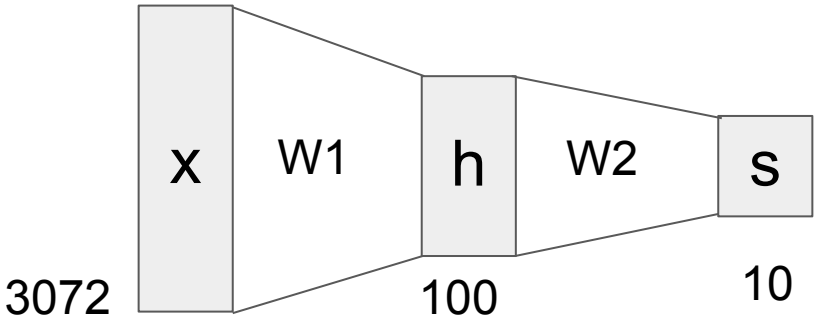
AI Community

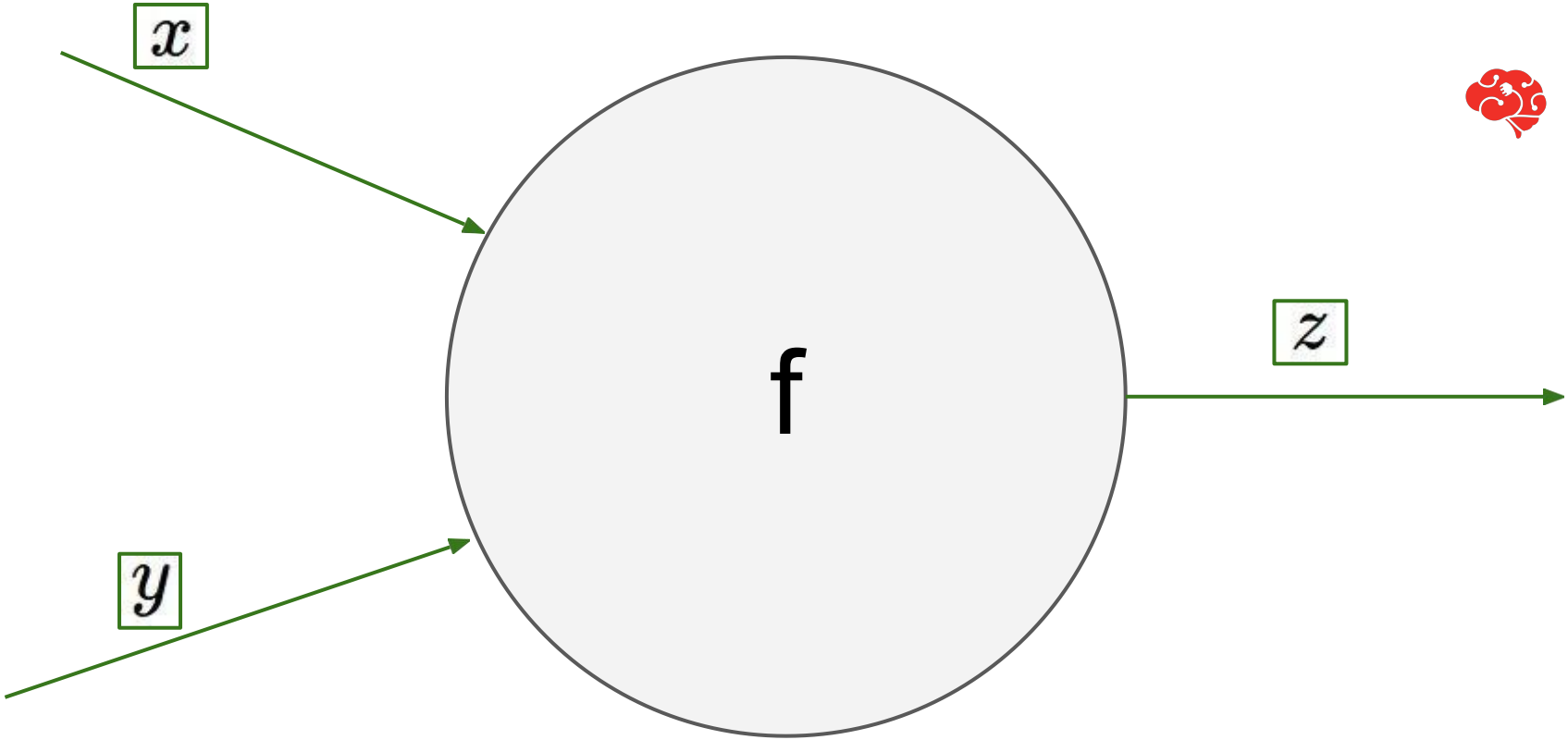
9. Свёрточные нейронные сети

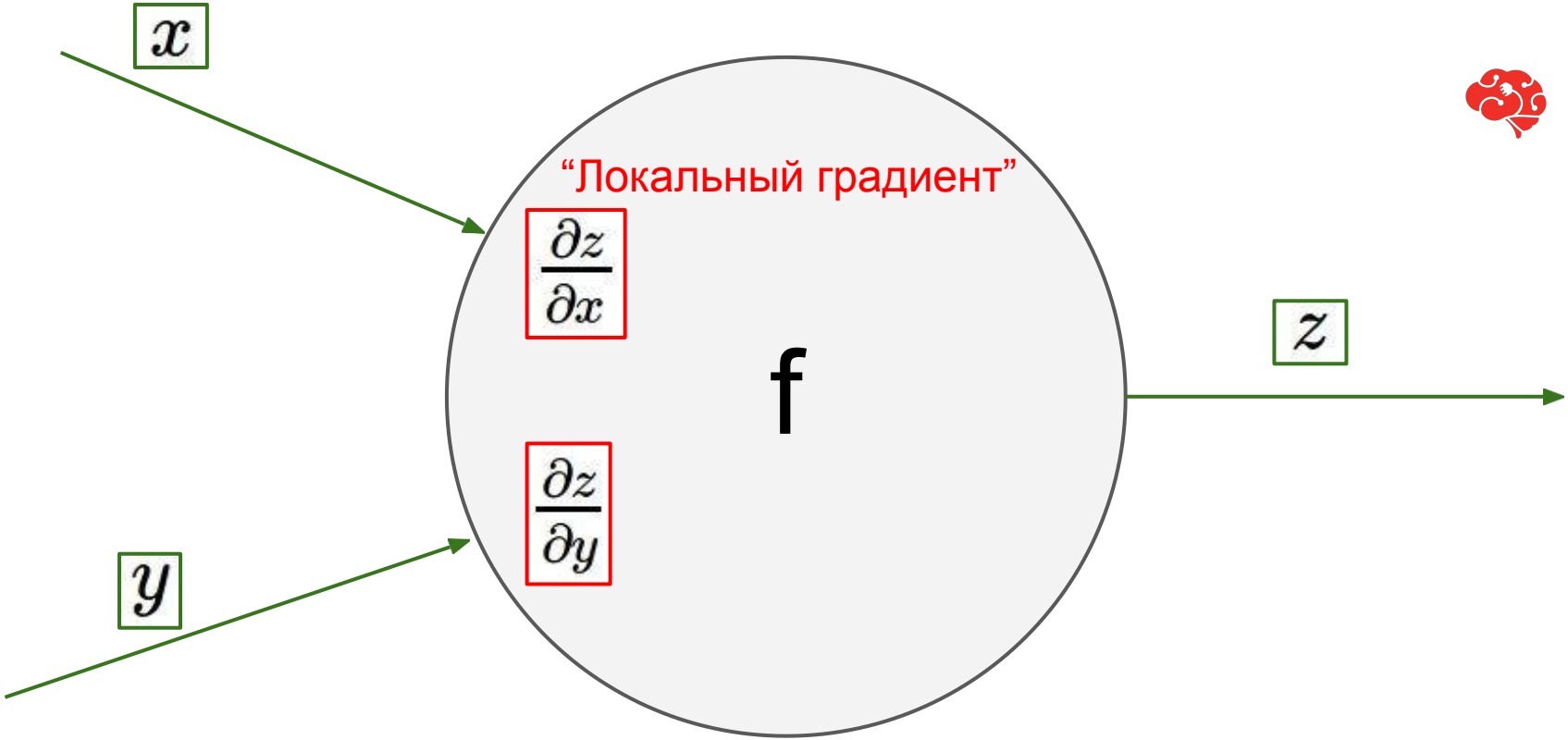
Нейронные сети

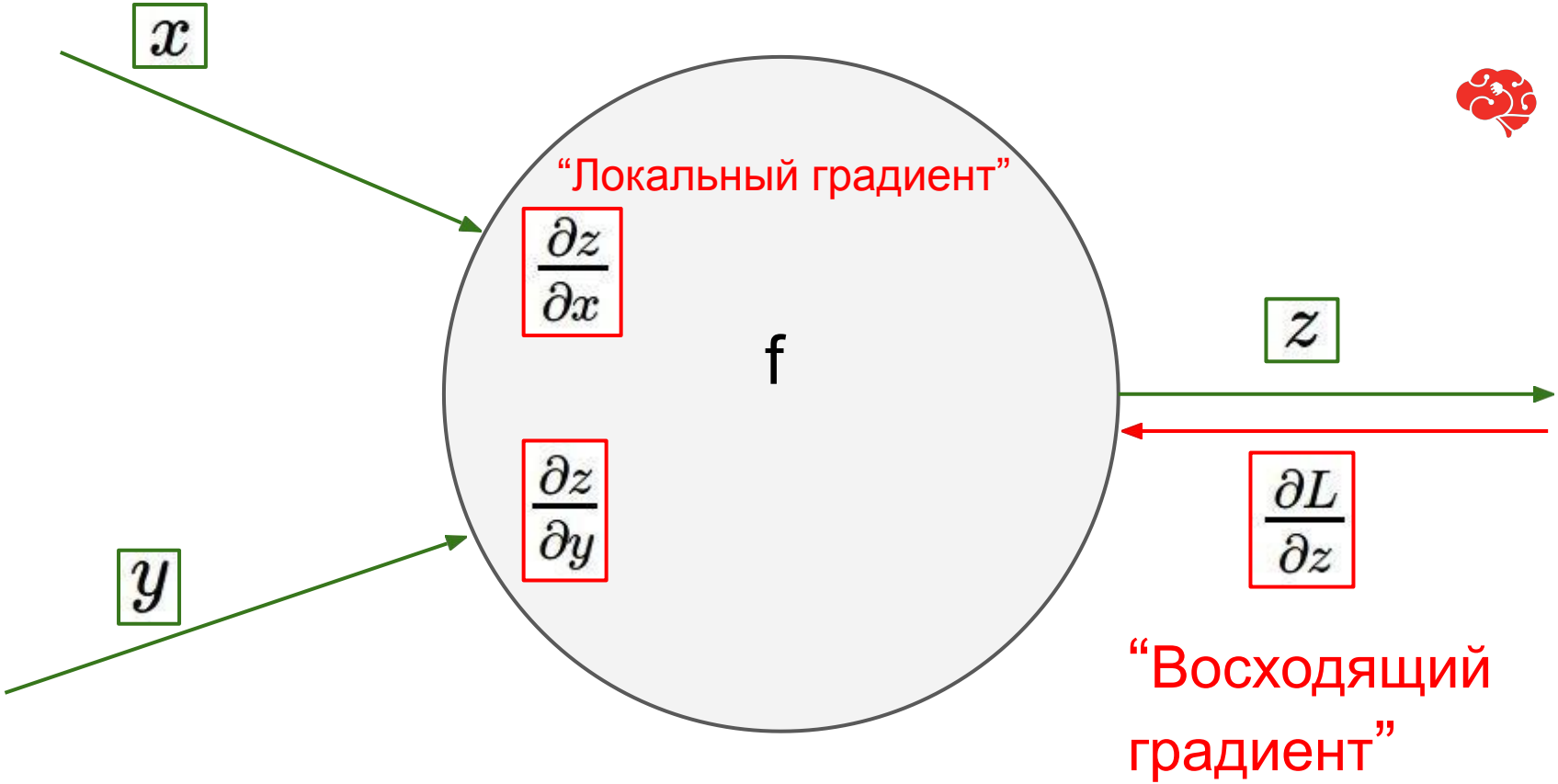
Линейная регрессия: $f = Wx$

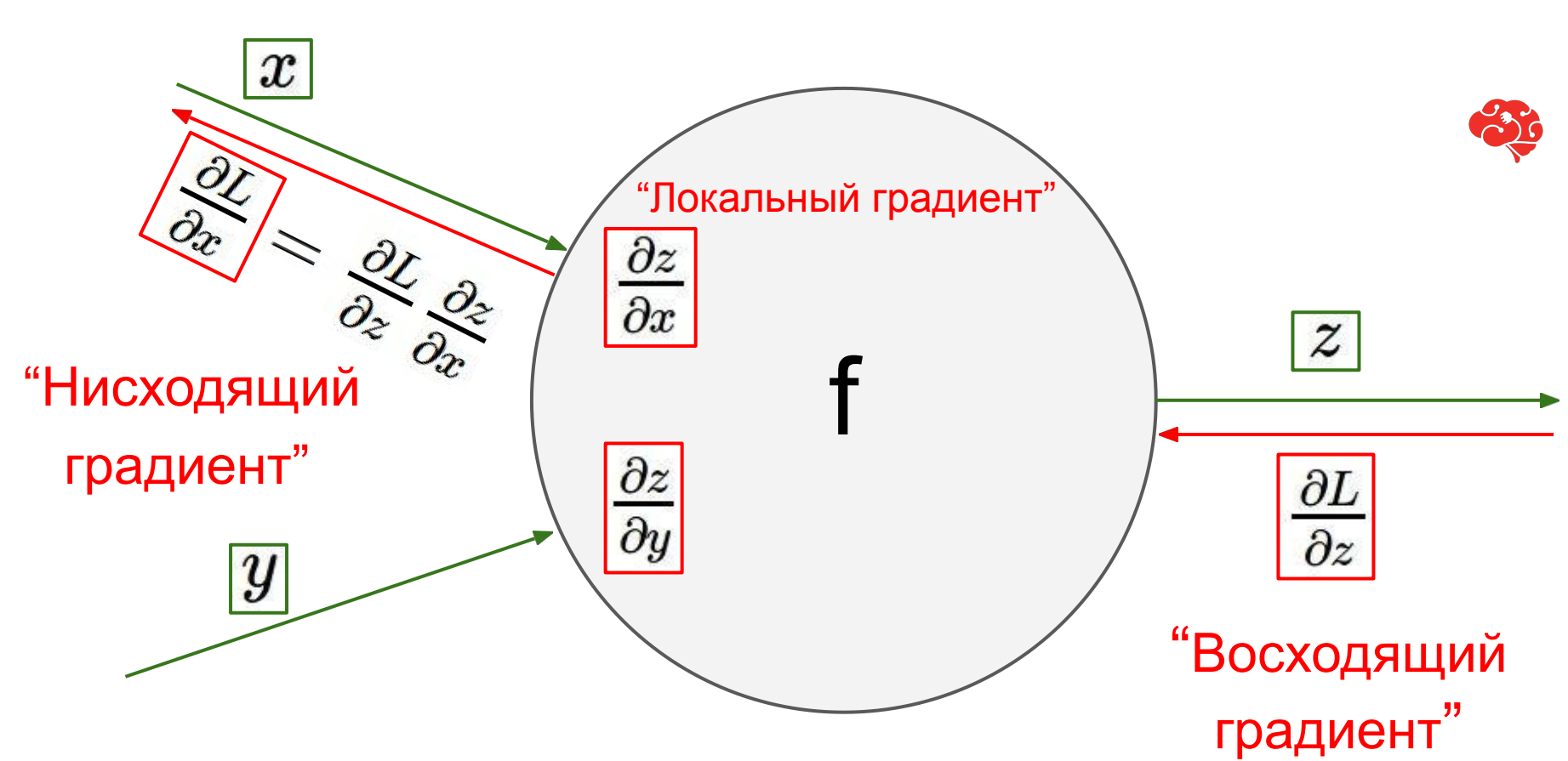
Двухслойная нейронная сеть $f = W_2 \max(0, W_1 x)$

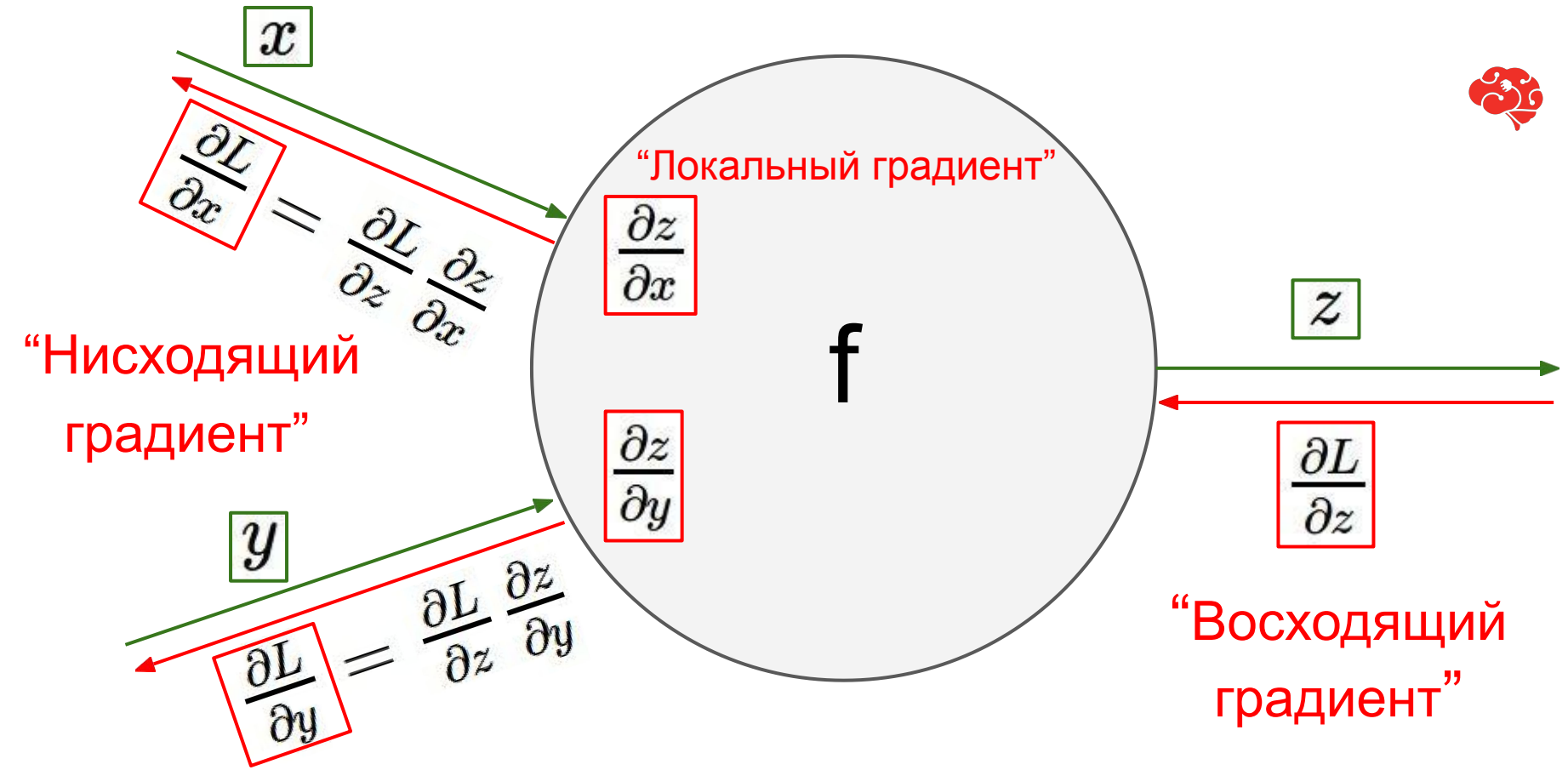


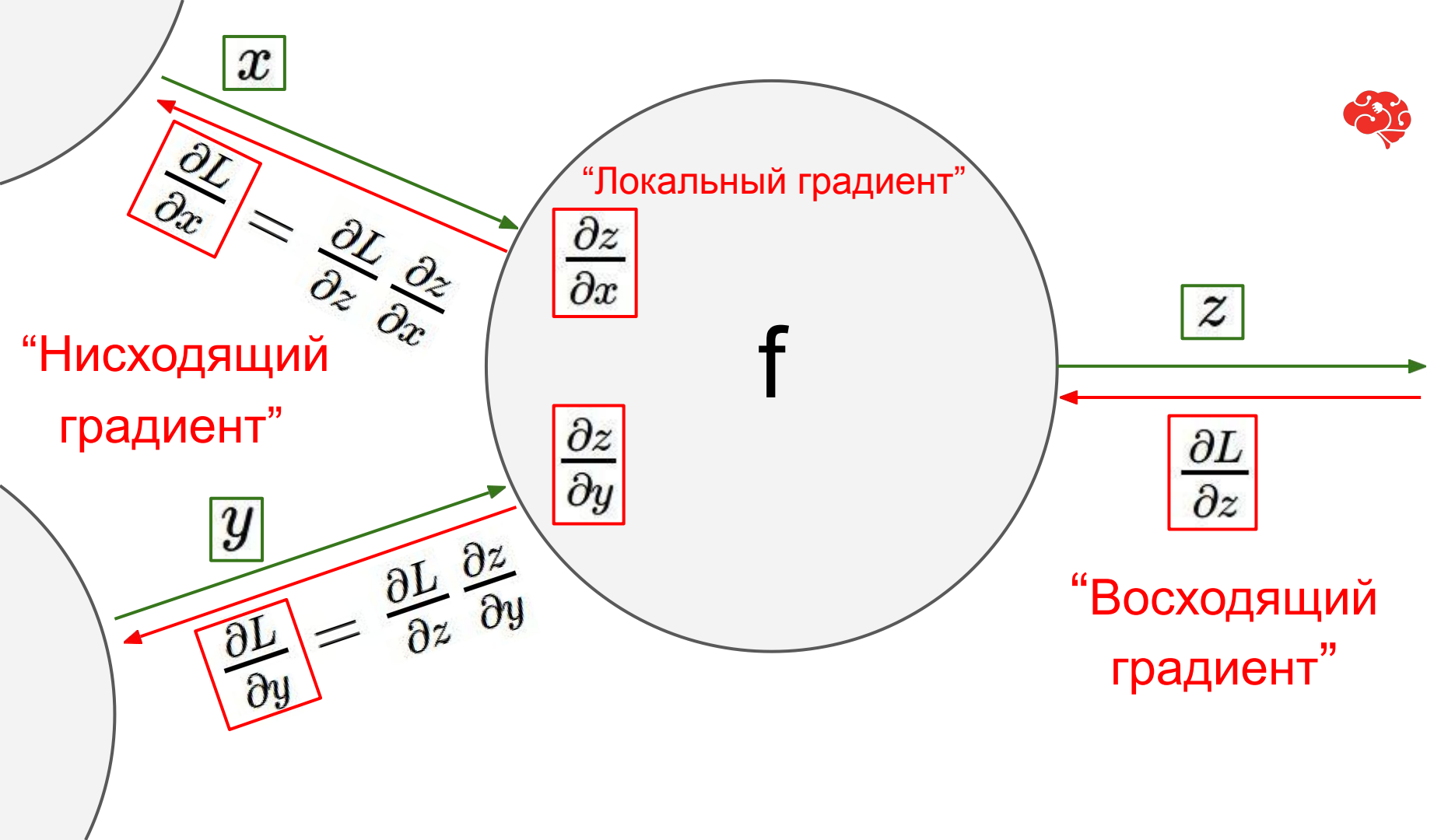

x
y
f
z


x
y
“Локальный градиент”
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z


x
y
“Локальный градиент”
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
“Восходящий градиент”

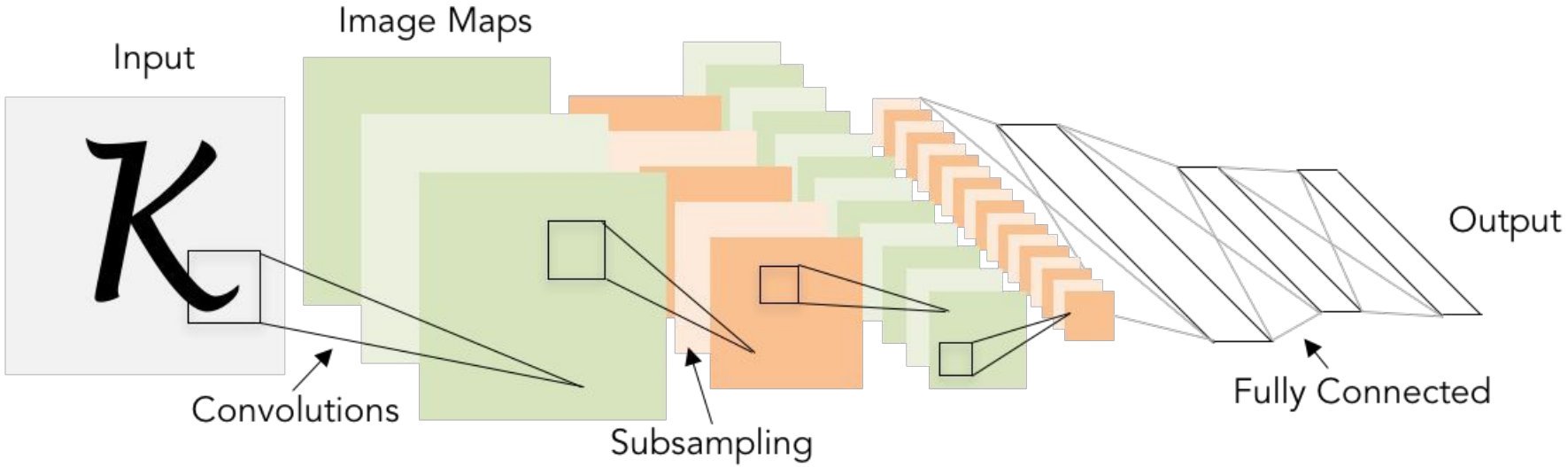

x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
“Нисходящий градиент”
y
“Локальный градиент”
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
“Восходящий градиент”


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
“Нисходящий градиент”
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
“Локальный градиент”
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
“Восходящий градиент”


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
“Нисходящий градиент”
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
“Локальный градиент”
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
“Восходящий градиент”

Свёрточные Нейронные сети

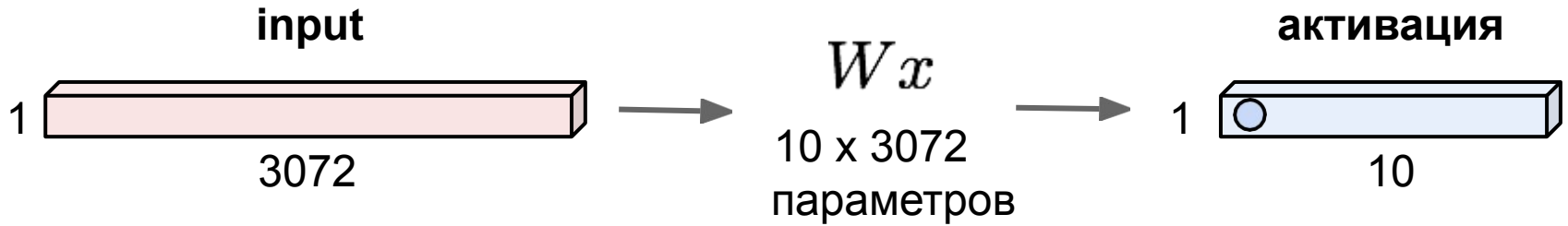
Свёрточные нейронные сети



LeNet-5 (Gradient-based learning applied to document recognition)
[LeCun, Bottou, Bengio, Haffner 1998]

Полносвязный слой

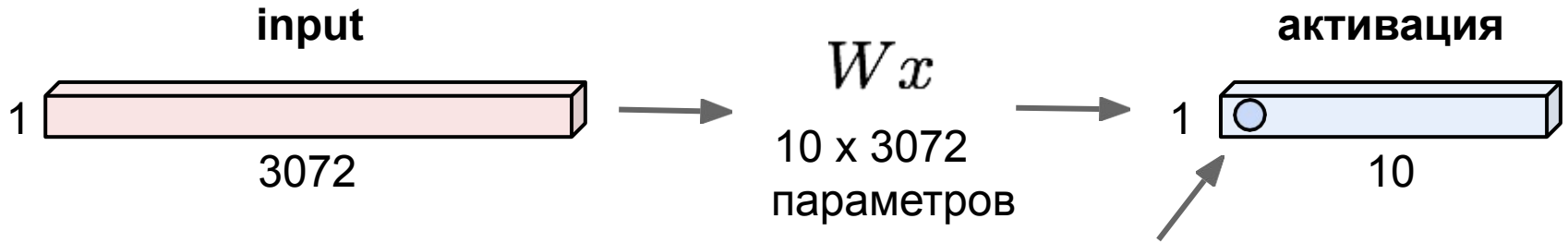
Для каждого входного признака представлен отдельный вектор параметров, переводящий входной вектор в другое пространство



Изображение размера 32x32x3
переводится в вектор размера 1x3072

Полносвязный слой

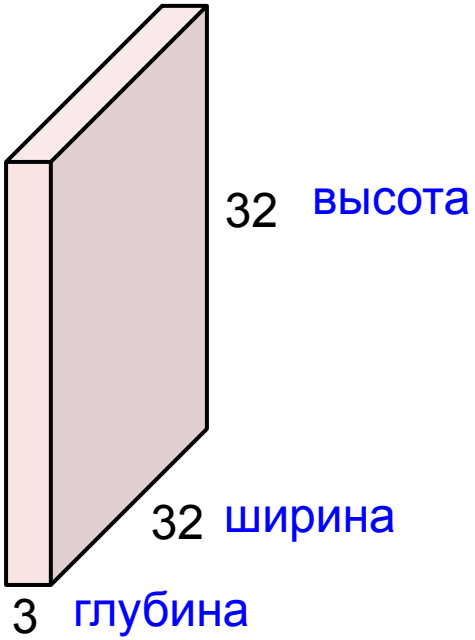
Для каждого входного признака представлен отдельный вектор параметров, переводящий входной вектор в другое пространство



Изображение размера 32x32x3
переводится в вектор размера 1x3072

Каждое число выходного вектора является скалярным произведением входного вектора и столбца матрицы весов (3072-мерное скалярное произведение + смещение)

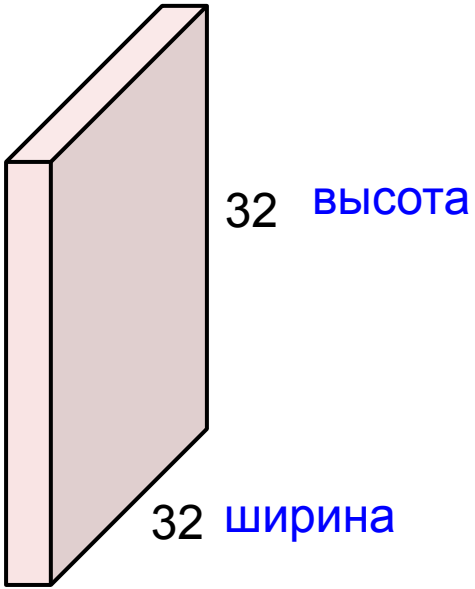
Полносвязный слой



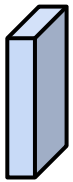
В свёрточных слоях изображение сохраняет свою пространственную форму

Свёрточный слой

Изображение 32x32x3

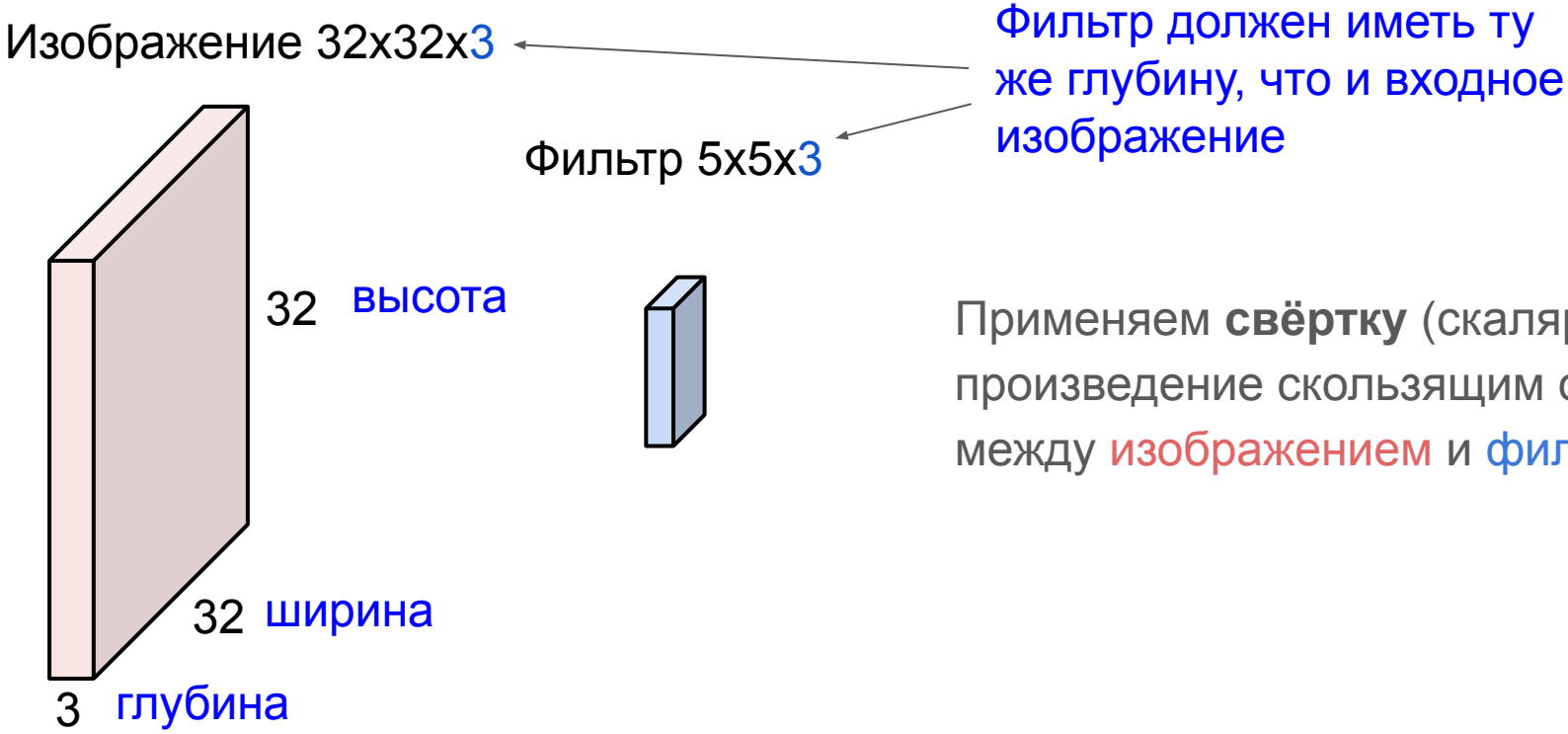


Фильтр 5x5x3



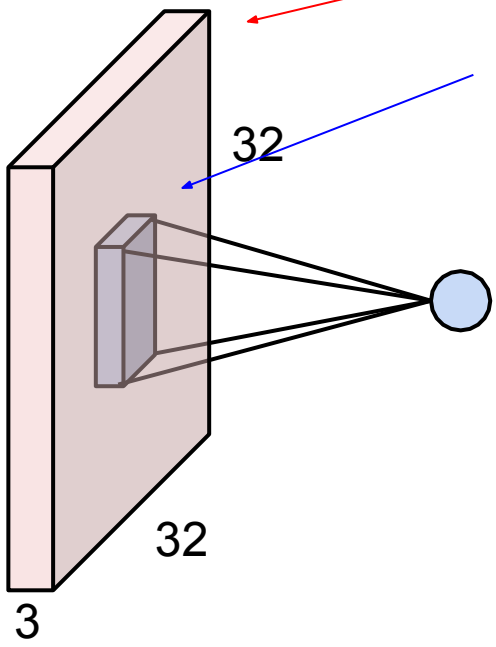
Применяем **свёртку** (скалярное произведение скользящим окном) между изображением и фильтром

Свёрточный слой



Применяем **свёртку** (скалярное произведение скользящим окном) между изображением и фильтром

Свёрточный слой



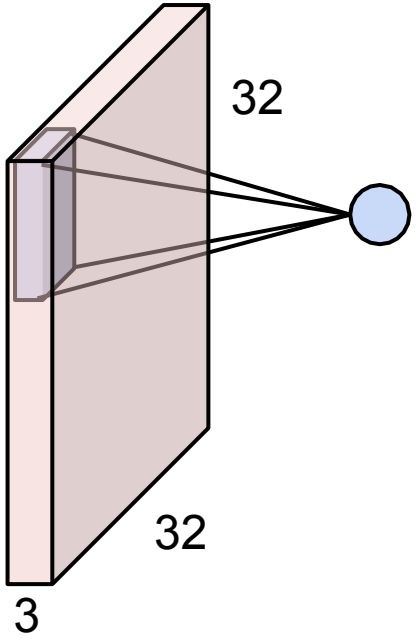
32x32x3 изображение

5x5x3 фильтр w

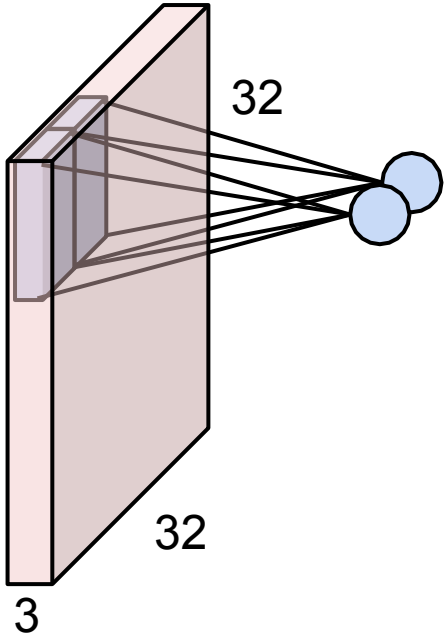
Каждое число выходного вектора является скалярным произведением фильтра и небольшого куска изображения размера 5x5x3 (5*5*3=75-мерное скалярное произведение + смещение)

$$w^T x + b$$

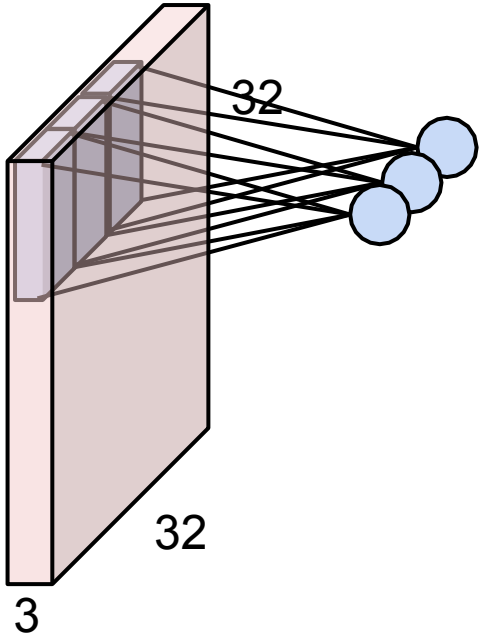
Свёрточный слой



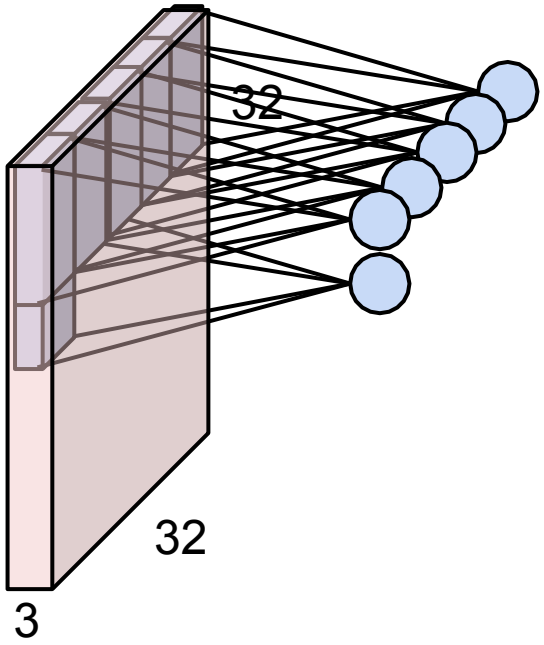
Свёрточный слой



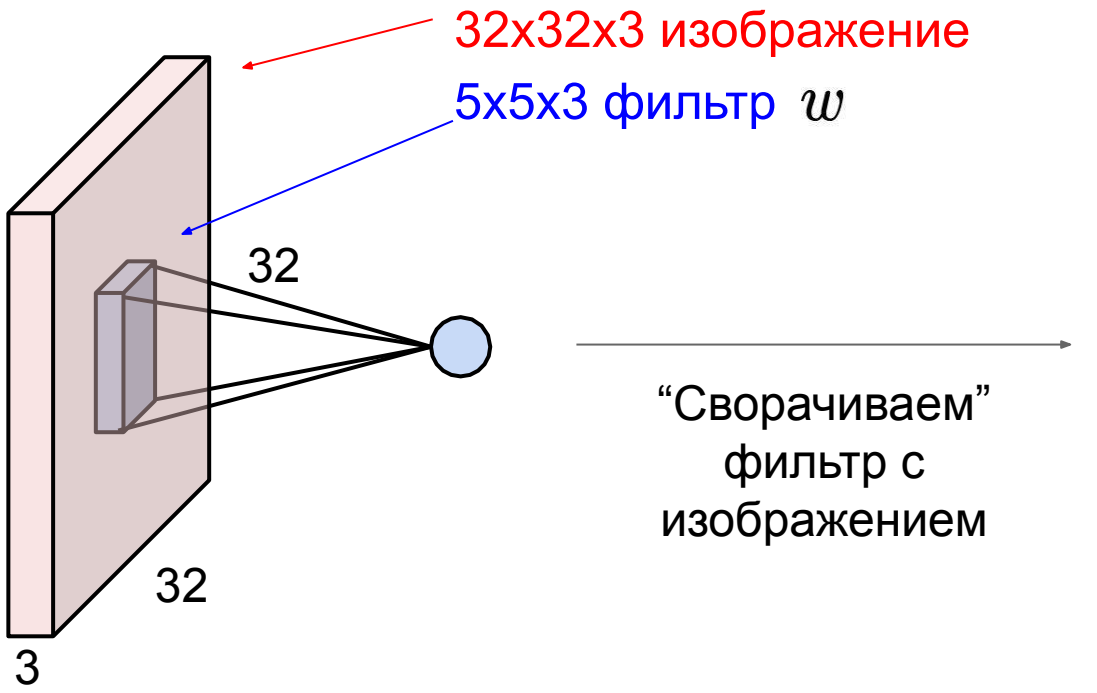
Свёрточный слой



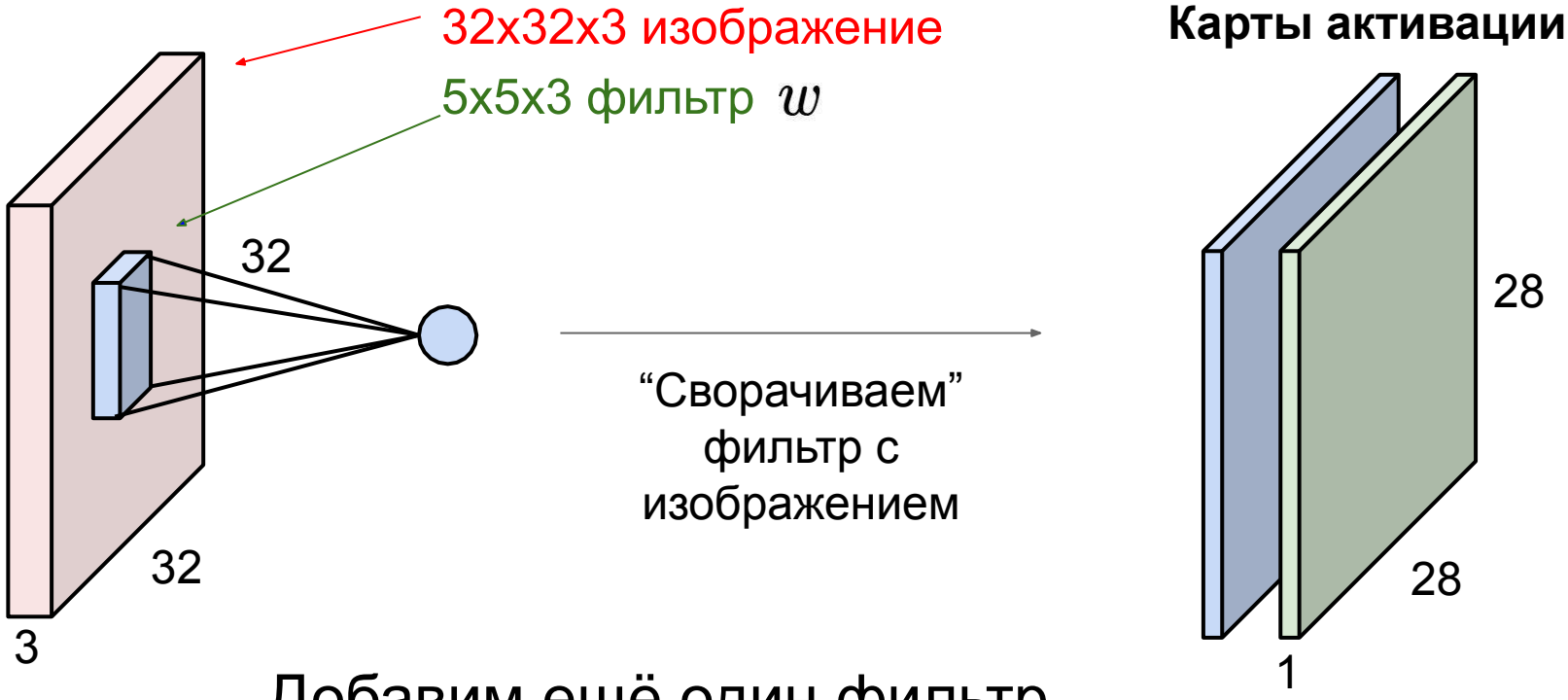
Свёрточный слой



Свёрточный слой

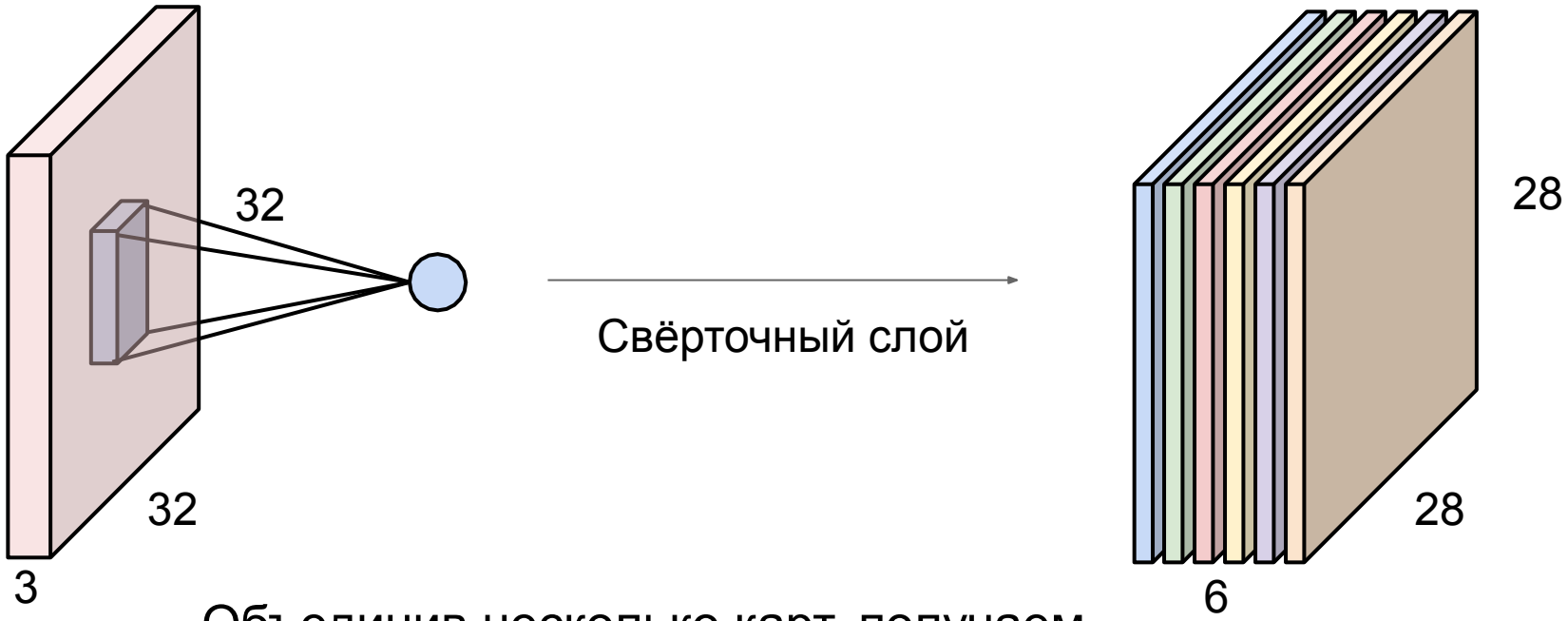


Свёрточный слой



Добавим ещё один фильтр

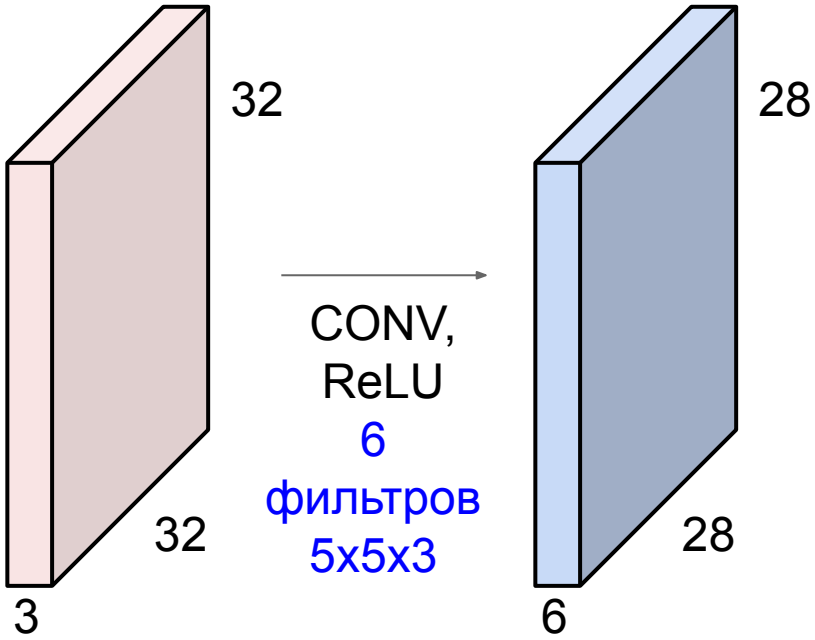
Свёрточный слой



Объединив несколько карт, получаем новое изображение размера 28x28x6

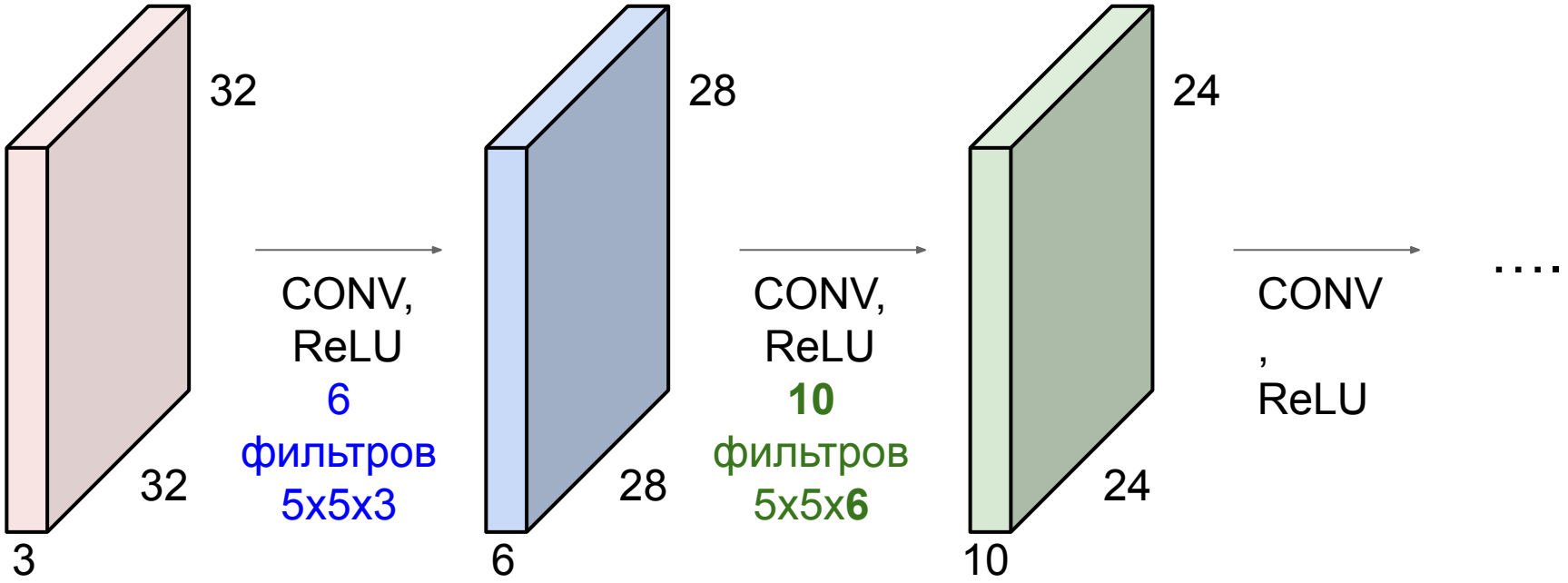
Свёрточная сеть

Свёрточная сеть является объединением несколький свёрточных слоёв, к каждому выходу которого применяется активация.

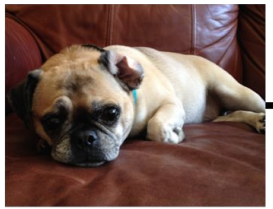


Свёрточная сеть

Свёрточная сеть является объединением несколький свёрточных слоёв, к каждому выходу которого применяется активация.



Свёрточная сеть

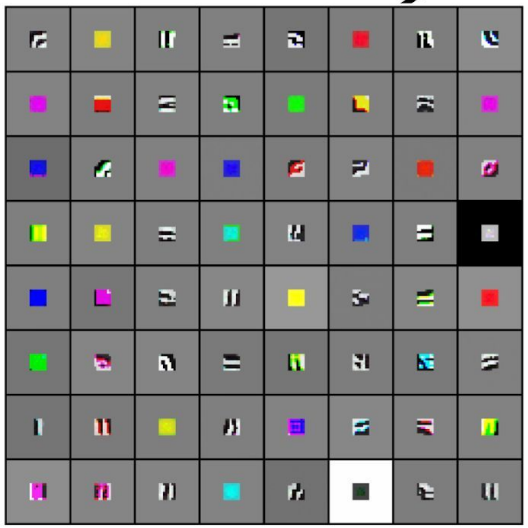


Low-level features

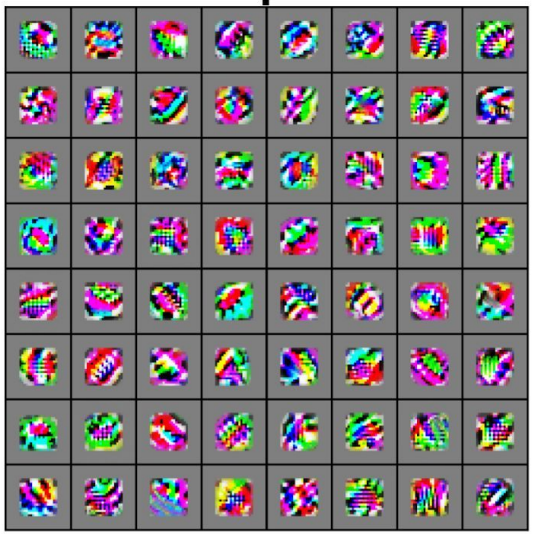
Mid-level features

High-level features

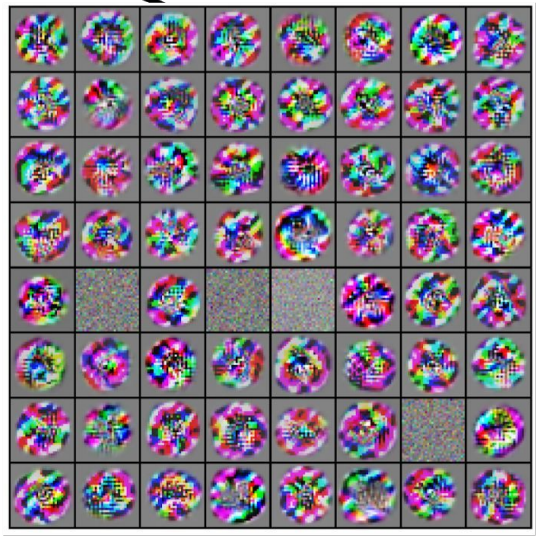
Linearly separable classifier



VGG-16 Conv1_1

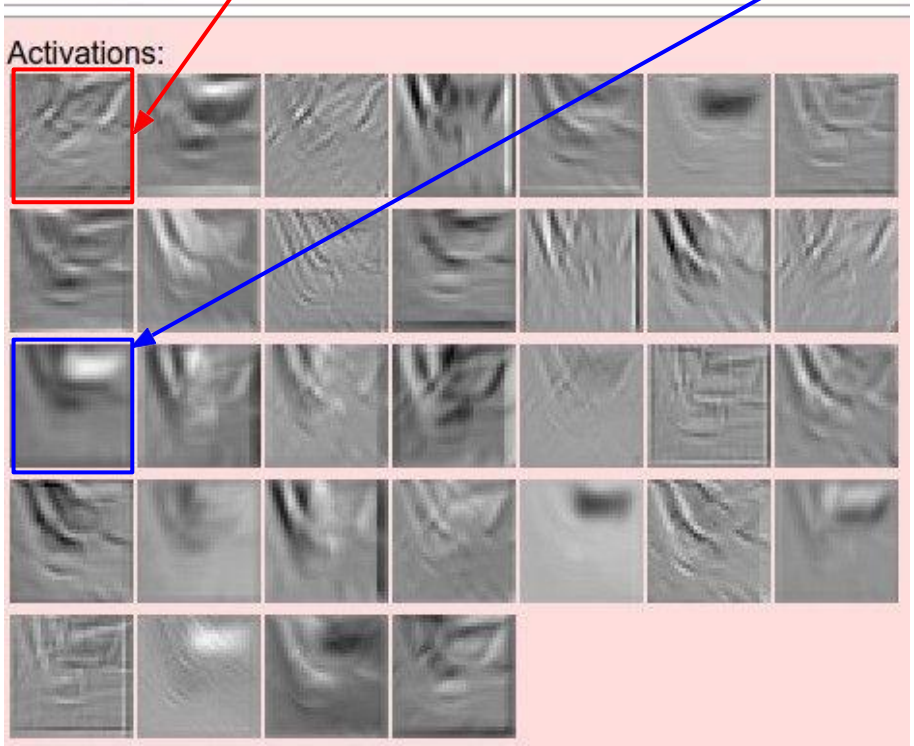


VGG-16 Conv3_2



VGG-16 Conv5_3

один фильтр =>
одна карта



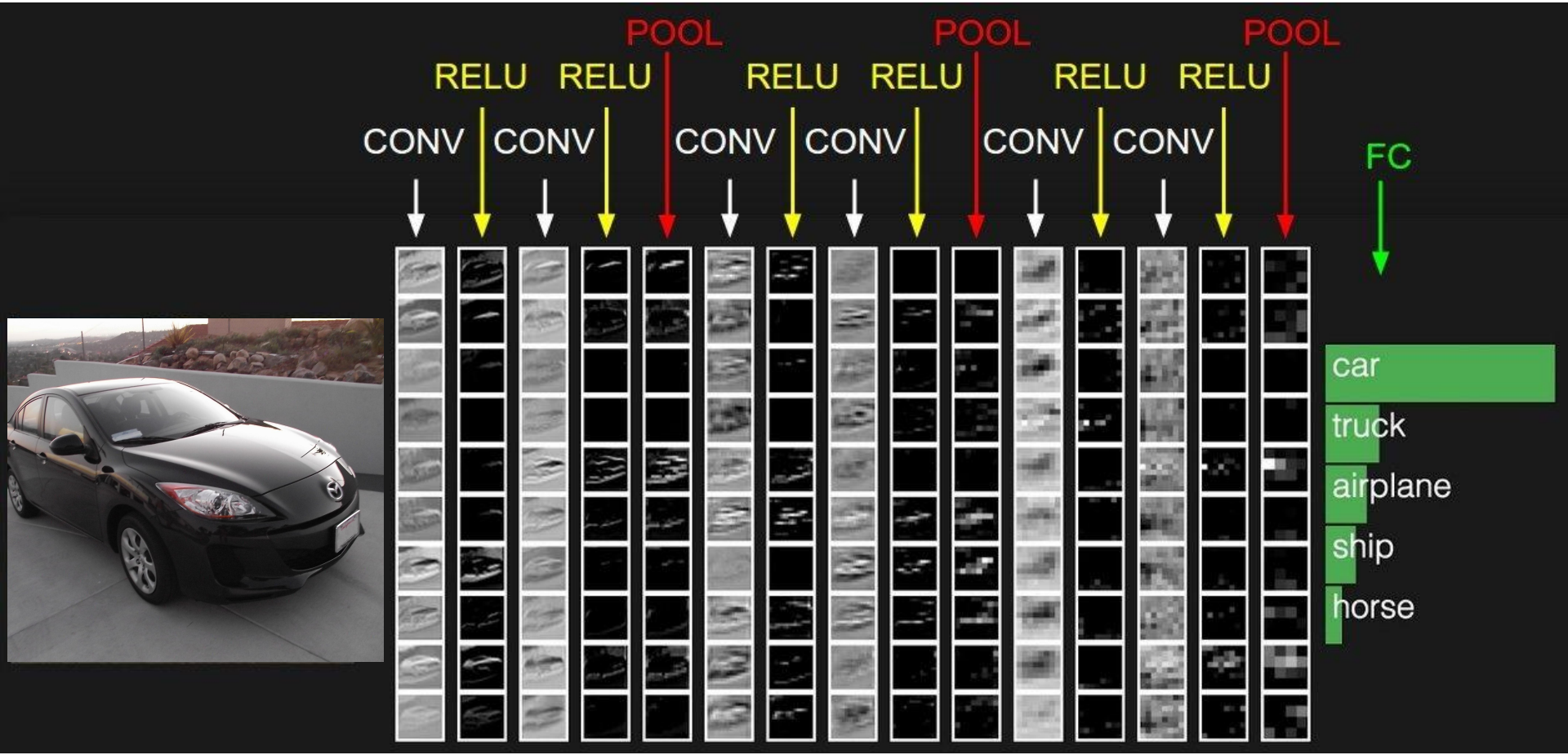
примеры фильтров 5x5

(32 всего)

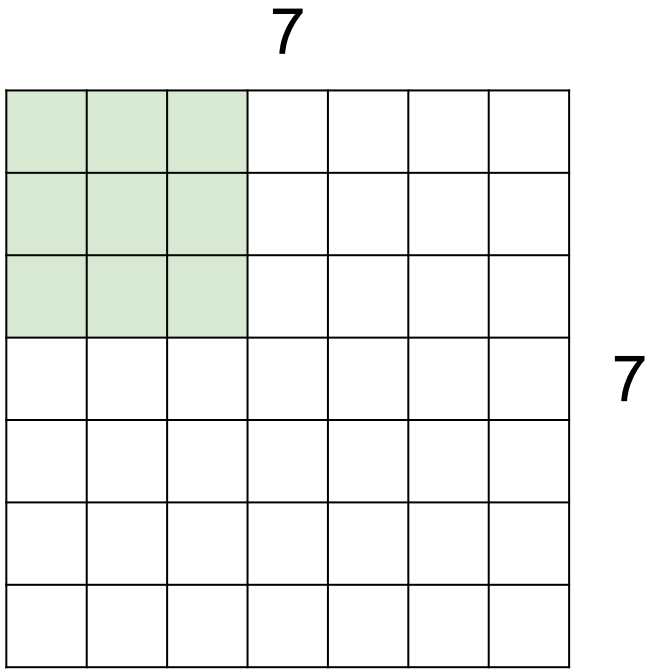
Слой называется свёрточным, так как это можно представить как свёртку двух сигналов:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

Поэлементное перемножение и сумма and фильтра и сигнала (изображения)


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Свёрточный слой. Подробный взгляд

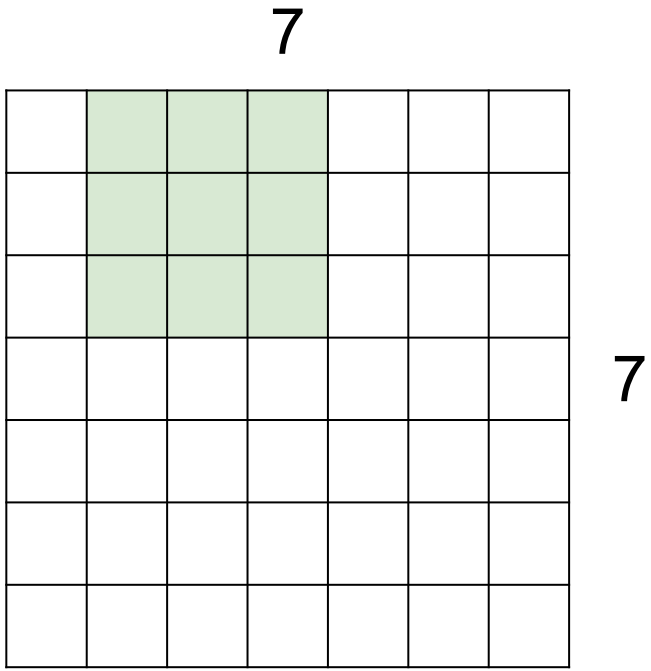


7x7 вход,

используем

фильтр 3x3

Свёрточный слой. Подробный взгляд

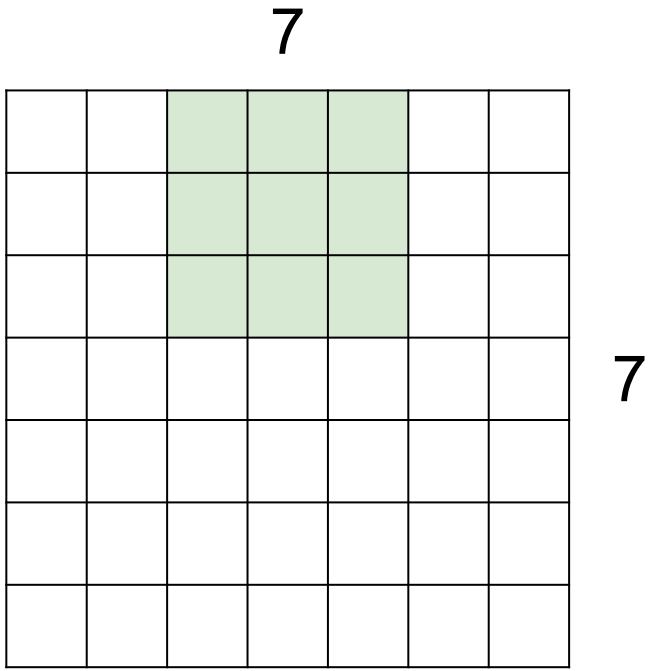


7x7 вход,

используем

фильтр 3x3

Свёрточный слой. Подробный взгляд

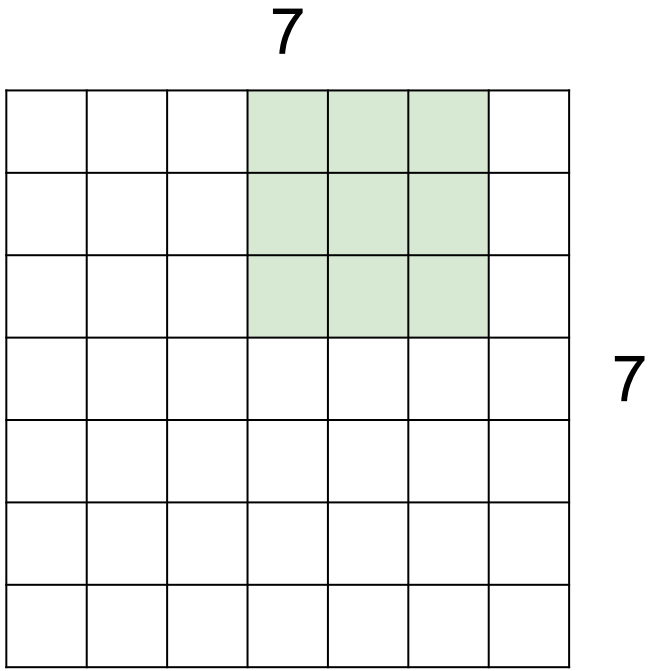


7x7 вход,

используем

фильтр 3x3

Свёрточный слой. Подробный взгляд

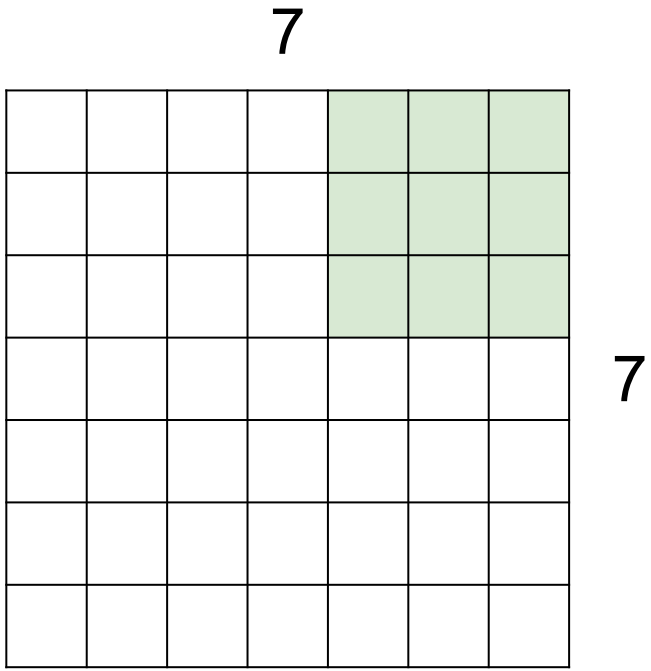


7x7 вход,

используем

фильтр 3x3

Свёрточный слой. Подробный взгляд



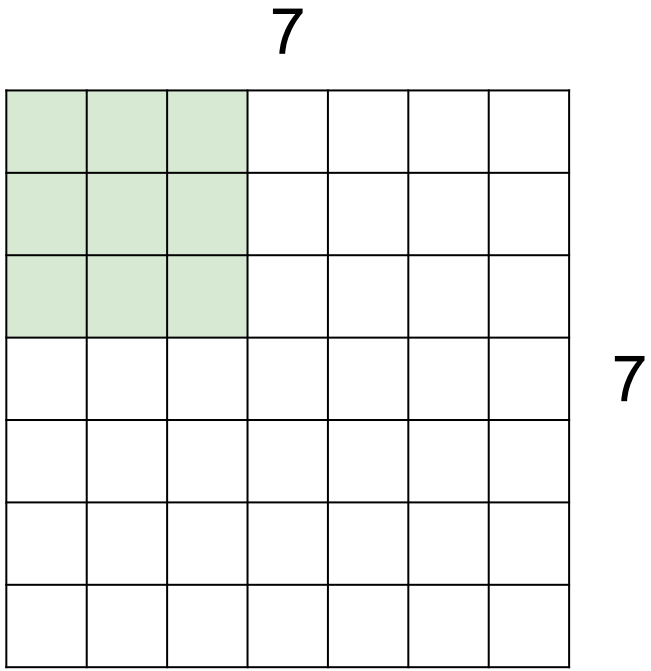
7x7 вход,

используем

фильтр 3x3

=> 5x5 выход

Свёрточный слой. Подробный взгляд

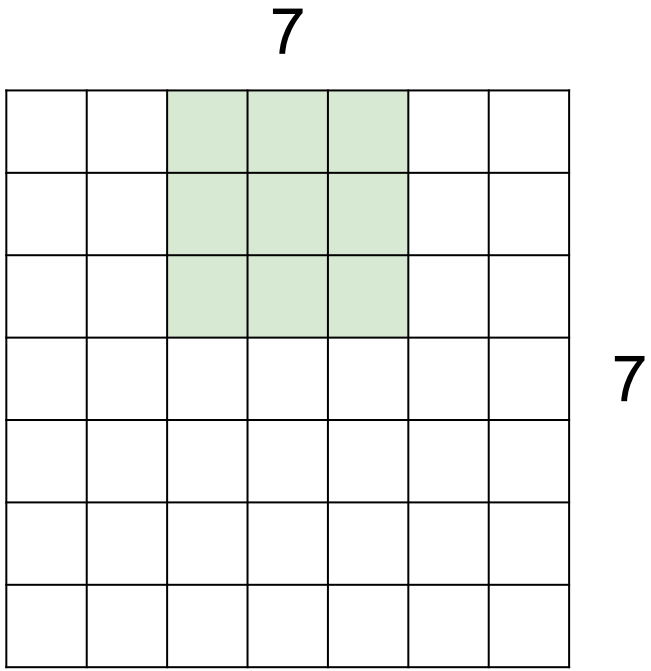


7x7 вход,

используем фильтр 3x3

с шагом (stride) 2

Свёрточный слой. Подробный взгляд

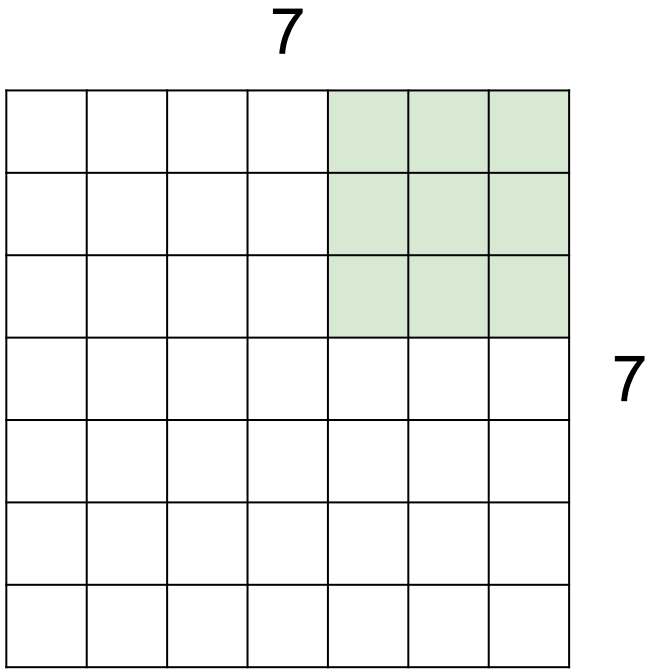


7x7 вход,

используем фильтр 3x3

с шагом (stride) 2

Свёрточный слой. Подробный взгляд



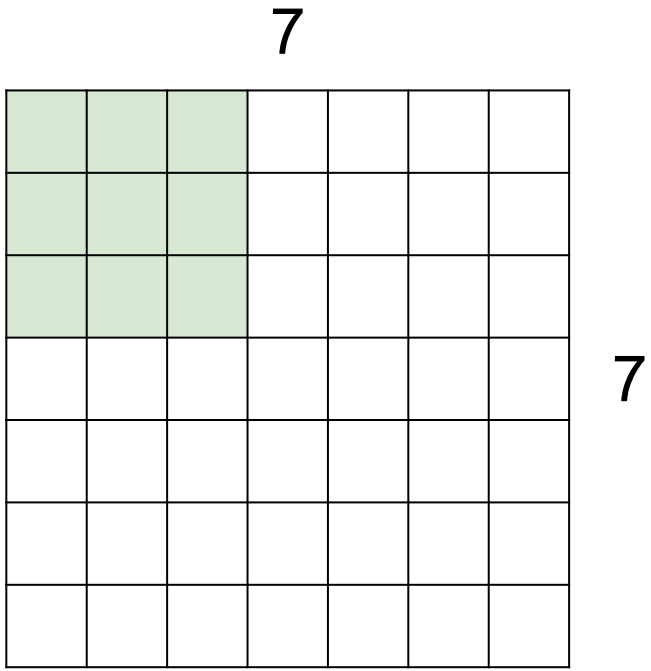
7x7 вход,

используем фильтр 3x3

с шагом (stride) 2

=> 3x3 выход!

Свёрточный слой. Подробный взгляд

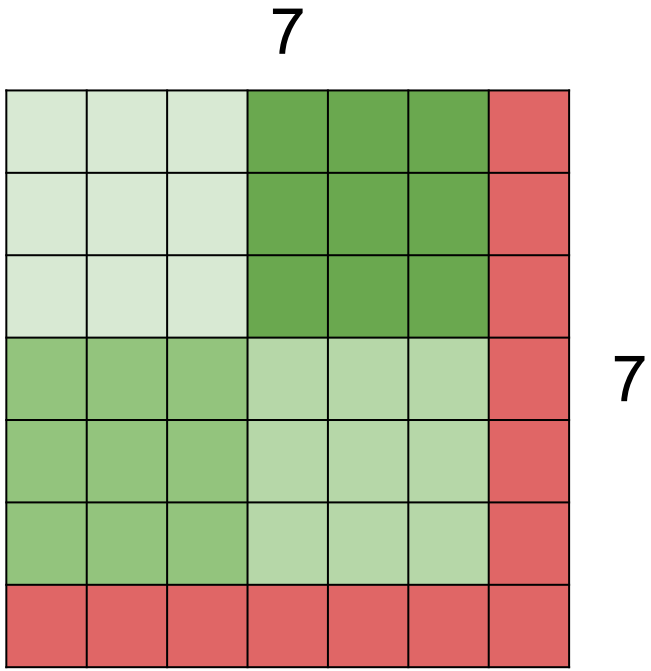


7x7 вход,

используем фильтр 3x3

с шагом (stride) 3?

Свёрточный слой. Подробный взгляд



7x7 вход,

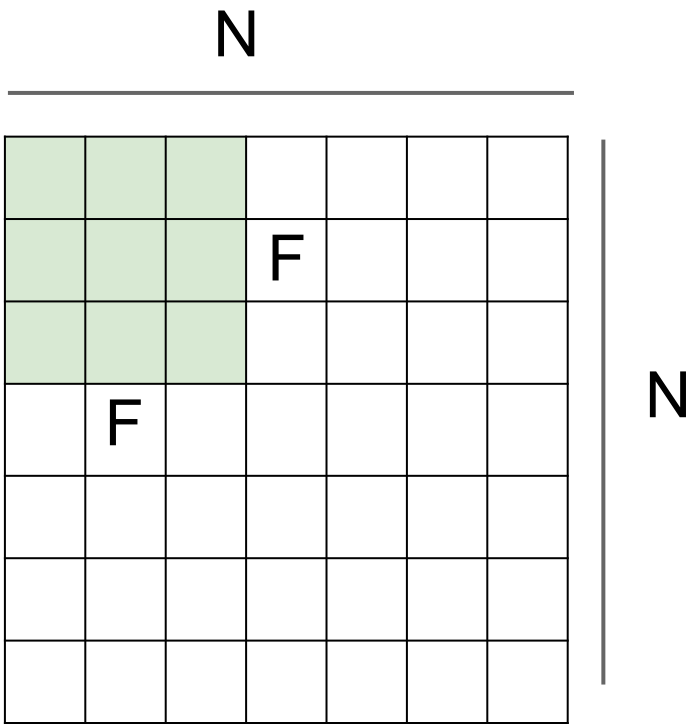
используем фильтр 3x3

с шагом (stride) 3?

Не подойдёт,

так как останется область, к которой свёртка не применяется

Свёрточный слой. Подробный взгляд



Выходной размер:

(N - F) / stride + 1

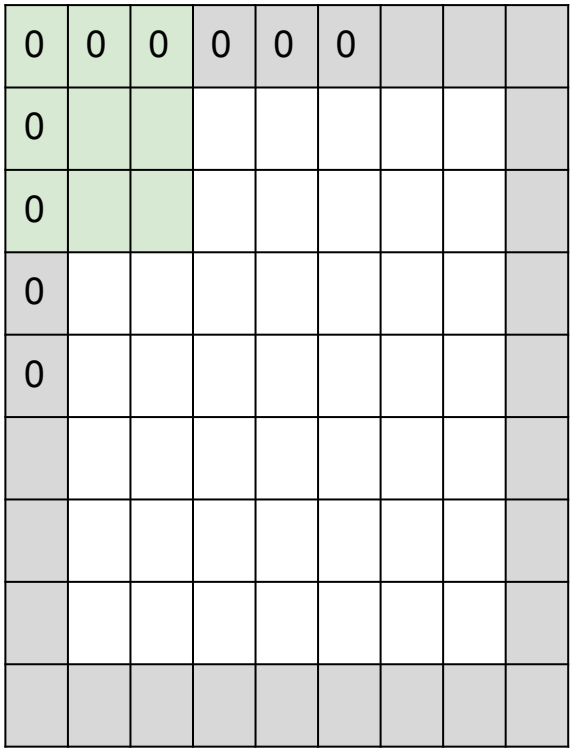
N = 7, F = 3:

stride 1 => (7 - 3)/1 + 1 = 5

stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = 2.33 :\

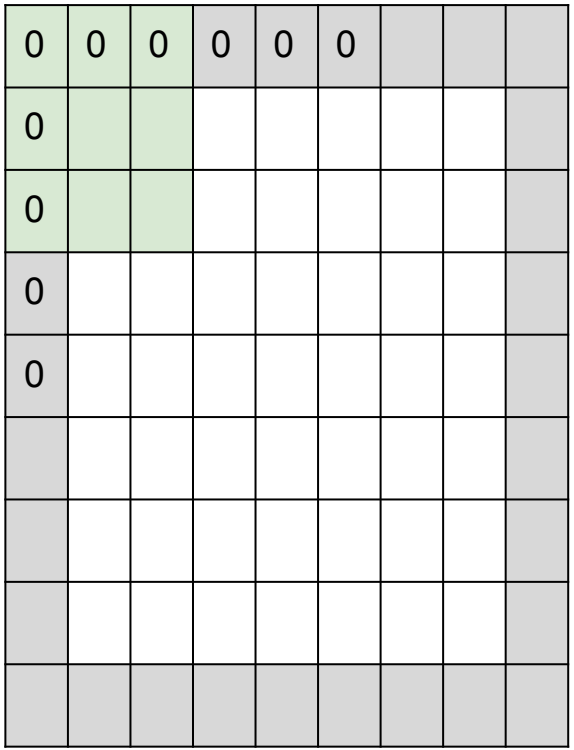
Свёрточный слой. Дополнение границ



7x7 вход
3x3 фильтр, **шаг (stride) 1**
дополняем границы 1 пикселем =>
Какой будет выходной размер?

(N - F) / stride + 1

Свёрточный слой. Дополнение границ

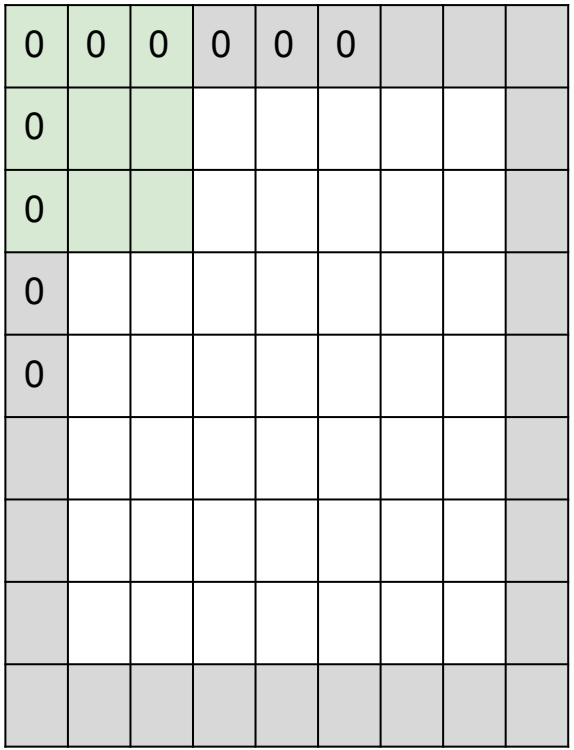


7x7 вход
3x3 фильтр, **шаг (stride) 1**
дополняем границы 1 пикселем =>
Какой будет выходной размер?

7x7 выход

(N + 2P - F) / stride + 1

Свёрточный слой. Дополнение границ



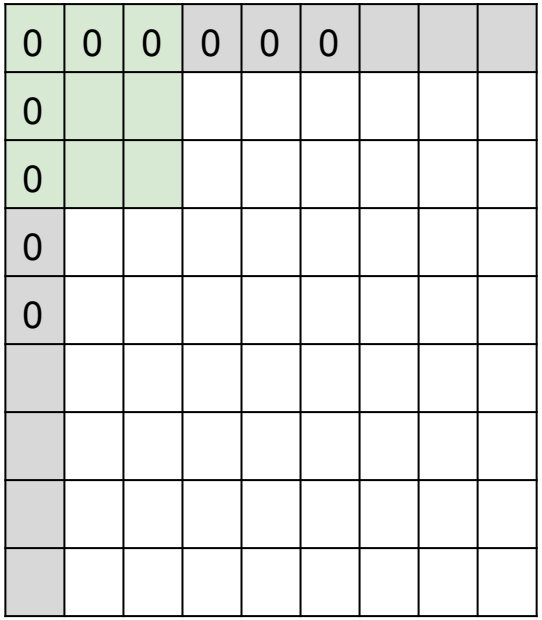
На практике для свёрточных слоёв с шагом 1 и фильтром размера FxF используют дополнение нулями (zero-padding) (F-1)/2. (Это сохраняет пространственный размер)

F = 3 => zero pad with 1

F = 5 => zero pad with 2

F = 7 => zero pad with 3

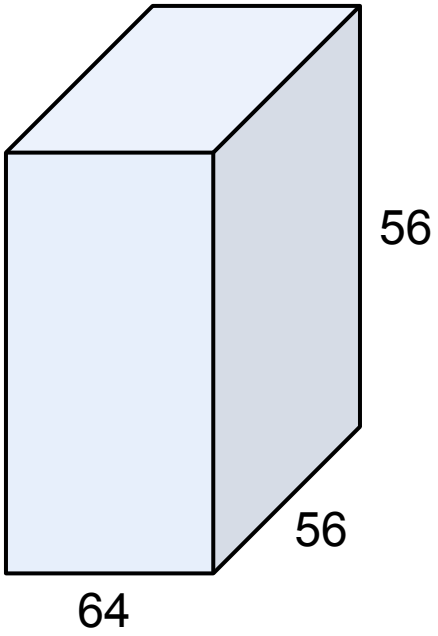
Свёрточный слой. Уменьшение размерности



Практически всегда бывает необходимо уменьшить пространственную размерность изображения. Для этого к изображению **размера NxN** применяют zero-padding **слева** и **сверху** и затем сворачивают с **фильтром 3x3** с **шагом 2**, получая изображение размера N/2xN/2.

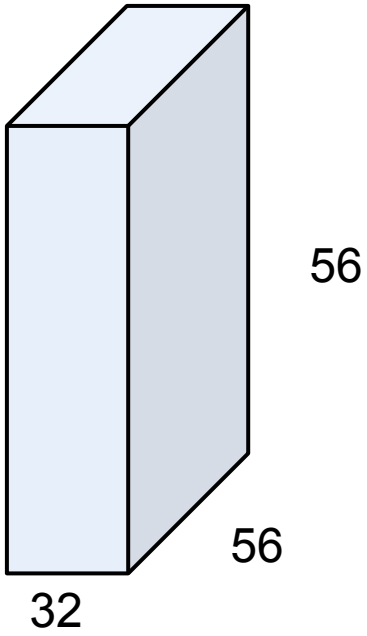
$$(N + 1 - 3) / 2 + 1 = N/2$$

Свёрточный слой 1x1

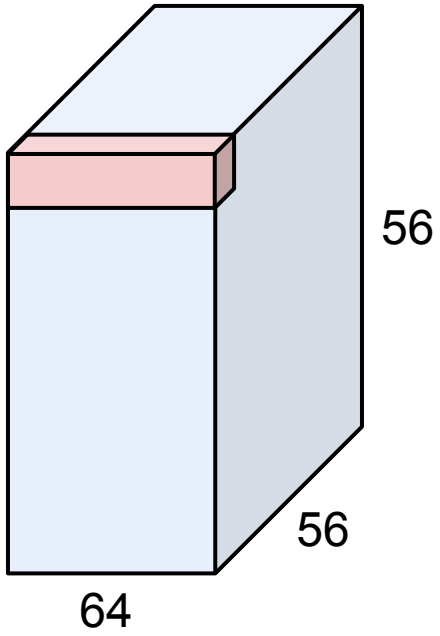


1x1 CONV
с 32 фильтрами

(каждый слой размера 1x1x64 выполняет 64-мерное скалярное произведение)

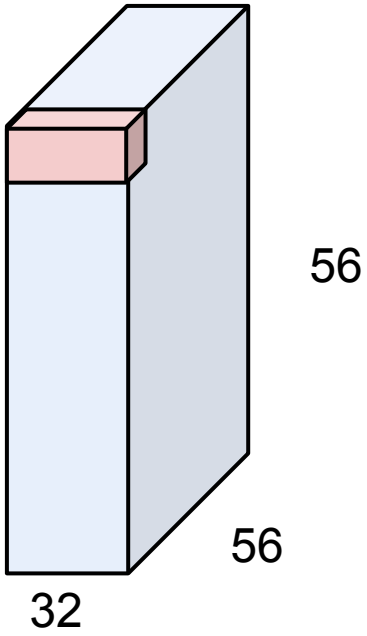


Свёрточный слой 1x1



1x1 CONV
с 32 фильтрами

(каждый слой размера 1x1x64 выполняет 64-мерное скалярное произведение)



Свёрточный слой в PyTorch

Необходимы 4 гиперпараметра:

- Number of filters **K**
- The filter size **F**
- The stride **S**
- The zero padding **P**

Conv2d

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of implicit zero-paddings on both sides for `padding` number of points for each dimension.
- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this link has a nice visualization of what `dilation` does.
- `groups` controls the connections between inputs and outputs. `in_channels` and `out_channels` must both be divisible by `groups`. For example,
  - At groups=1, all inputs are convolved to all outputs.
  - At groups=2, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels, and producing half the output channels, and both subsequently concatenated.
  - At groups= `in_channels`, each input channel is convolved with its own set of filters, of size: $\left\lfloor \frac{C_{\text{out}}}{C_{\text{in}}} \right\rfloor$.

The parameters `kernel_size`, `stride`, `padding`, `dilation` can either be:

- a single `int` – in which case the same value is used for the height and width dimension
- a `tuple` of two ints – in which case, the first *int* is used for the height dimension, and the second *int* for the width dimension

Свёрточный слой в Keras

Conv layer needs 4 hyperparameters:

- Number of filters **K**
- The filter size **F**
- The stride **S**
- The zero padding **P**

Conv2D [source]

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=None, d
```

2D convolution layer (e.g. spatial convolution over images).

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If `use_bias` is True, a bias vector is created and added to the outputs. Finally, if `activation` is not `None`, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument `input_shape` (tuple of integers, does not include the batch axis), e.g. `input_shape=(128, 128, 3)` for 128x128 RGB pictures in `data_format="channels_last"`.

Arguments

- **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
- **kernel_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
- **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.
- **padding**: one of `"valid"` or `"same"` (case-insensitive). Note that `"same"` is slightly inconsistent across backends with `strides` != 1, as described here
- **data_format**: A string, one of `"channels_last"` or `"channels_first"`. The ordering of the dimensions in the inputs. `"channels_last"` corresponds to inputs with shape `(batch, height, width, channels)` while `"channels_first"` corresponds to inputs with shape `(batch, channels, height, width)`. It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".

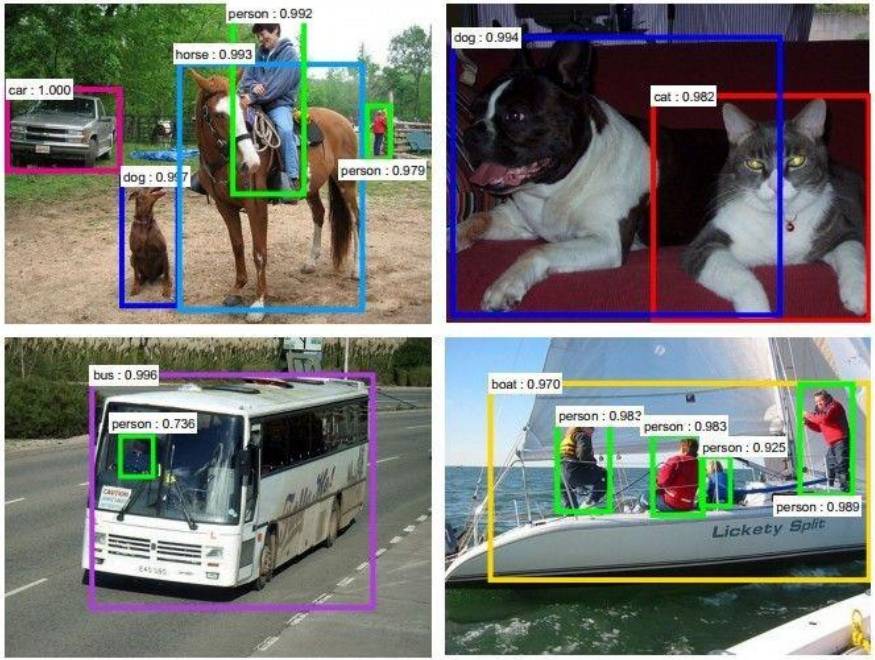
Свёрточные сети повсюду)

Классификация

Поиск похожих изображений

Свёрточные сети повсюду)

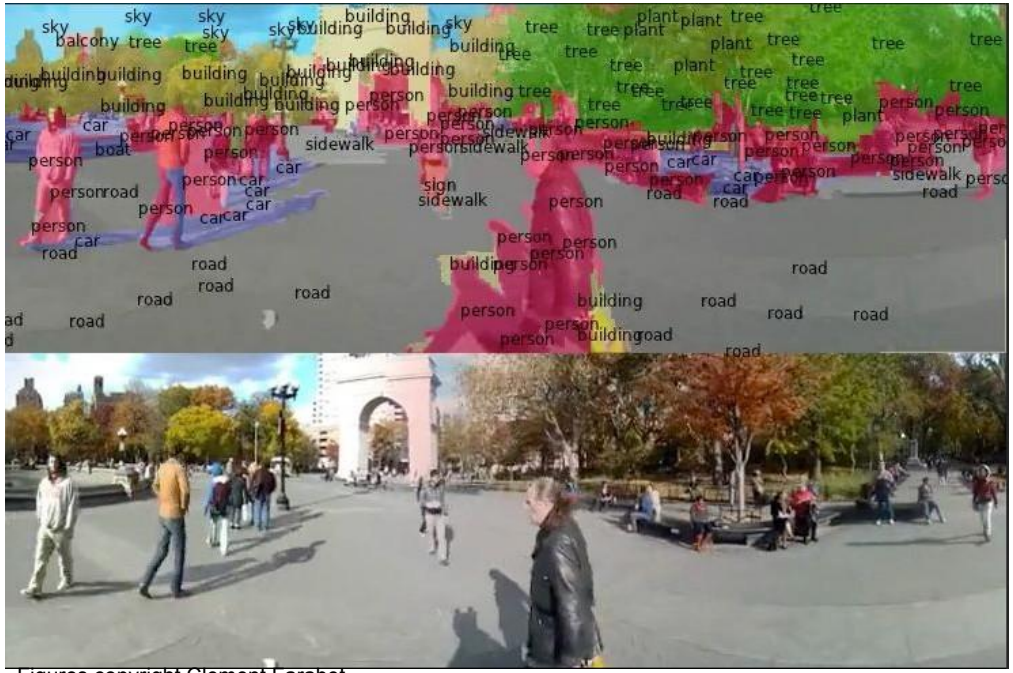
Детектирование



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

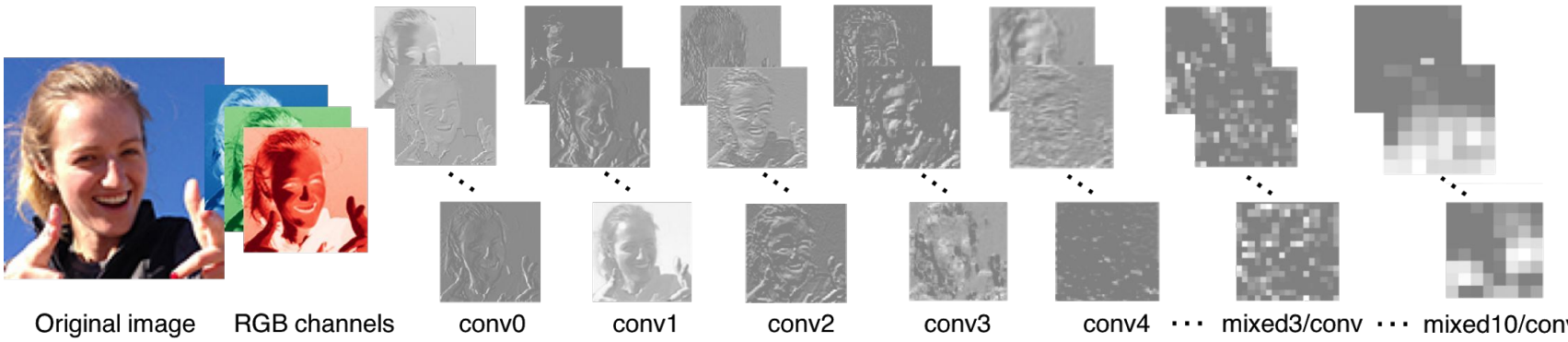
Сегментация



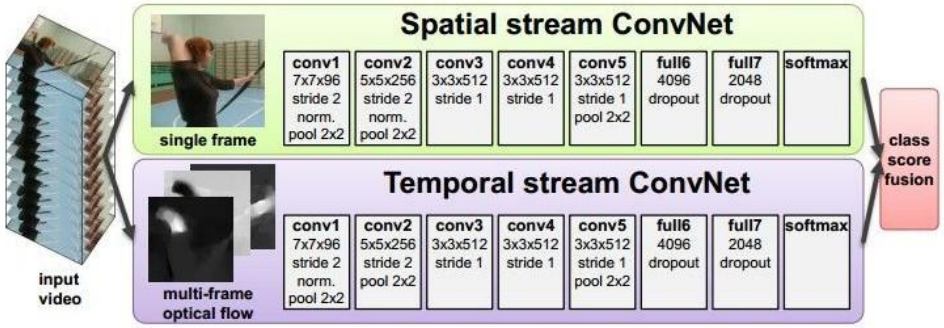
Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Свёрточные сети повсюду)



[Taigman et al. 2014]



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.

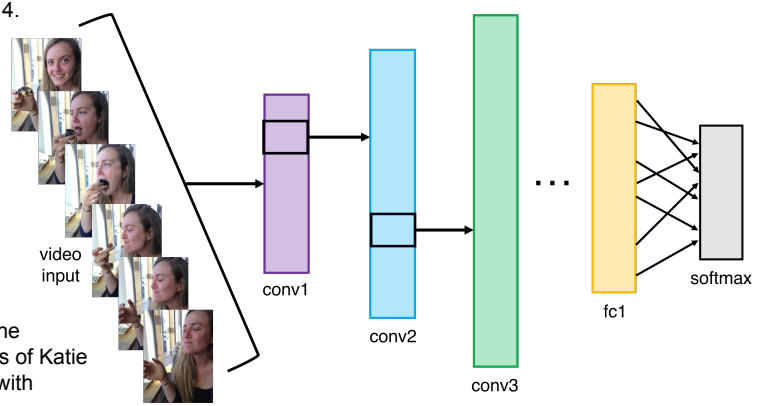
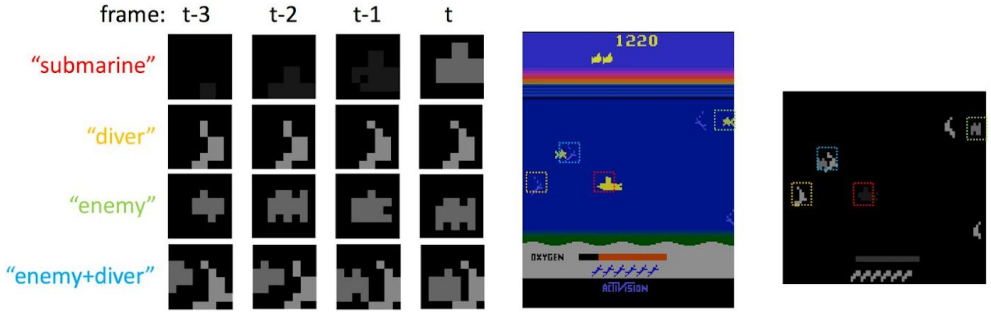


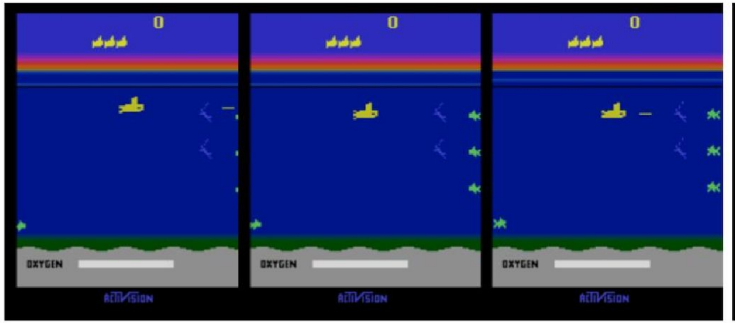
Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

Свёрточные сети повсюду)

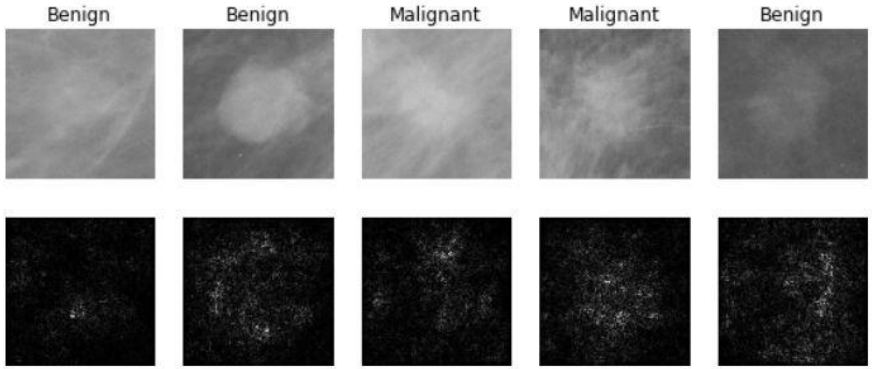
[Toshev, Szegedy 2014]



[Guo et al. 2014]



Свёрточные сети повсюду)



[Levy et al. 2016]

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011] [Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

Свёрточные сети повсюду)

Whale recognition, Kaggle Challenge

Mnih and Hinton, 2010

Свёрточные сети повсюду)

Без ошибок

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

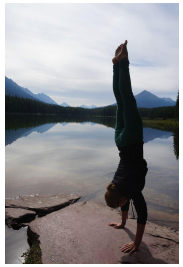
Небольшие ошибки

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Немного похоже

A woman is holding a cat in her hand

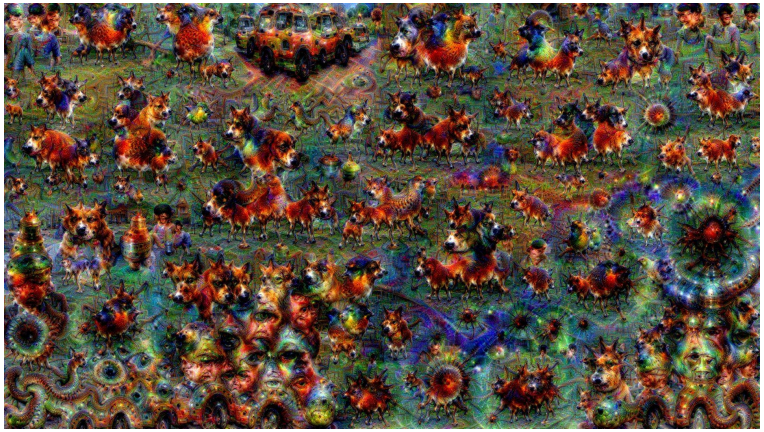


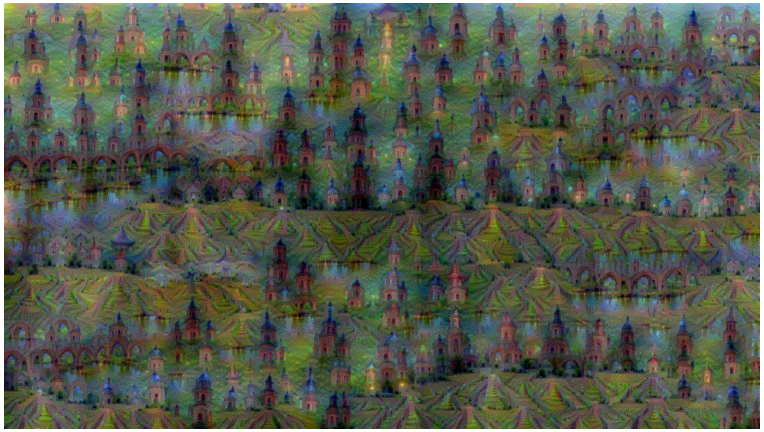
A woman standing on a beach holding a surfboard

Описание Изображений

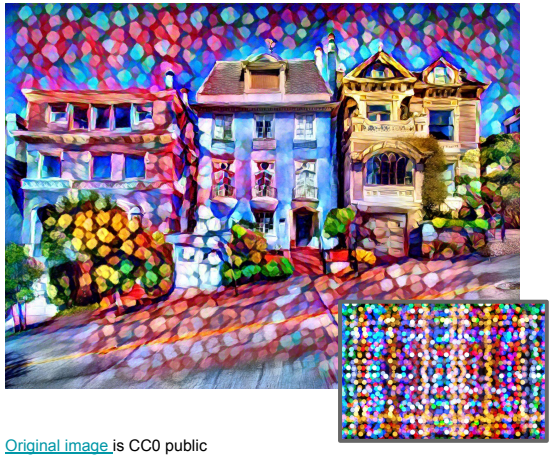
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Captions generated by Justin Johnson using Neuraltalk2





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



Original image is CC0 public domain

Starry Night and Tree Roots by Van Gogh are in the public domain Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017; reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016 Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

[ConvNetJS demo: training on CIFAR-10]

ConvNetJS CIFAR-10 demo

Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.

http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html