Computer Vision

第15-2章 深度学习中的图像特征

计算机科学与技术学院

華東師範大學
EAST CHINA NORMAL UNIVERSITY

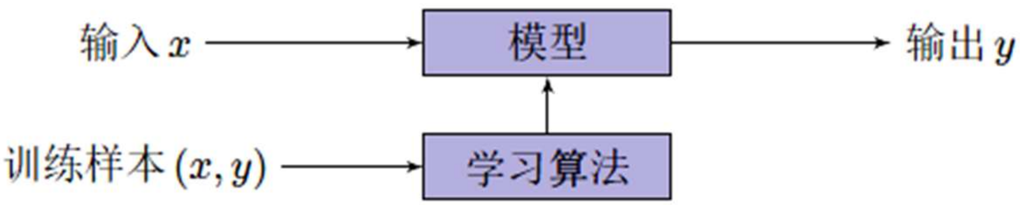
本次课程内容

1. 卷积神经网络知识回顾
2. 典型的网络结构
3. 卷积神经网络中特征
4. VGG Net

1. 卷积神经网络知识回顾

- 知识回顾
- 机器学习概念

$$\hat{y} = f(\phi(x), \theta), \quad \text{或者} \quad \hat{y} = f(x, \theta).$$



机器学习系统示例

3

发展历史

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

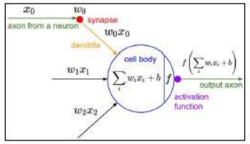
The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized letters of the alphabet

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

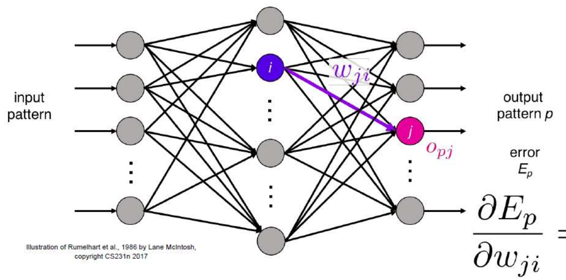
update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$



Frank Rosenblatt, ~1957: Perceptron

Widrow and Hoff, ~1960: Adaline/Madaline



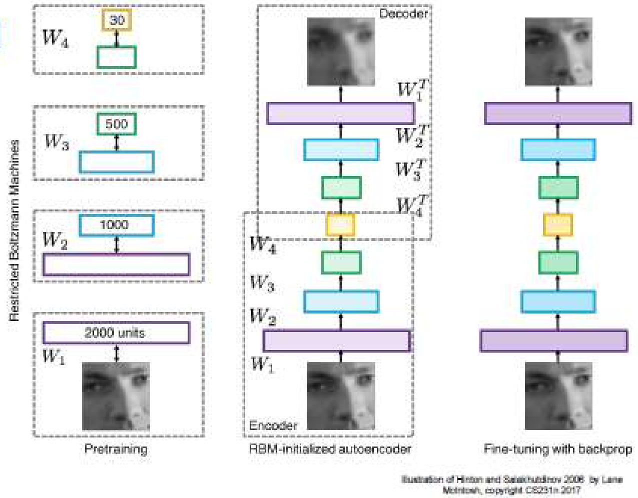
recognizable math

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$

Rumelhart et al., 1986: First time back-propagation became popular

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning



ImageNet Classification with Deep Convolutional Neural Networks
[Krizhevsky, Sutskever, Hinton, 2012]

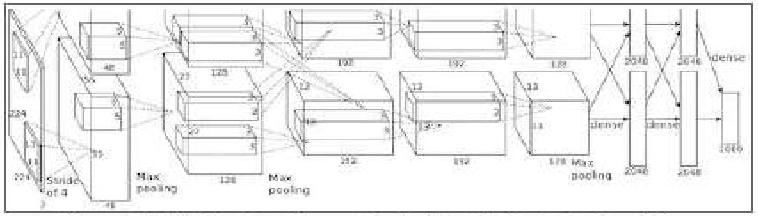
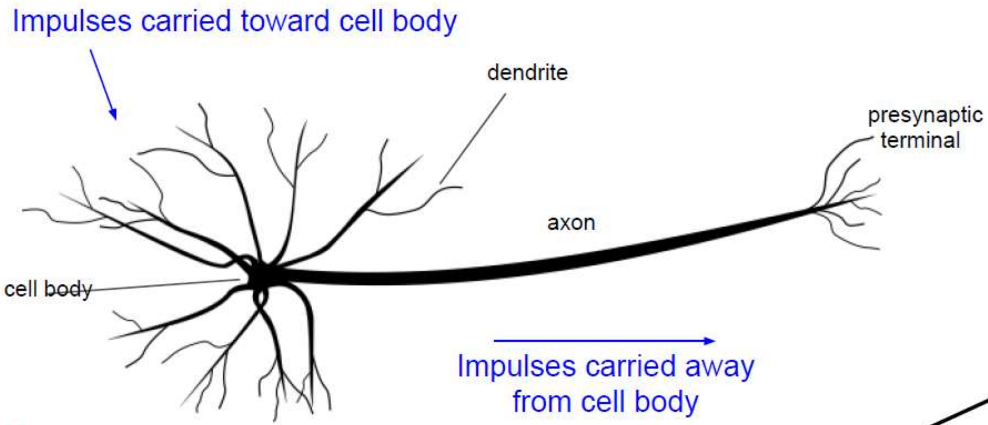


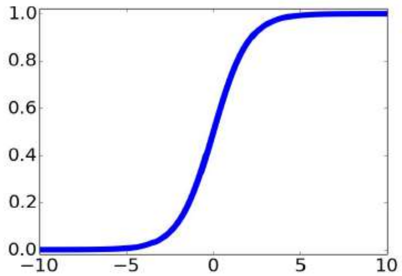
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

http://cs231n.stanford.edu/slides/2017

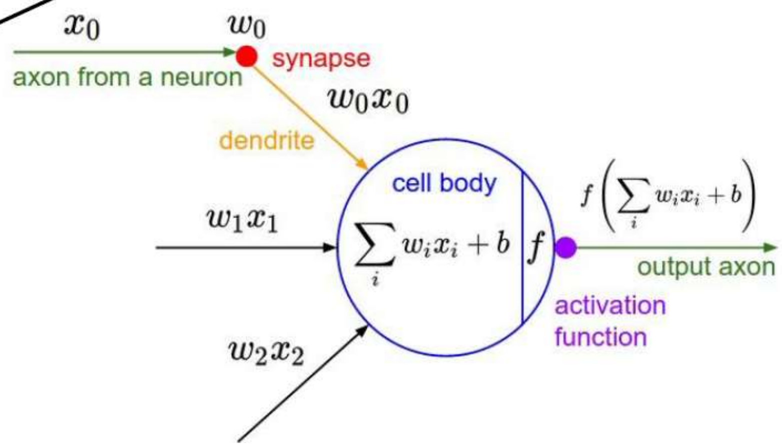



```python
class Neuron:
  # ...
  def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation func
    return firing_rate
```



sigmoid activation function

$$\frac{1}{1+e^{-x}}$$

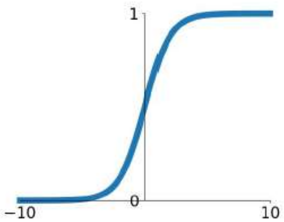


http://cs231n.stanford.edu/slides/2017

激活函数Activation functions

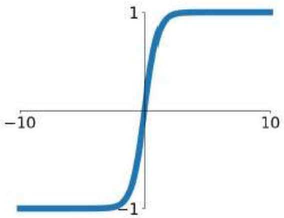
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



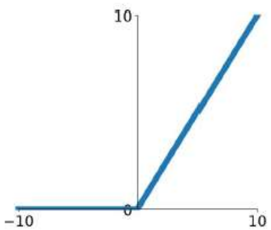
tanh

$\tanh(x)$



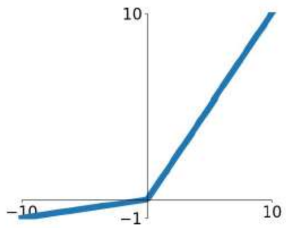
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

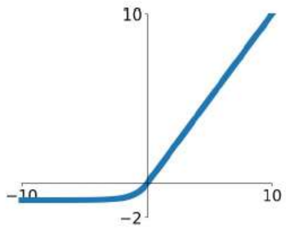


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



http://cs231n.stanford.edu/slides/2017

软件架构

Caffe
(UC Berkeley)
→
Caffe2
(Facebook)

Paddle
(Baidu)

Torch
(NYU / Facebook)
→
PyTorch
(Facebook)

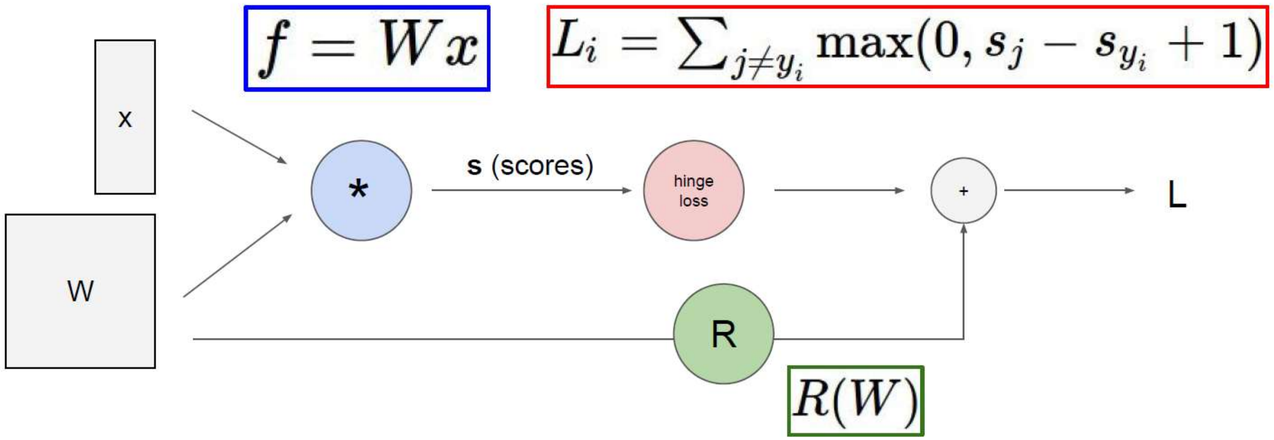
CNTK
(Microsoft)

Theano
(U Montreal)
→
TensorFlow
(Google)

MXNet
(Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

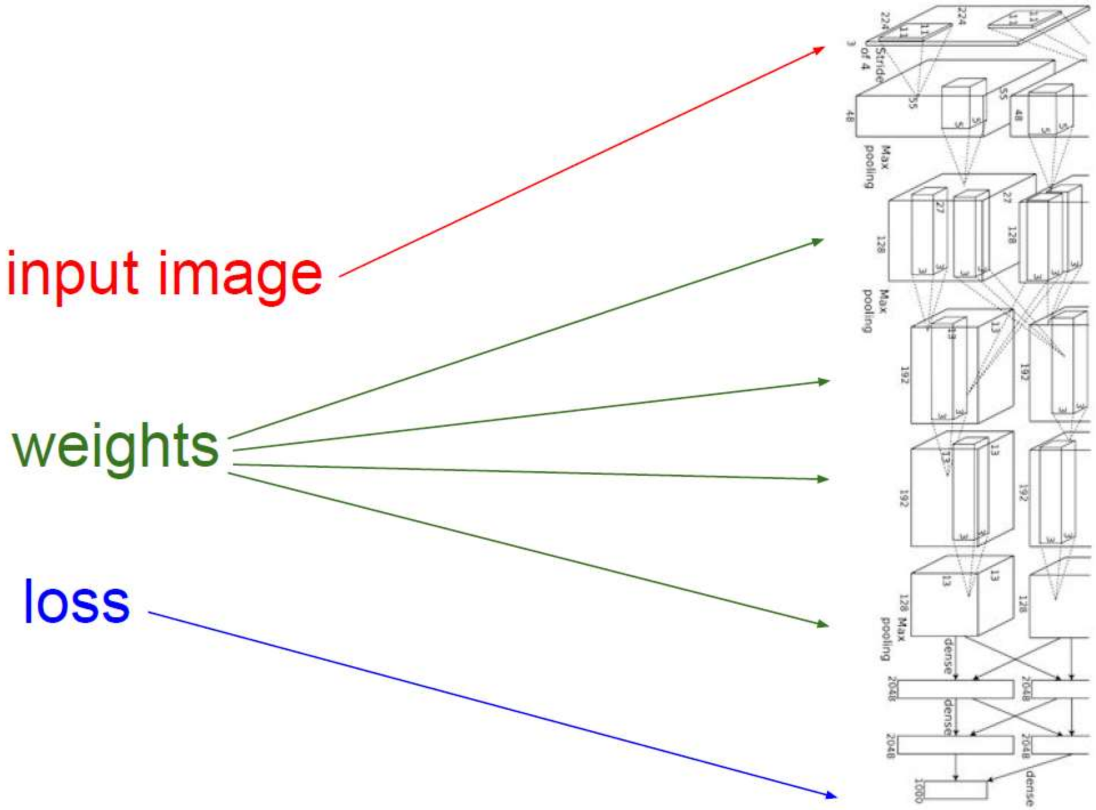
http://cs231n.stanford.edu/slides/2017

计算图



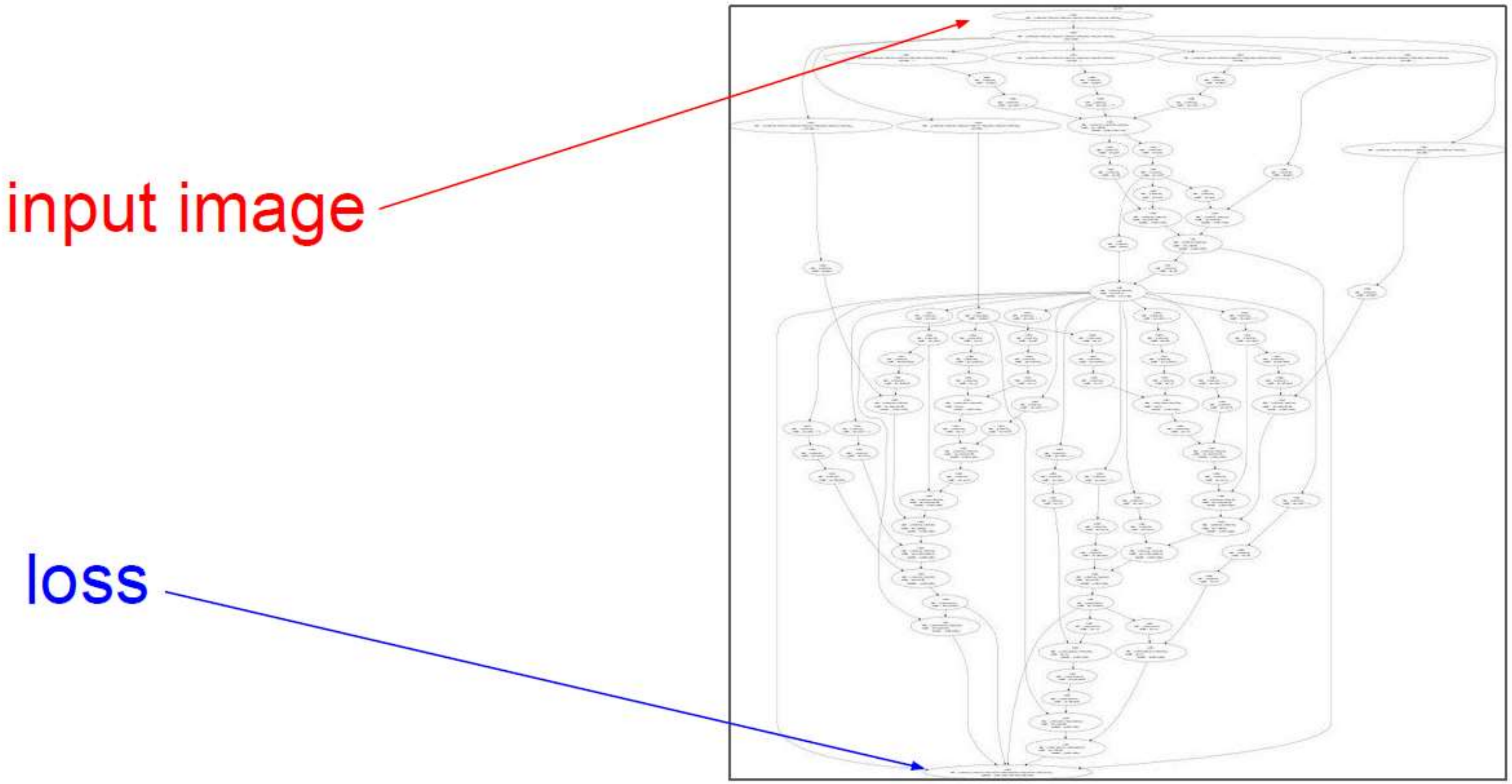
http://cs231n.stanford.edu/slides/2017

计算图



http://cs231n.stanford.edu/slides/2017

计算图



http://cs231n.stanford.edu/slides/2017

计算图代码

Numpy

```python
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

TensorFlow

```python
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

PyTorch

```python
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```


http://cs231n.stanford.edu/slides/2017

Tensorflow 代码

TensorFlow:

```python
import numpy as np
import tensorflow as tf
```

(Assume imports at the top of each snipppet)

First **define** computational graph

```python
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])
```

Then **run** the graph many times

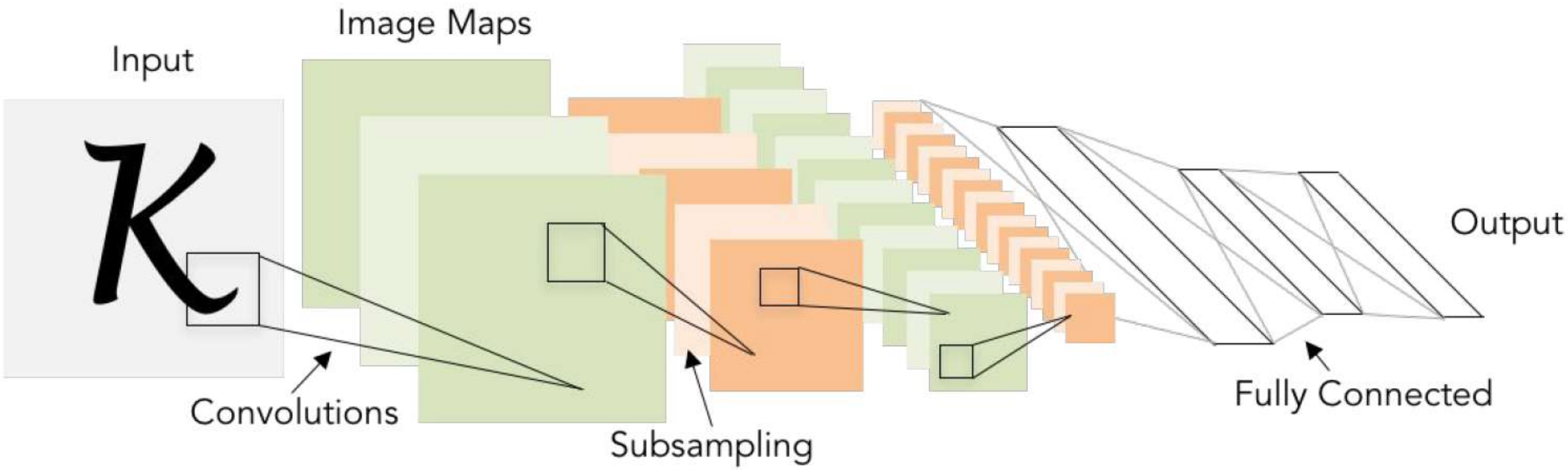
```python
with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

http://cs231n.stanford.edu/slides/2017

2. 典型的网络结构

LeNet-5

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

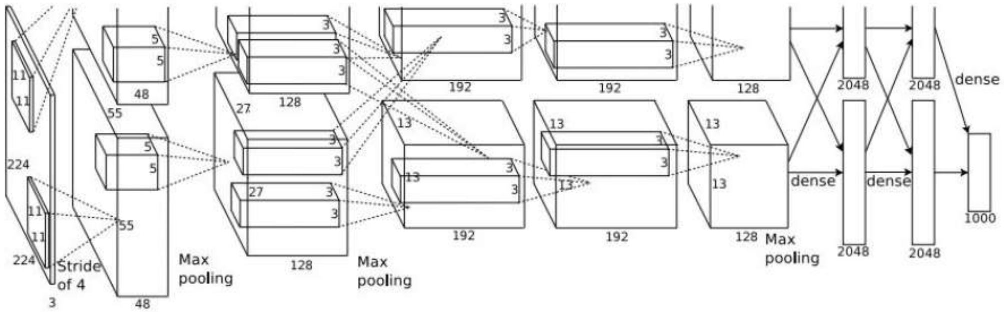
http://cs231n.stanford.edu/slides/2017

典型的网络结构

Case Study: AlexNet

[Krizhevsky et al. 2012]

Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8



http://cs231n.stanford.edu/slides/2017

典型的网络结构

Case Study: VGGNet

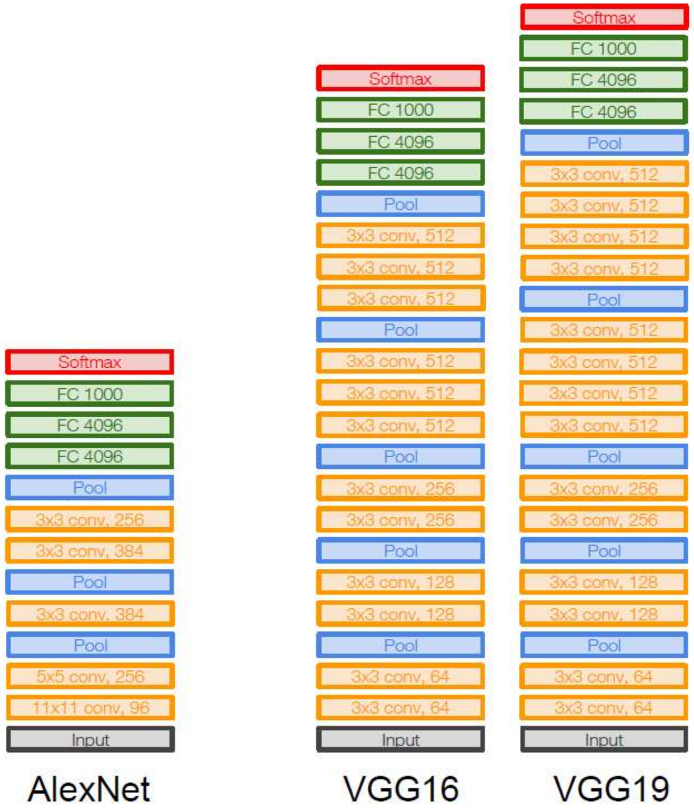
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



http://cs231n.stanford.edu/slides/2017

神经网络实例 NEURAL NETWORKS

◆ 简单的神经元结构

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

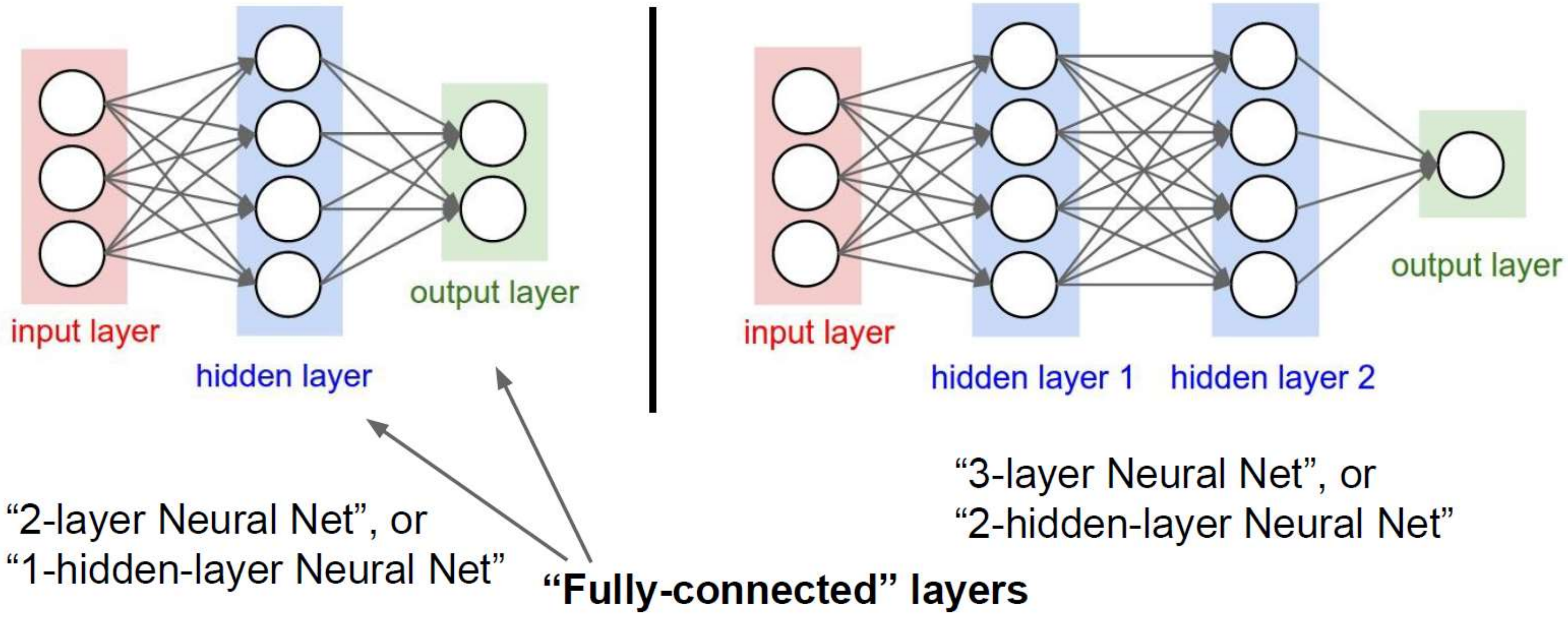
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

http://cs231n.stanford.edu/slides/2017

神经网络实例---全连接结构

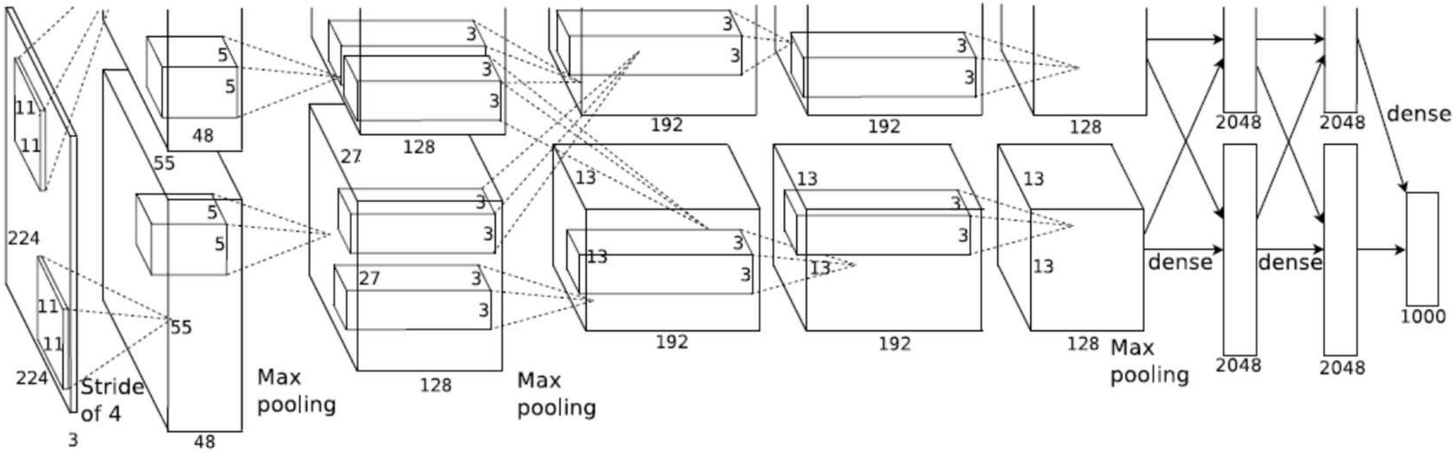
◆ 全连接的神经元结构



http://cs231n.stanford.edu/slides/2017

神经网络实例---卷积神经网络

◆卷积神经网络：有卷积也有全连接层

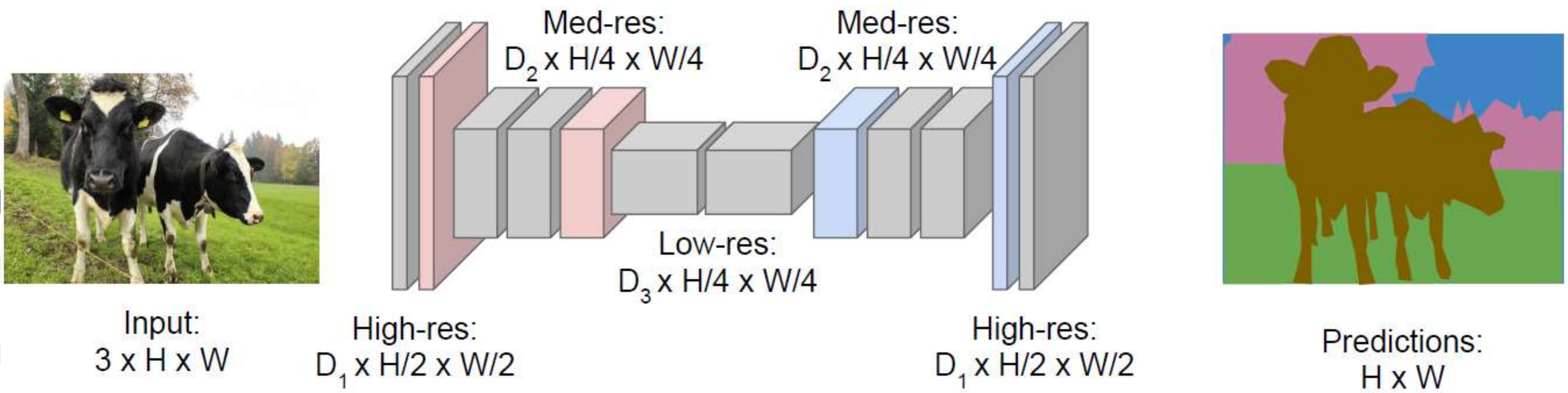


http://cs231n.stanford.edu/slides/2017

18

神经网络实例---全卷积神经网络

◆全卷积神经网络：只有卷积和反卷积，没有全连接



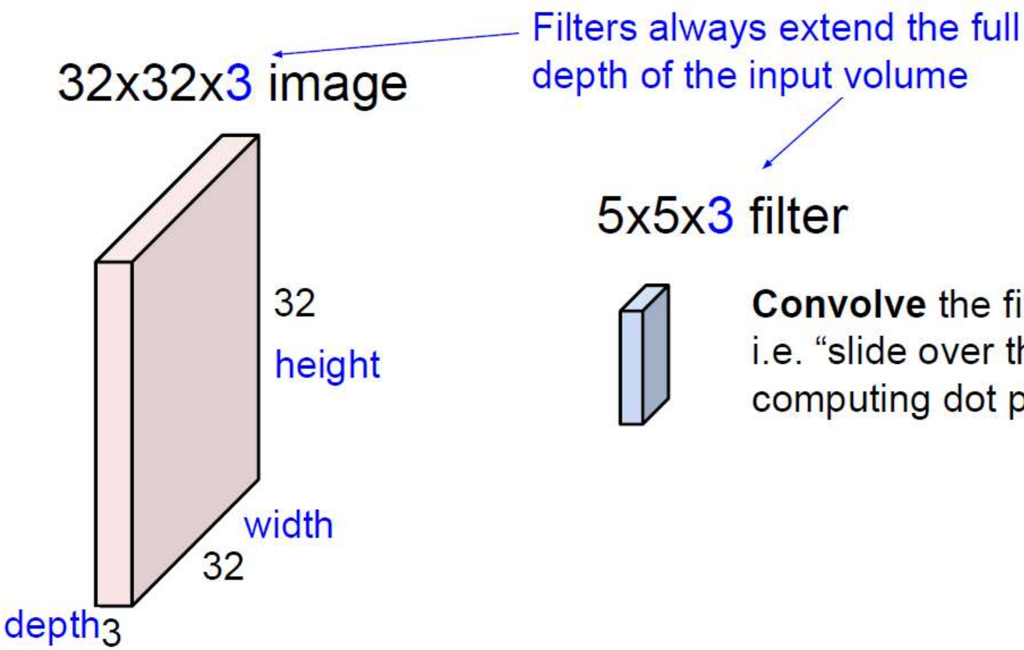
http://cs231n.stanford.edu/slides/2017

3. 卷积神经网络中特征

- 卷积层
- 全连接层

20

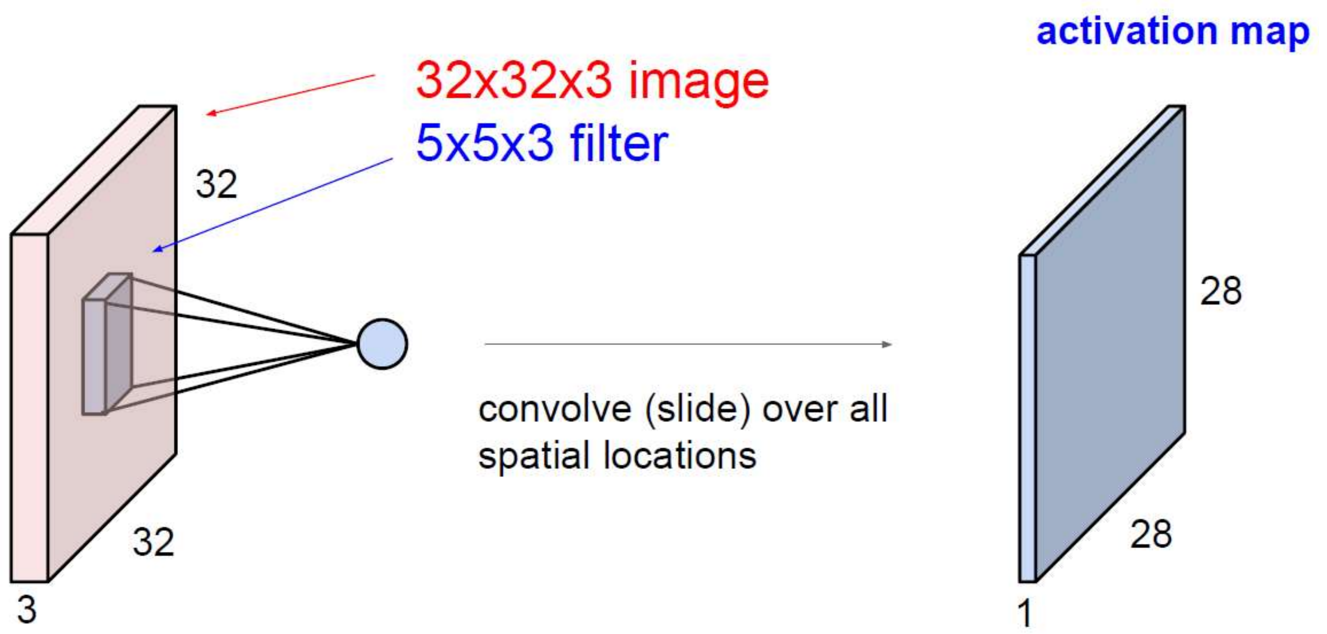
卷积层



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

http://cs231n.stanford.edu/slides/2017

卷积层

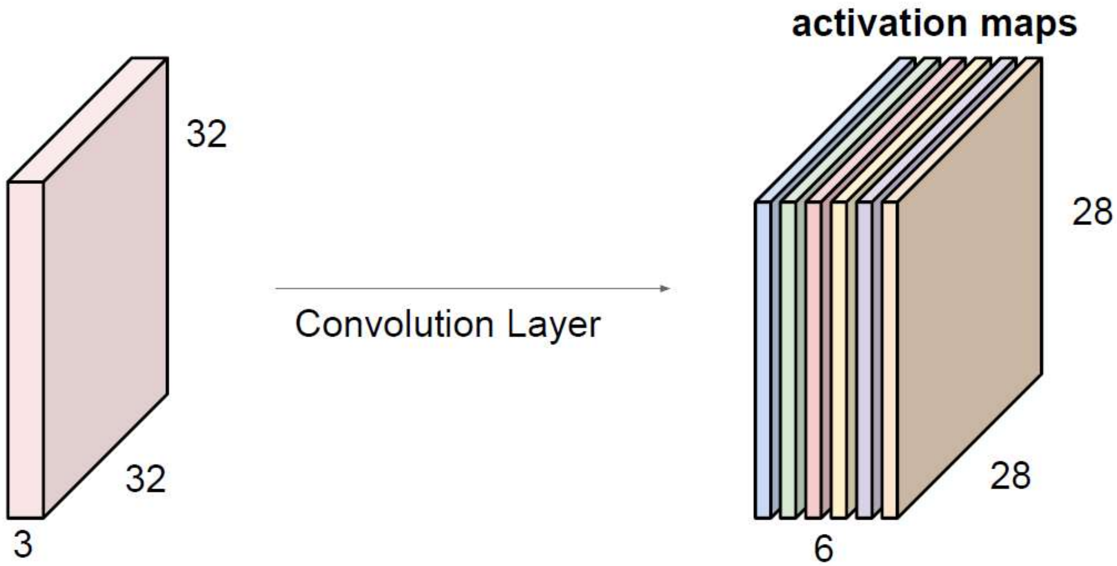


http://cs231n.stanford.edu/slides/2017

22

卷积层

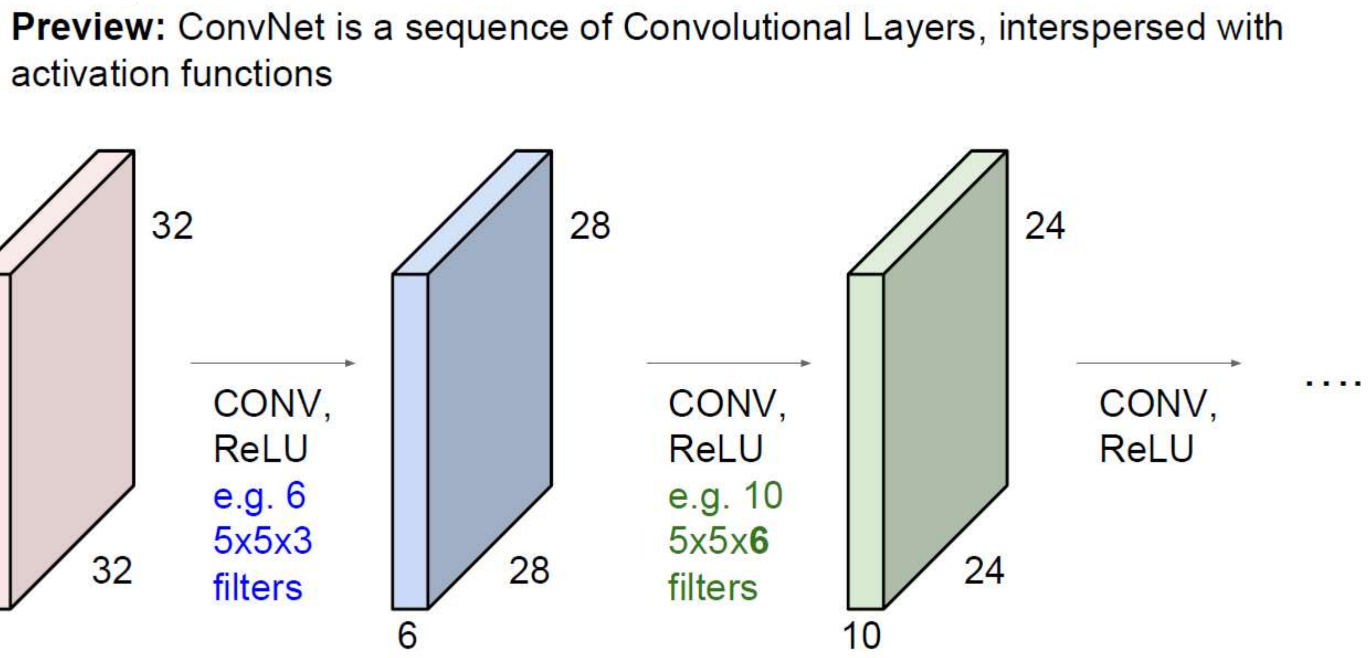
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

http://cs231n.stanford.edu/slides/2017

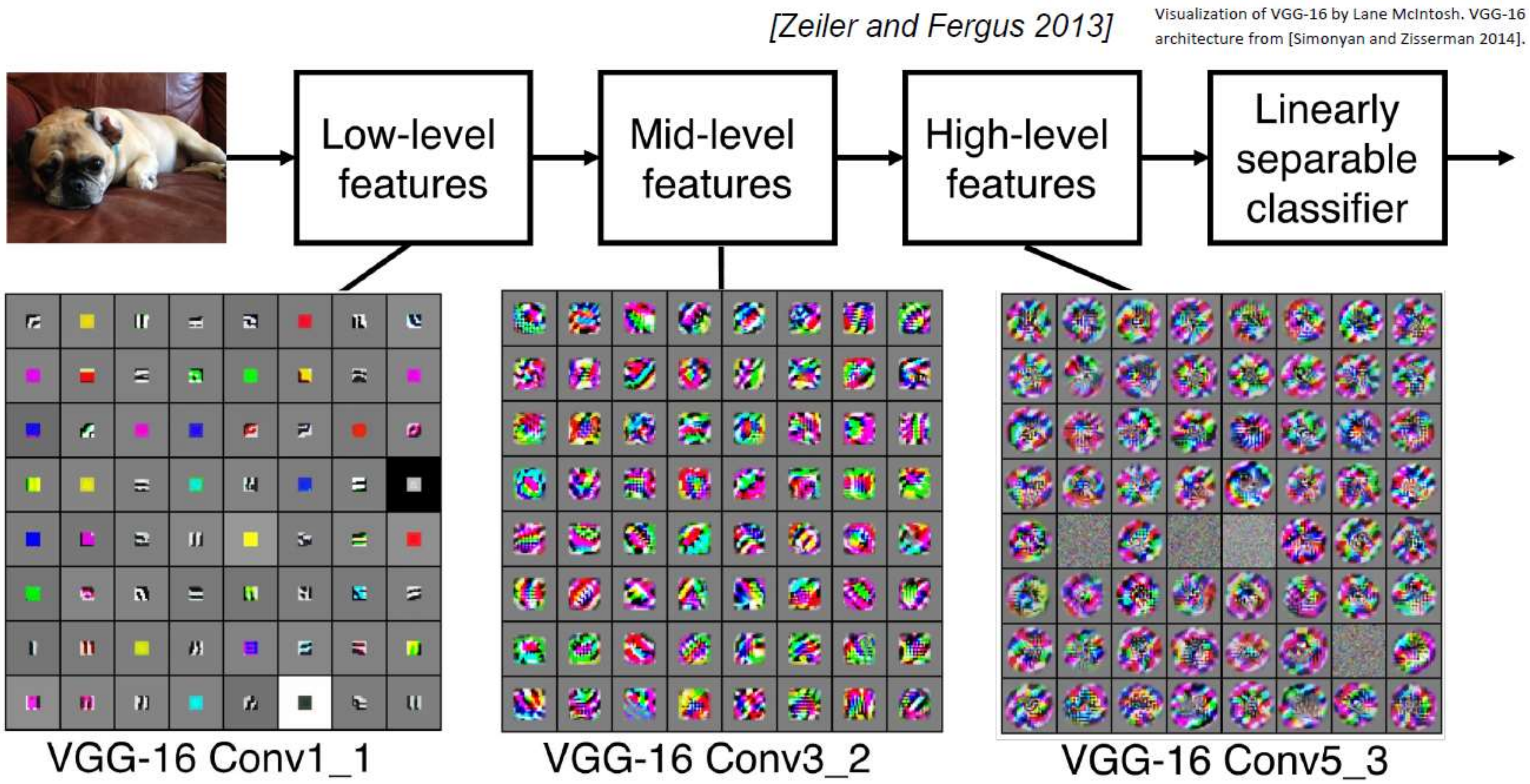
卷积层



http://cs231n.stanford.edu/slides/2017

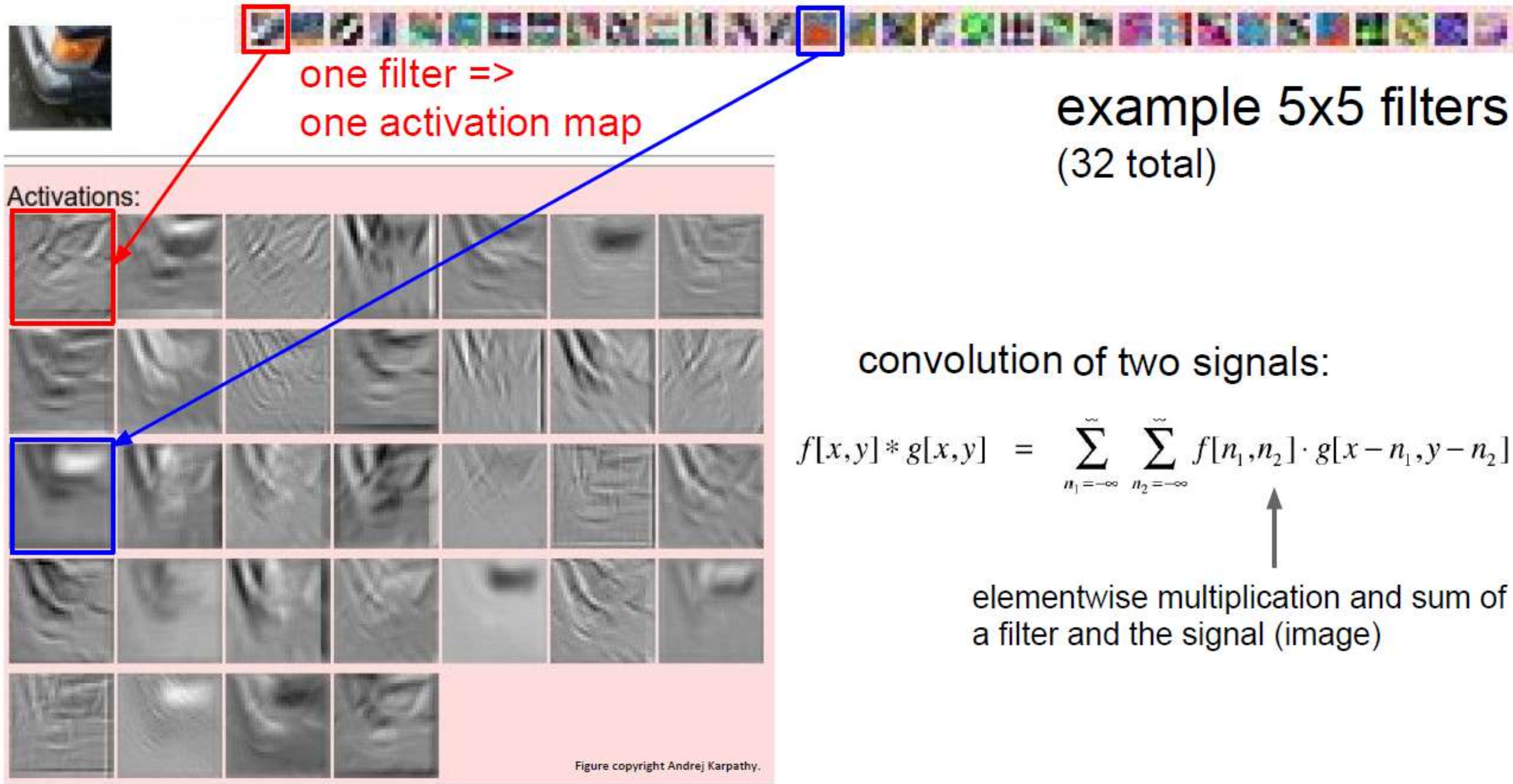
24

卷积层



http://cs231n.stanford.edu/slides/2017

卷积层



example 5x5 filters
(32 total)

convolution of two signals:

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

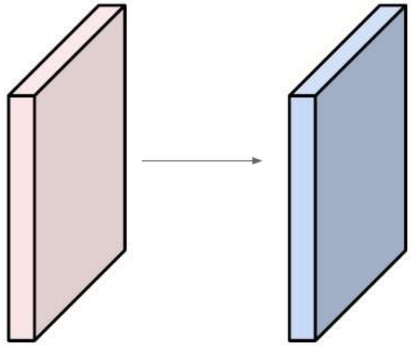
elementwise multiplication and sum of a filter and the signal (image)

http://cs231n.stanford.edu/slides/2017

参数个数

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params

(+1 for bias)

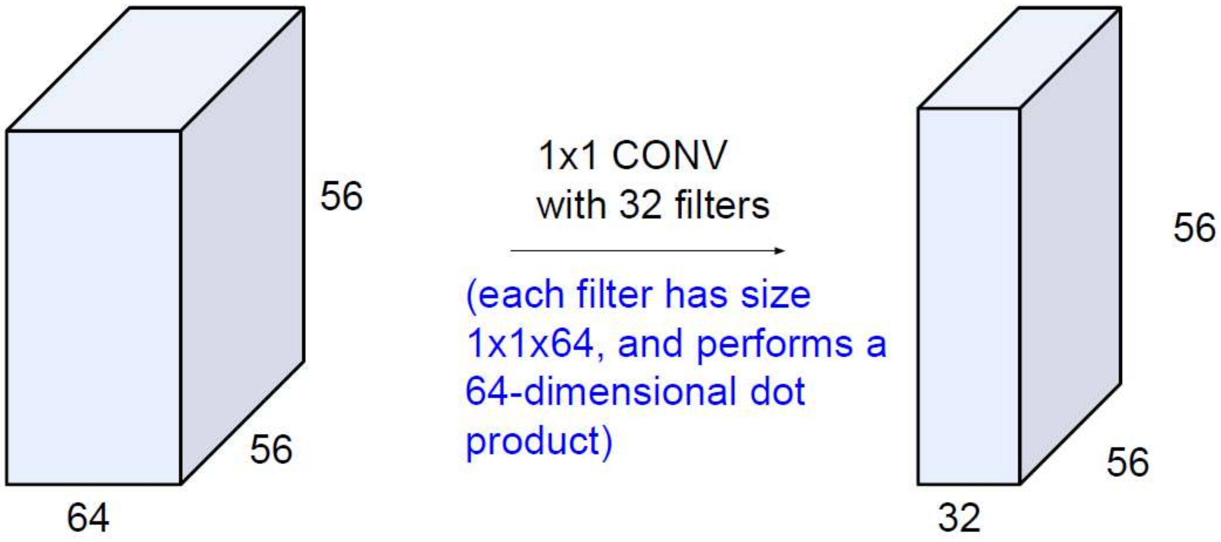
=> 76*10 = **760**

27

http://cs231n.stanford.edu/slides/2017

参数个数

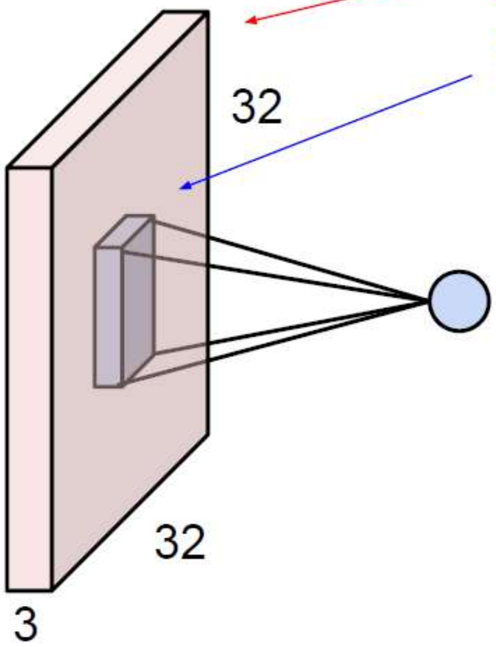
1x1 convolution layers make perfect sense



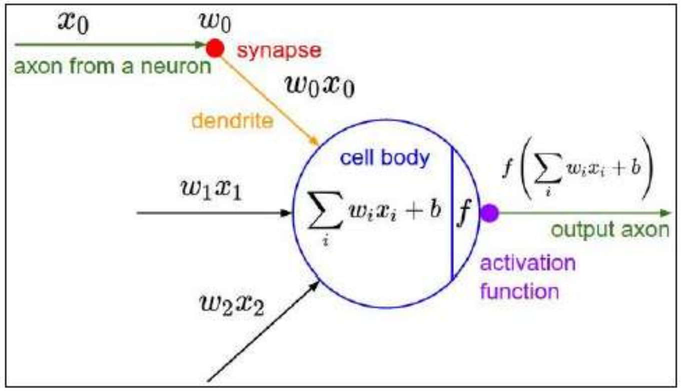
http://cs231n.stanford.edu/slides/2017

28

The brain/neuron view of CONV Layer



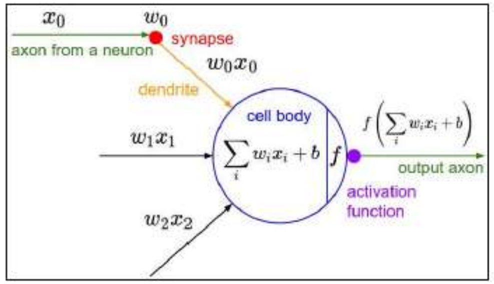
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

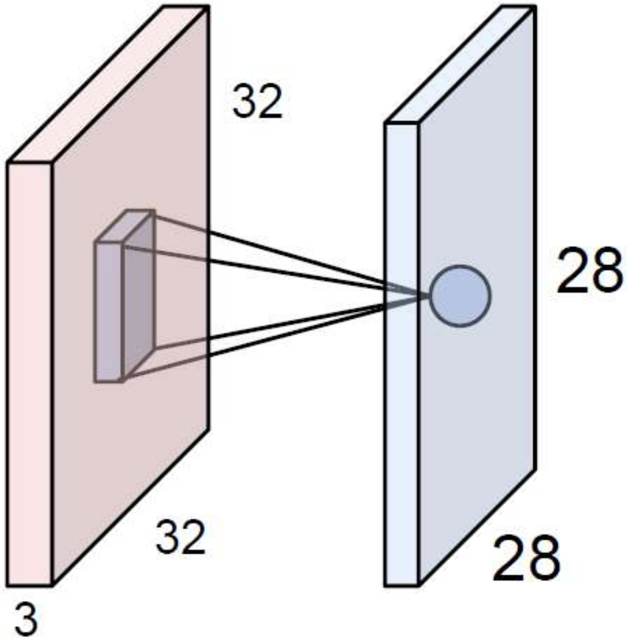


It's just a neuron with local connectivity...

http://cs231n.stanford.edu/slides/2017

The brain/neuron view of CONV Layer



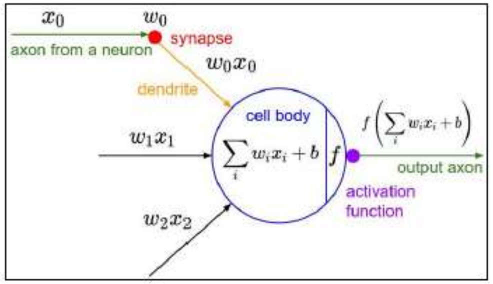


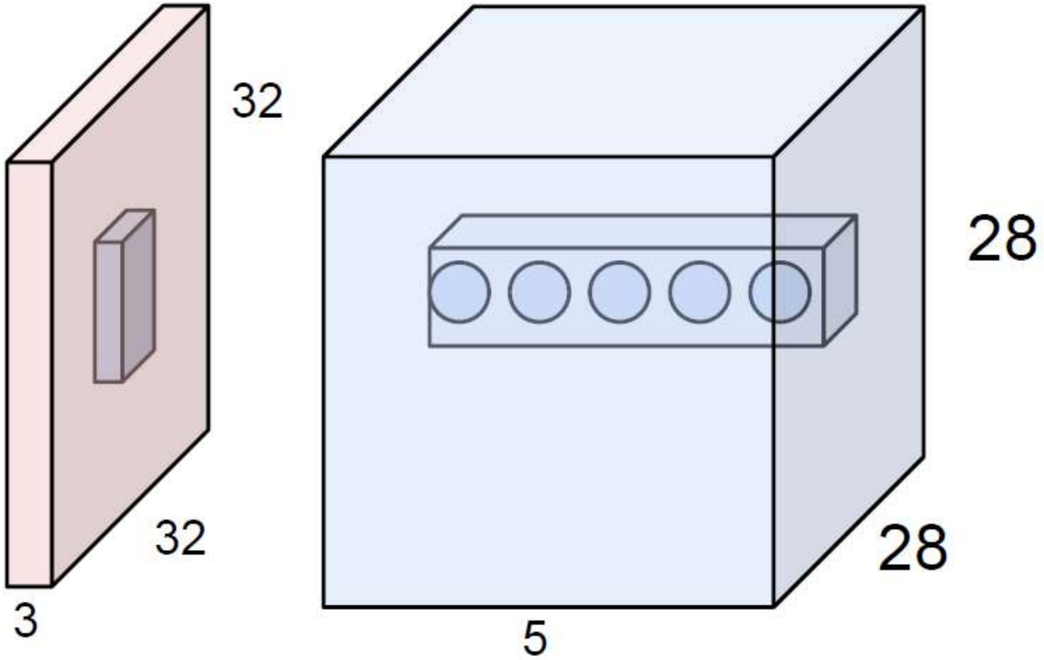
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The brain/neuron view of CONV Layer





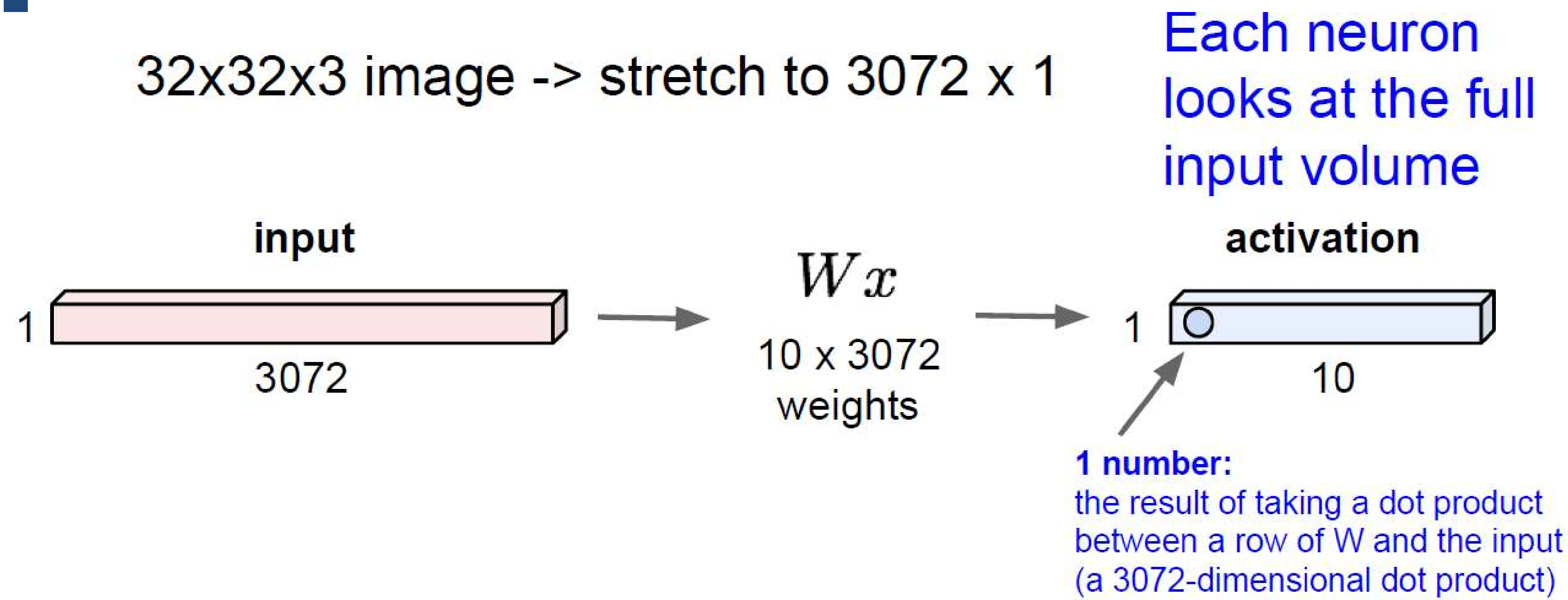
E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

http://cs231n.stanford.edu/slides/2017

全连接层

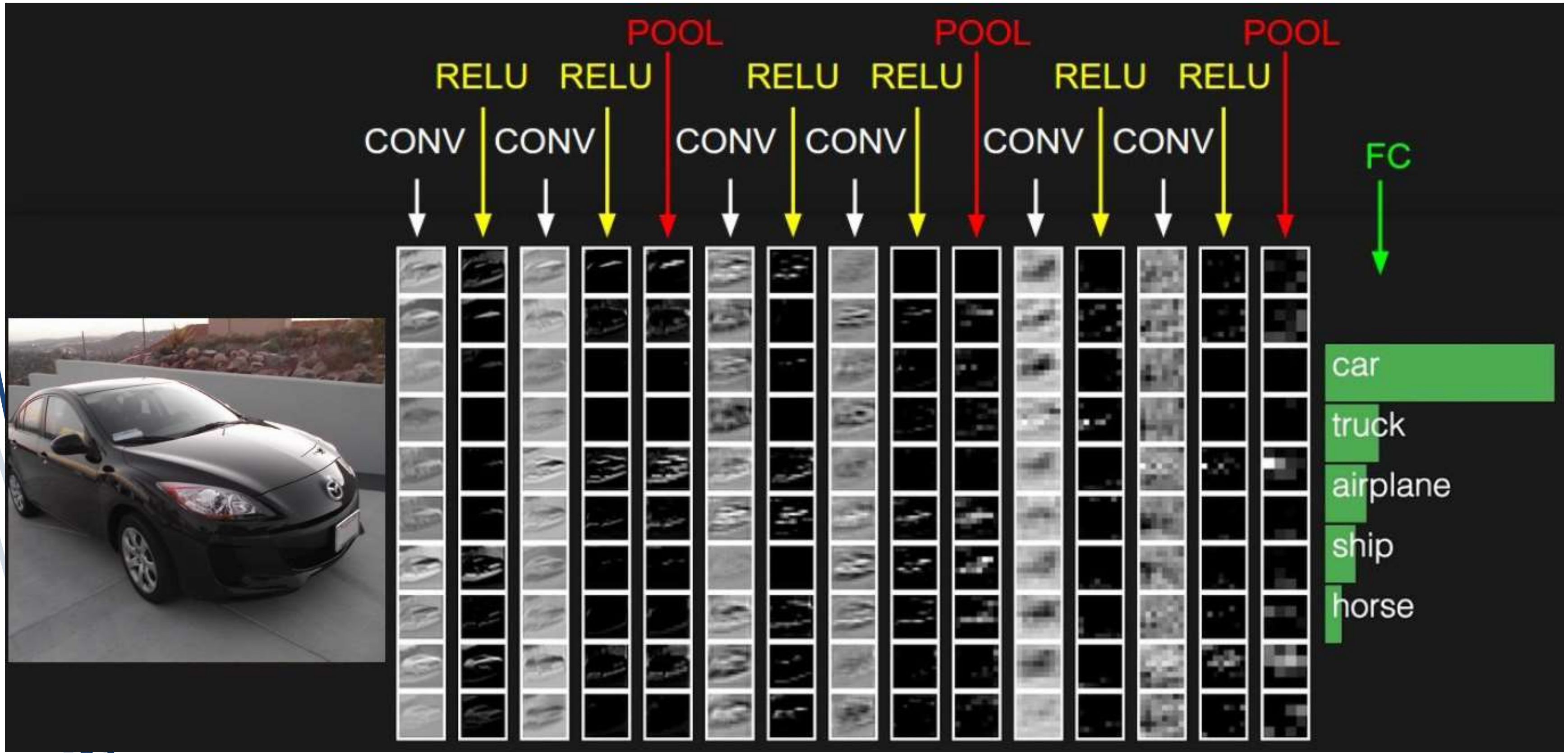
32x32x3 image -> stretch to 3072 x 1



32

http://cs231n.stanford.edu/slides/2017

多层CNN



http://cs231n.stanford.edu/slides/2017

33

4. VGG Net

- K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations, May 2015.

- VGGNet 是牛津大学计算机视觉组（Visual Geometry Group）和谷歌 DeepMind 一起研究出来的深度卷积神经网络，因而冠名为 VGG。

- VGGNet 结构中大量使用 3x3 的卷积核和 2x2 的池化核，首次将卷积神经网络的卷积深度推向更深，最为典型的 VGGNet 是 VGG16 和 VGG19
  - 16 的含义即网络中包含16个卷积层和全连接层
  - 19即即网络中包含19个卷积层和全连接层。

- VGGNet 的网络证明了卷积网络深度的重要性。
  - 深度卷积网络能够提取图像低层次、中层次和高层次的特征
  - 网络结构需要的一定的深度来提取图像不同层次的特征

◆ 作者使用了 A-E 五个不同深度水平的卷积网络进行试验，从A到E网络深度不断加深：

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

◆ 其中 D 和 E 即我们常说的 VGG16 和 VGG19。可以看到 VGG16 网络需要训练的参数数量达到了 1.38 亿个，这个数量是巨大的。以 VGG16 为例简单探究一下它的网络结构。

VGG16 各层的结构和参数如下：

C1-1 层是个卷积层，其输入输出结构如下：
输入： 224 x 224 x 3 滤波器大小： 3 x 3 x 3 滤波器个数：64
输出： 224 x 224 x 64

C1-2 层是个卷积层，其输入输出结构如下：
输入： 224 x 224 x 3 滤波器大小： 3 x 3 x 3 滤波器个数：64
输出： 224 x 224 x 64

P1 层是 C1-2 后面的池化层，其输入输出结构如下：
输入： 224 x 224 x 64 滤波器大小： 2 x 2 滤波器个数：64
输出： 112 x 112 x 64

C2-1 层是个卷积层，其输入输出结构如下：
输入： 112 x 112 x 64 滤波器大小： 3 x 3 x 64 滤波器个数：128
输出： 112 x 112 x 128

C2-2 层是个卷积层，其输入输出结构如下：
输入： 112 x 112 x 64 滤波器大小： 3 x 3 x 64 滤波器个数：128
输出： 112 x 112 x 128