль сгенерированного изображения должен быть похож на стиль референсного изображения
- активации сгенерированного изображения должны быть близки к активациям исходного
- ковариационная матрица фильтров сгенерированного изображения должна быть близка к матрице референсного изображения

# Style Transfer



# Style Transfer - Content Loss

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

F - активация пикселя i,j на слое l исходного изображения

P - активация пикселя i,j на слое l сгенерированного изображения

# Style Transfer - Style Loss

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l$$

i, j - индексы фильтров

k - номер активации

F<sub>k</sub> - значение активации k-го пикселя

G, A - матрица ковариации фильтров для заданного слоя

N - число фильтров в слое

M - число активаций (размер изображения на выходе сверточного слоя)

E - мера расхождения в стиле

w - вес ошибки для заданного слоя

L - число слоев

# Style Transfer - Total Loss

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

р - исходное изображение

а - изображение с референсным стилем

х - сгенерированное изображение

alpha, beta - параметры обучения

# Style Transfer

## 1 Upload photo

The first picture defines the scene you would like to have painted.



## 2 Choose style

Choose among predefined styles or upload your own style image.



## 3 Submit

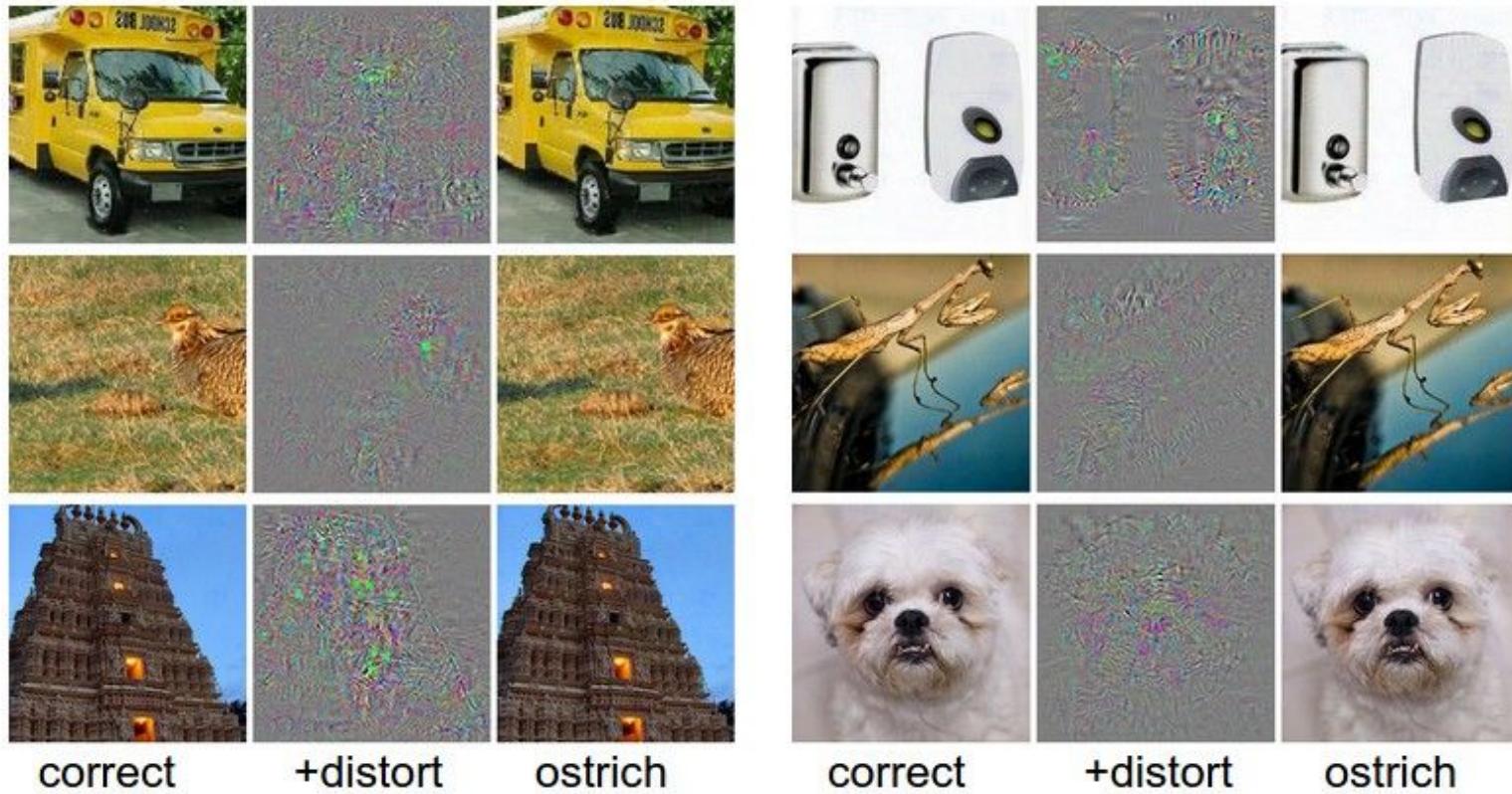
Our servers paint the image for you. You get an email when it's done.



<https://deepart.io>

# Как обмануть сверточную сеть

# Как обмануть сверточную сеть



# Резюме

- в отличие от полносвязных, сверточные сети содержат меньше параметров, как результат менее склонны к переобучению
- основные параметры свертки: размер фильтра,
- основной операцией в процессе работы сверточной нейронной сети является перемножение матриц
- GPU позволяет ускорить вычисления за счет параллелизации

# Резюме

- с увеличением глубины слоя признаки становятся более общими (увеличивается область видимости изображения)
- для оценки стиля изображения можно использовать ковариационную матрицу фильтров для каждого слоя
- добавление специального шума может заставить нейронную сеть выдавать неверные ответы

# Полезные материалы

- [Convolutional Neural Networks \(CNNs / ConvNets\)](#)
- [Convolutional Neural Networks for Visual Recognition](#)
- [Gradient-based Learning Applied to Document Recognition](#)
- [Understanding CNN](#)
- [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#)