《神经网络理论与应用》第四讲

Neural Network Theory and Applications

主讲教师：郑伟龙
助教：尹昊龙、史涵雯

上海交通大学计算机科学与工程系

weilong@sjtu.edu.cn
http://bcmi.sjtu.edu.cn

Outline of Lecture Four

- Introduction to Deep Learning
- Convolutional Neural Networks

Renaissance of Neural Network--- “Deep Learning,” 2006

- Geoff Hinton invented Deep Belief Networks (DBN) to make neural net learning fast and effective; *Science, 2006*
  - Pre-train each layer from bottom up
  - Each pair of layers is an Restricted Boltzmann Machine (RBM)
  - Jointly fine-tune all layers using back-propagation

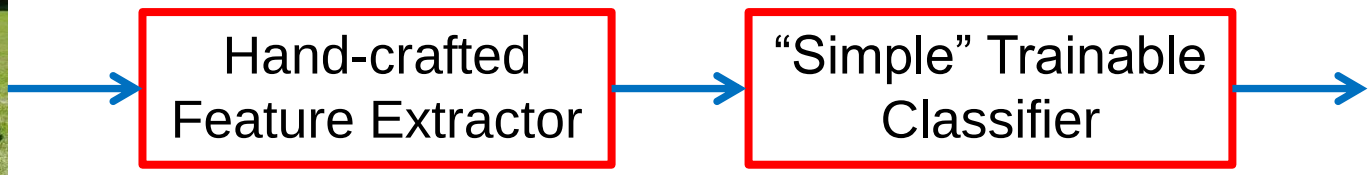
Major Breakthrough in 2006

- Biological Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed

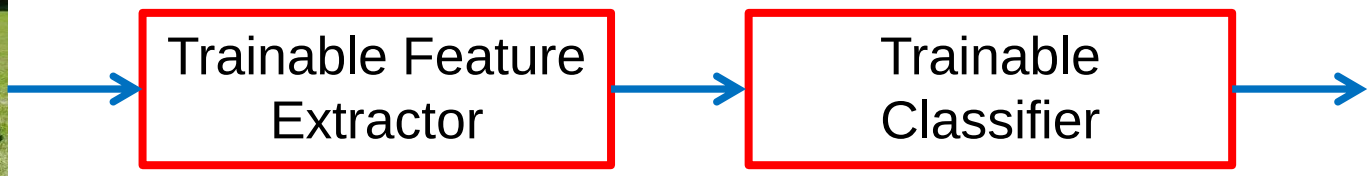
- Unsupervised feature learners:
  - RBMs
  - Auto-encoder variants
  - Sparse coding variants

Deep Learning = Learning Features

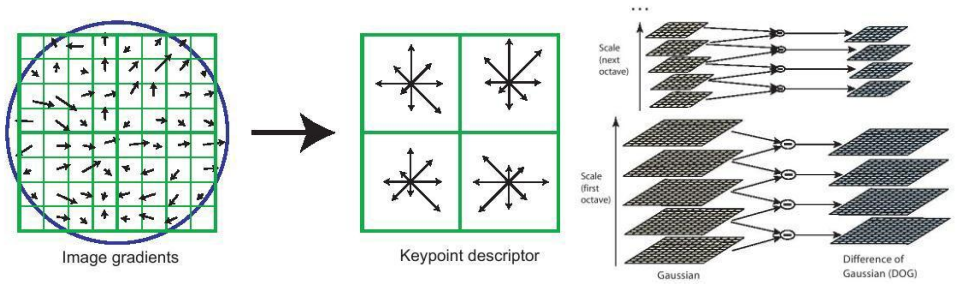
- Traditional model of pattern recognition
  - Fixed features + trainable classifier



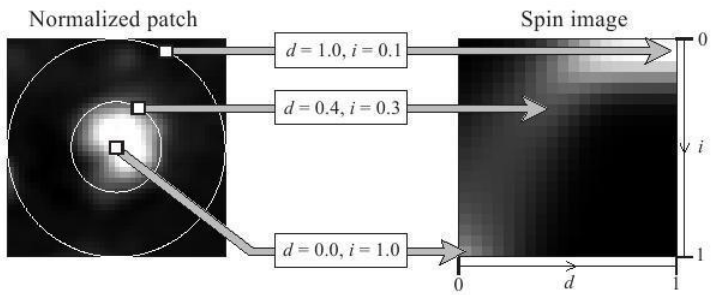
- Deep Learning
  - Trainable features + trainable classifier



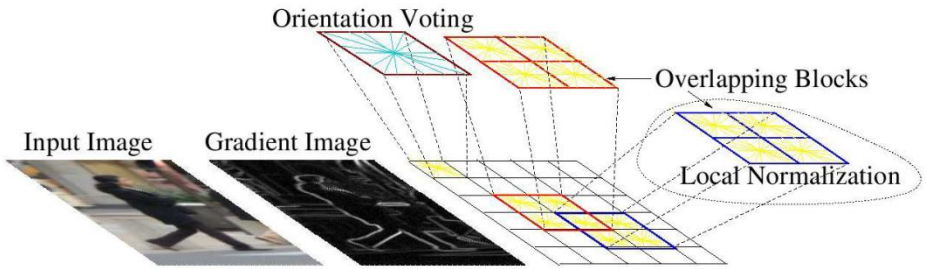
Computer vision features



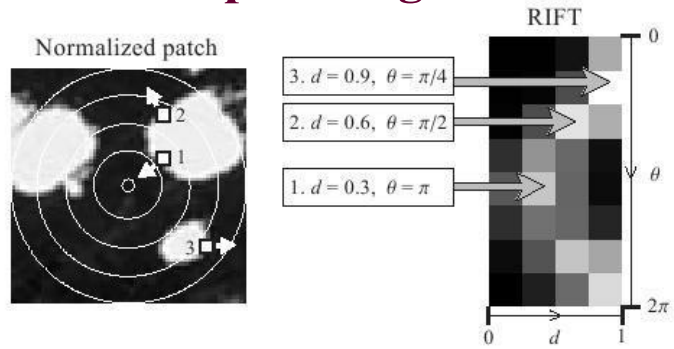
SIFT



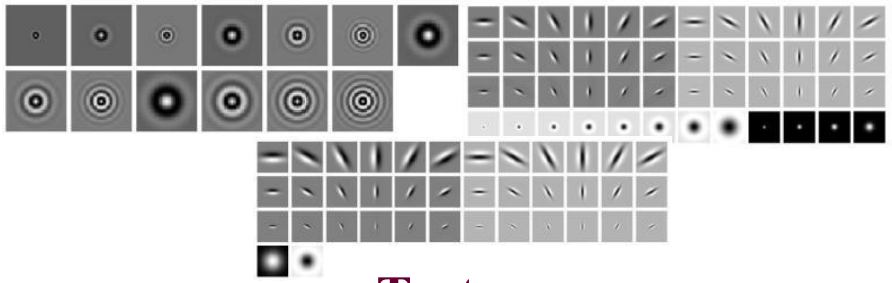
Spin image



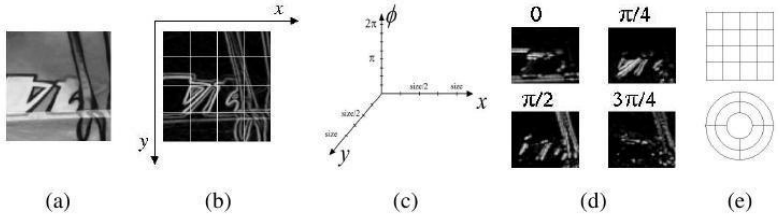
HoG



RIFT



Textons

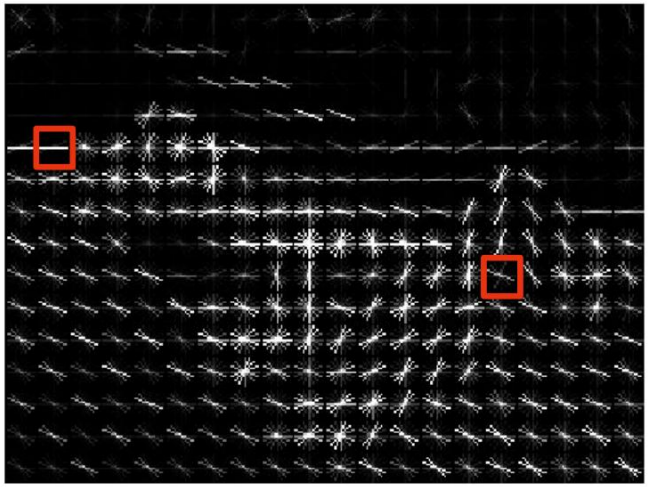


GLOH

Computer vision features

Example: Histogram of Oriented Gradients (HoG)

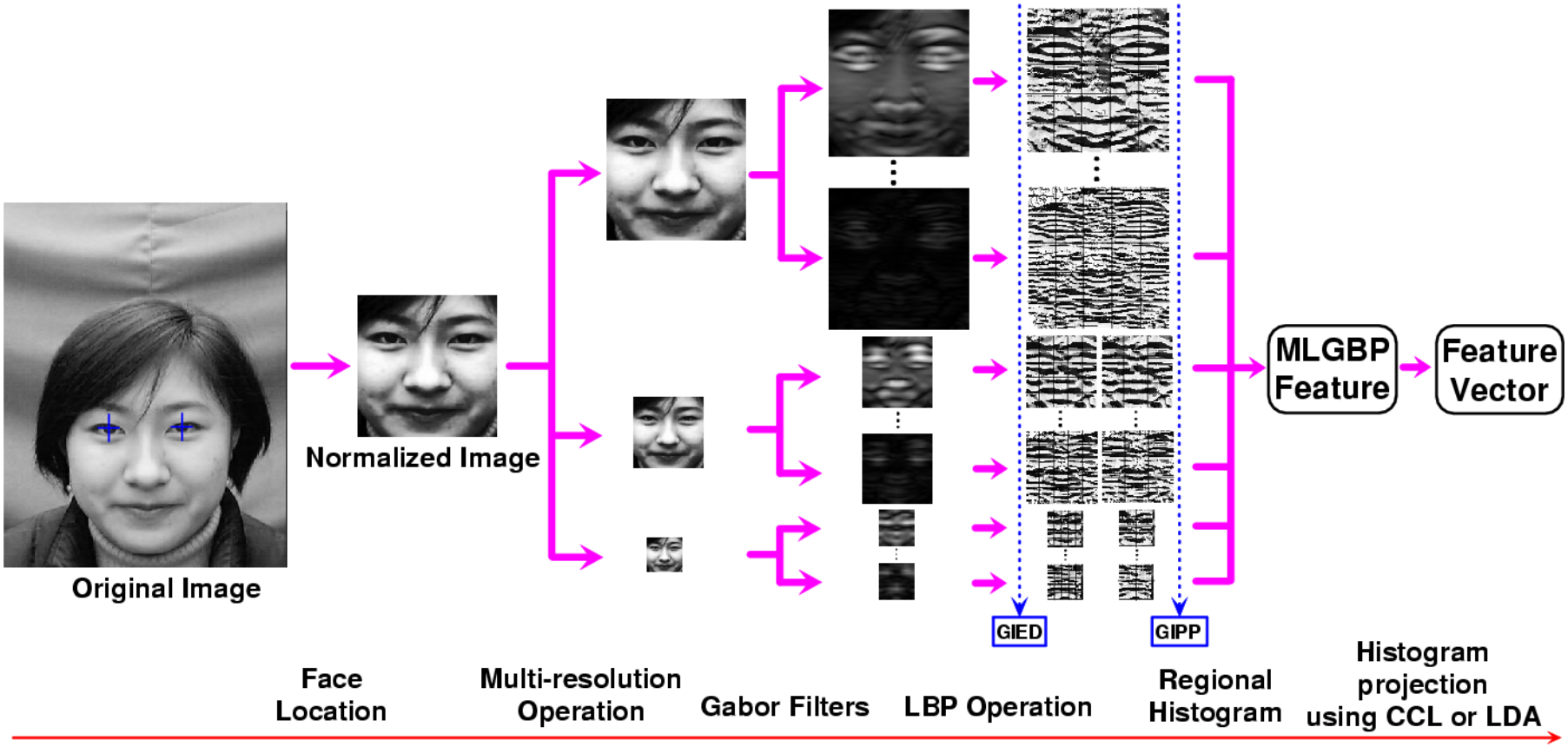
Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins



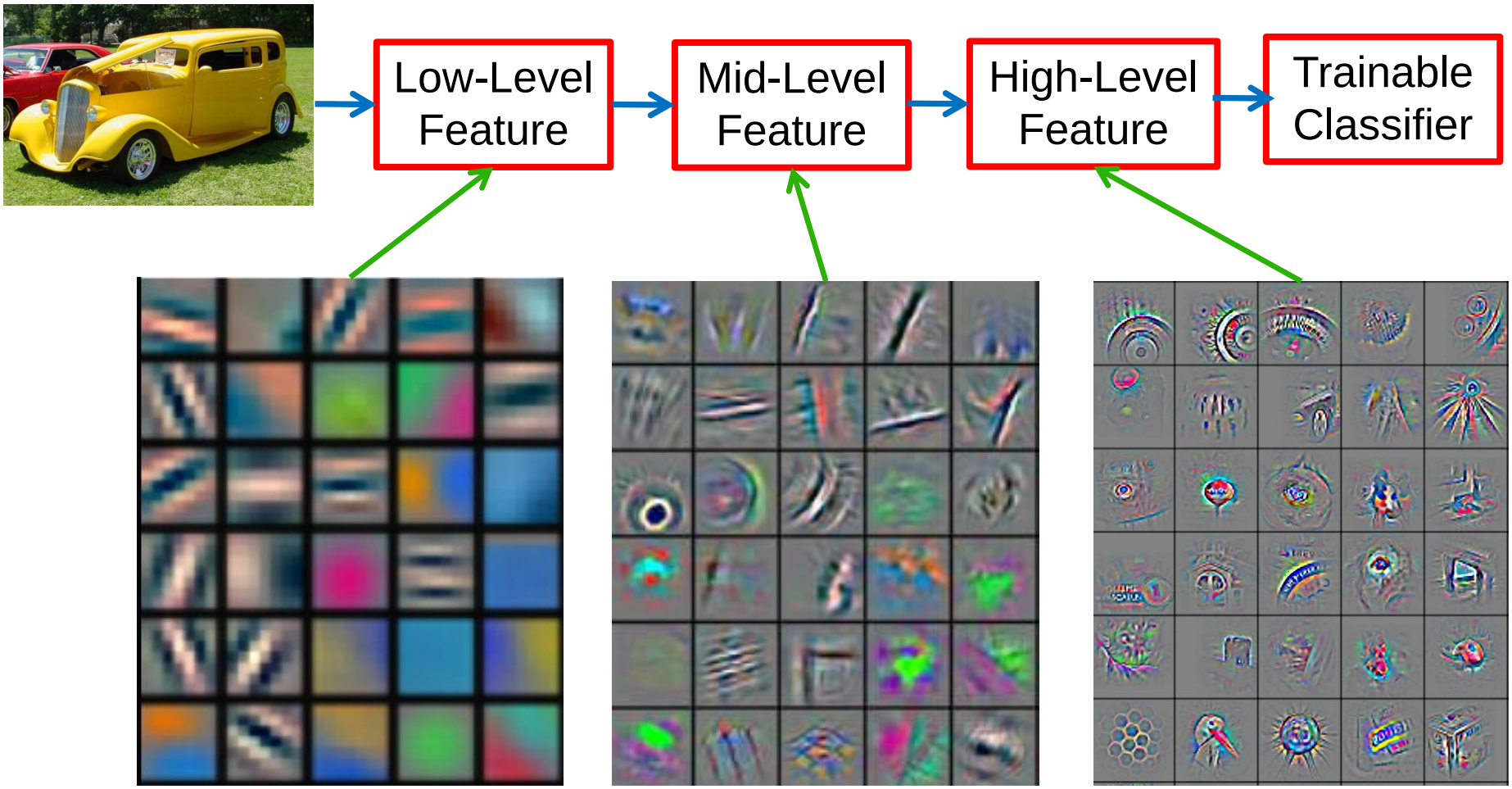
Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

人脸特征（MLGBP）


Original Image
Normalized Image
GIED
GIPP
MLGBP Feature
Feature Vector
Face Location
Multi-resolution Operation
Gabor Filters
LBP Operation
Regional Histogram
Histogram projection using CCL or LDA

Deep Learning = Learning Hierarchical Representations

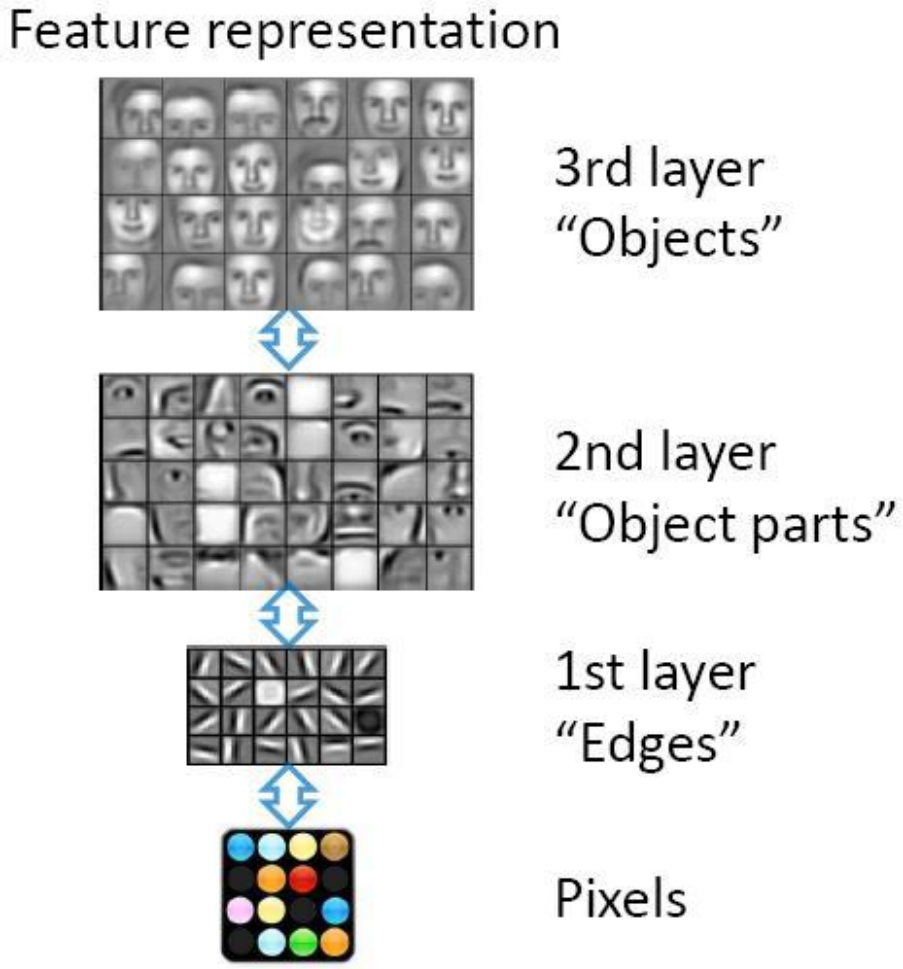


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Learning Feature Hierarchy

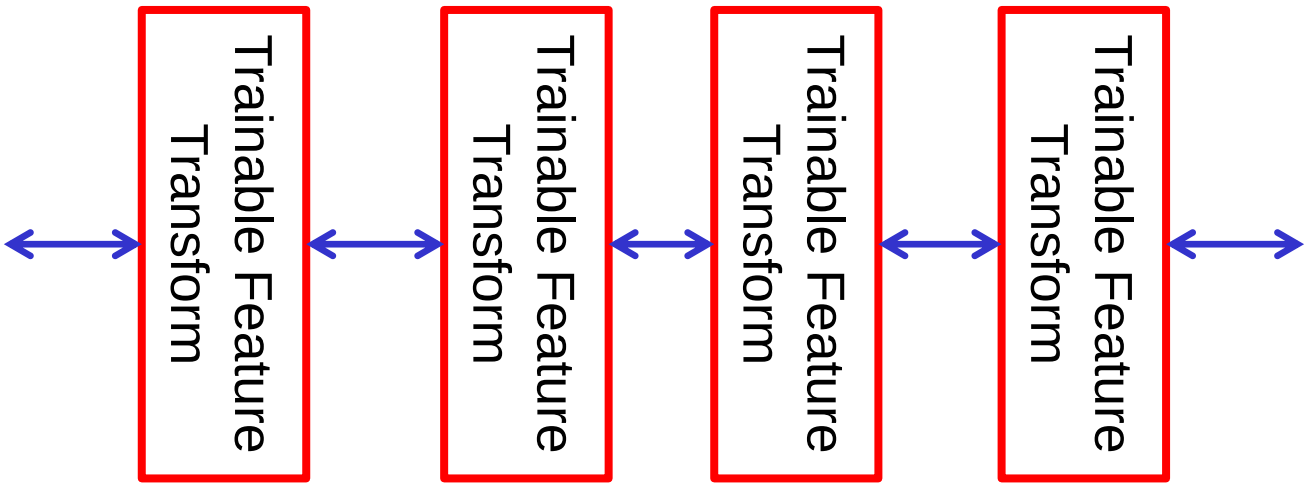
Deep Learning

- Deep architectures can be representationally efficient.
- Natural progression from low level to high level structures.
- Can share the lower-level representations for multiple tasks



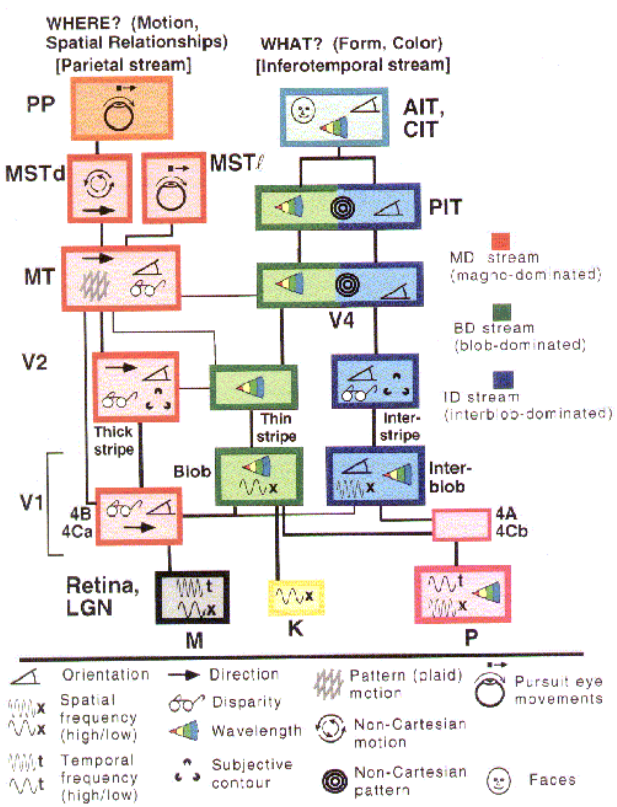
Trainable Feature Hierarchy

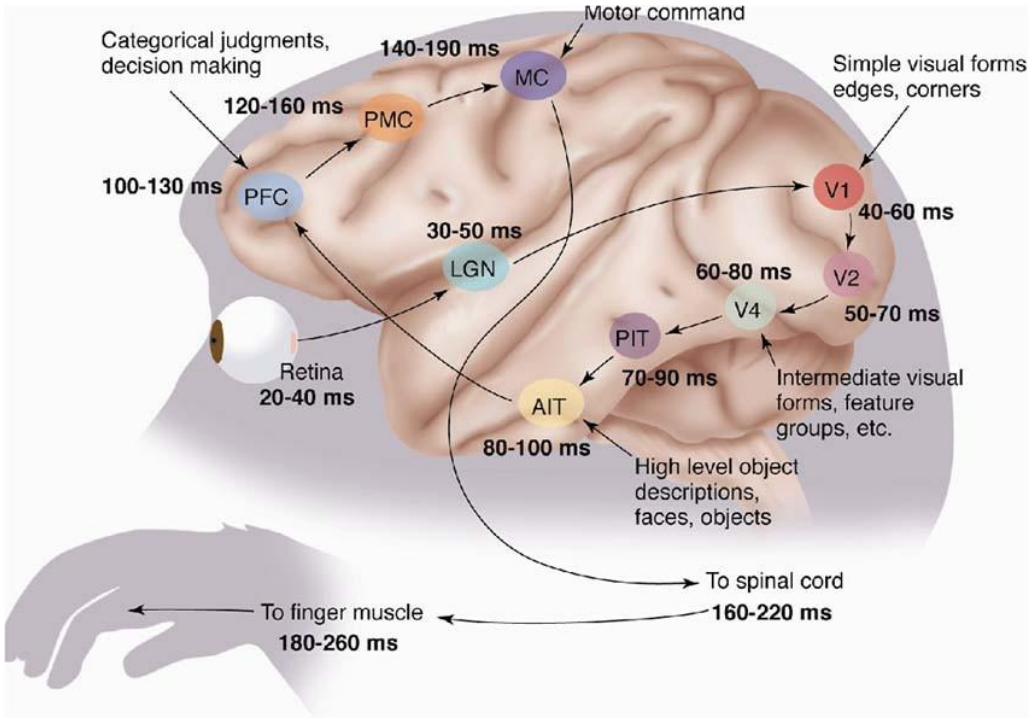
- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → … → phone → phoneme →word



The Mammalian Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations

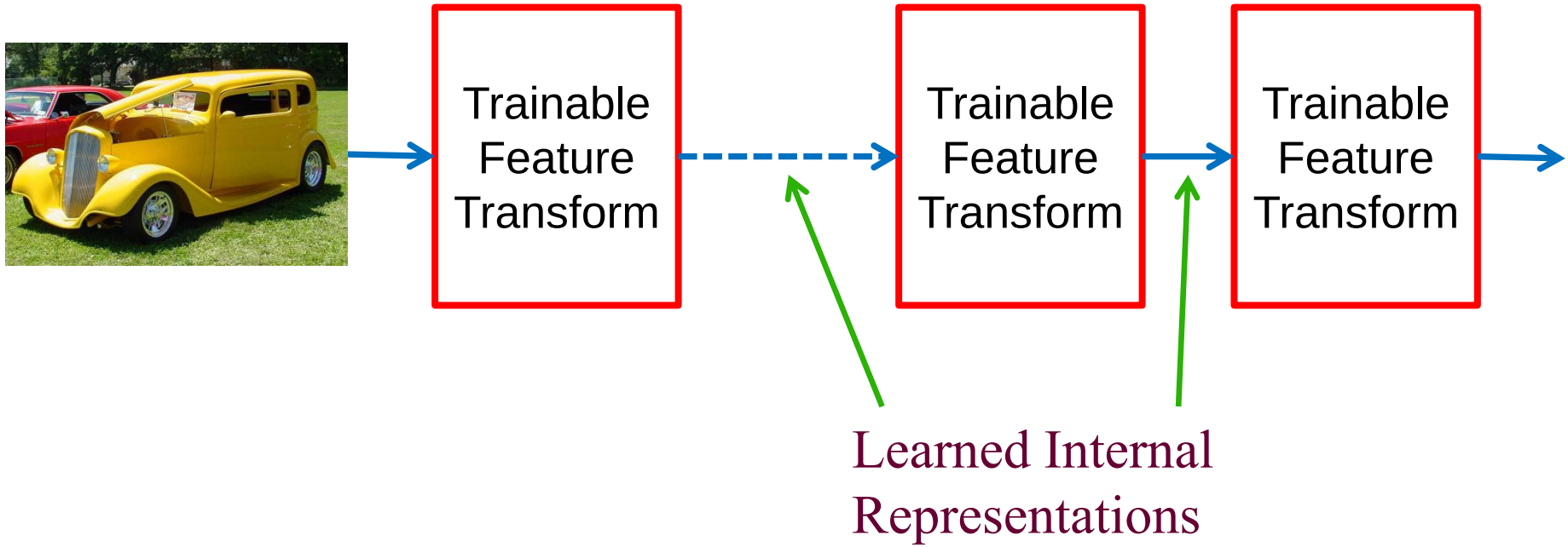




外侧膝状体（Lateral Geniculate Nucleus, LGN）

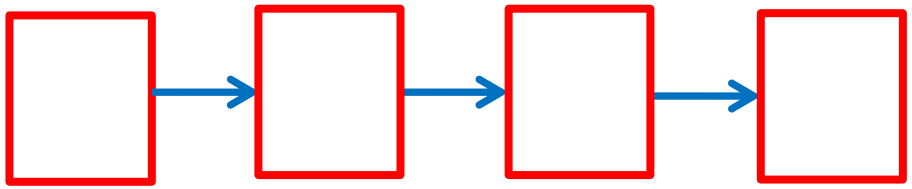
Trainable Feature Hierarchies

- Each module transforms its input representation into a higher-level one.
- High-level features are more global and more invariant
- Low-level features are shared among categories

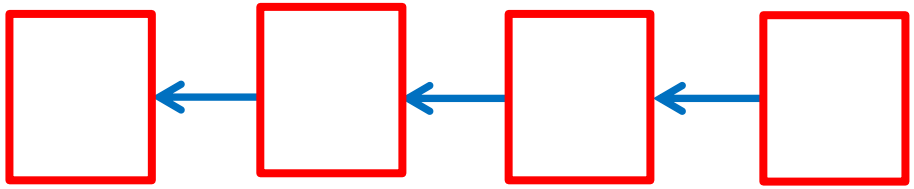


Three Types of Deep Architectures

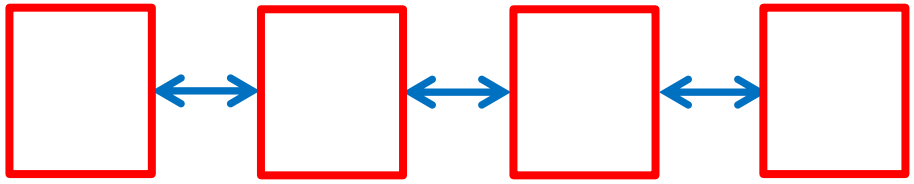
- Feed-Forward: multilayer neural nets, convolutional nets



- Feed-Back: Stacked Sparse Coding, Deconvolutional Nets



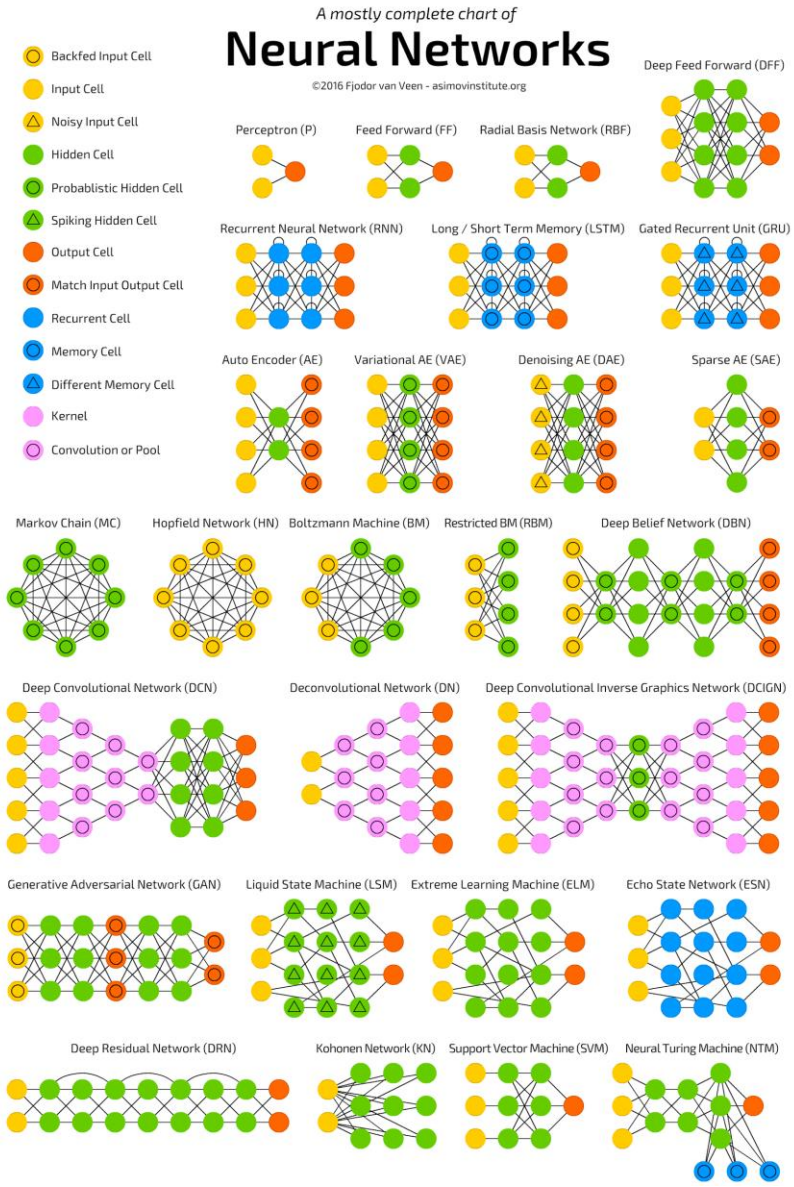
- Bi-Drectional: Deep Boltzmann Machines, Stacked Auto-Encoders



Three Types of Training Protocols

- Purely Supervised
  - Initialize parameters randomly
  - Train in supervised mode
  - Used in most practical systems for speech and image recognition
- Unsupervised, layerwise + supervised classifier on top
  - Train each layer unsupervised, one after the other
  - Train a supervised classifier on top, keeping the other layers fixed
  - Good when very few labeled samples are available
- Unsupervised, layerwise + global supervised fine-tuning
  - Train each layer unsupervised, one after the other
  - Add a classifier layer, and retrain the whole thing supervised
  - Good when label set is poor

Various Neural Networks



F. Van Veen, "The Neural Network Zoo" (2016)

Why Deep Learning ?

- Biological Plausibility – e.g. Visual Cortex
- Learning features, not just handcrafting them
- Unsupervised feature learning
- Learning multiple levels of representation
- Highly varying functions can be efficiently represented with deep architectures
  - Less weights/parameters to update than a less efficient shallow representation
- Sub-features created in deep architecture can potentially be shared between multiple tasks
  - Type of Transfer/Multi-task learning

Outline of Lecture Six

- Introduction to Deep Learning
- **Convolutional Neural Networks**

Convolutional Neural Networks

卷积神经网络

- CNN的历史
- CNN的生物学基础
- CNN的结构元件
- CNN的结构：LeNet-5
- CNN VS 全连接网络

Image Classification

Image Classification: A core task in Computer Vision

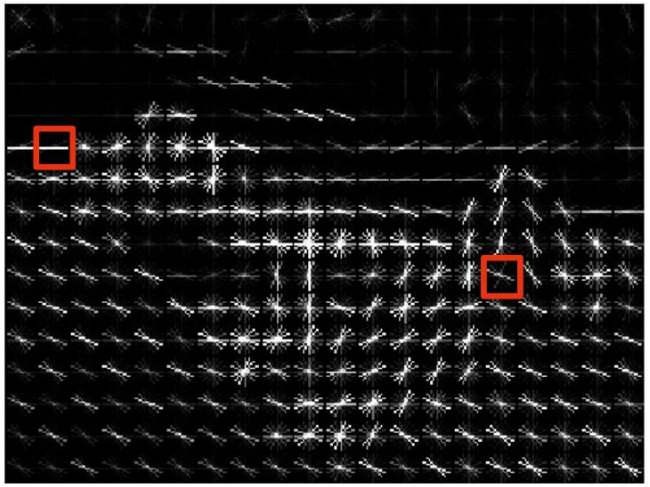
(assume given a set of labels)
{dog, cat, truck, plane, ...}

cat
dog
bird
deer
truck

Image Classification

Example: Histogram of Oriented Gradients (HoG)

Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins



Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image Classification

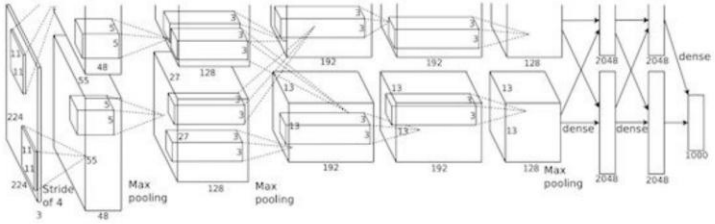
Image features vs. ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

training

Image Classification

Last Time: Neural Networks

Linear score function: $f = Wx$

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

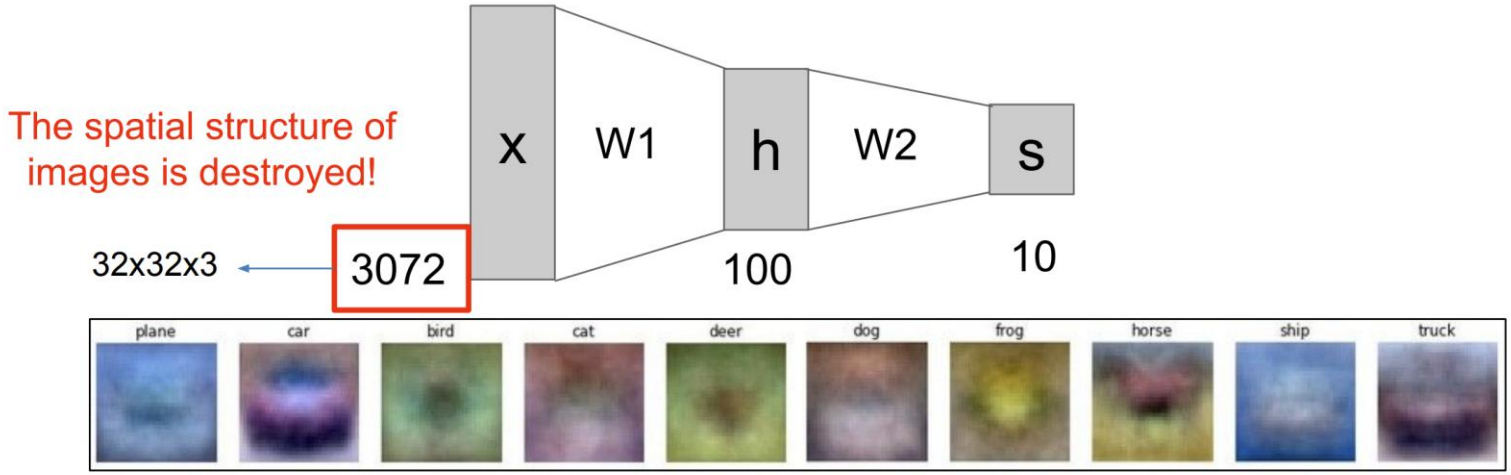
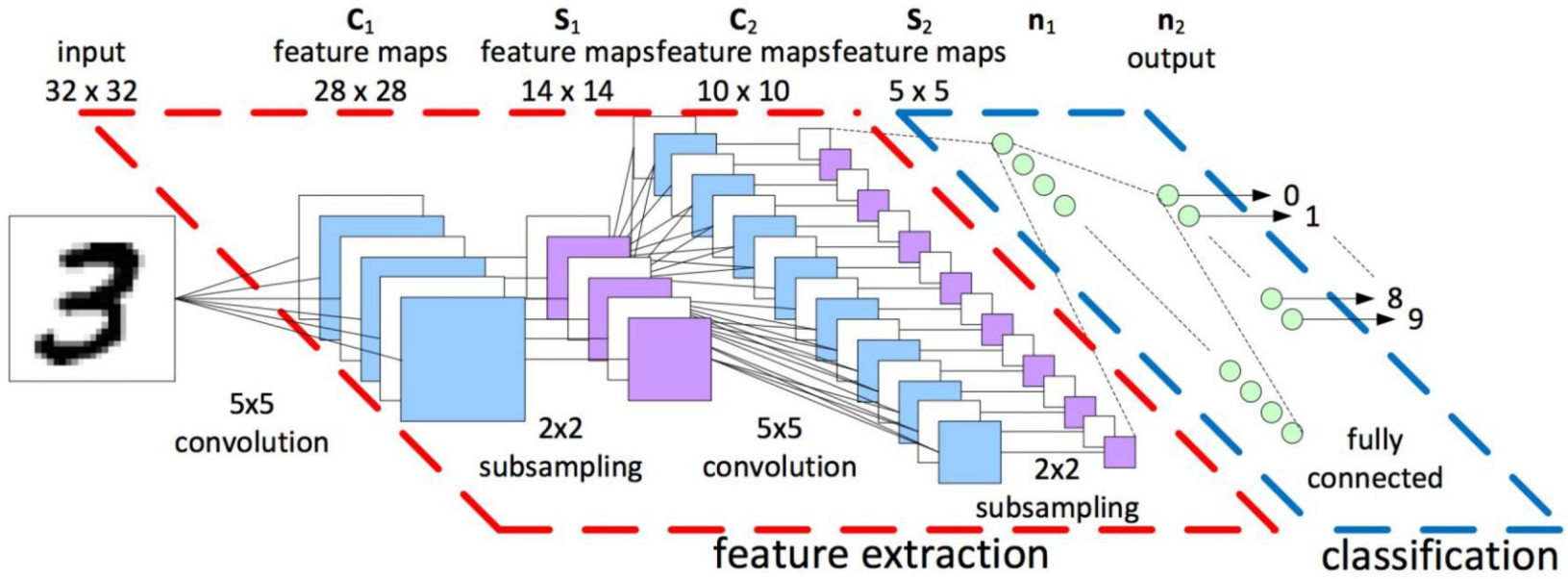


Image Classification

Next: Convolutional Neural Networks

MNIST example



CNN的历史

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

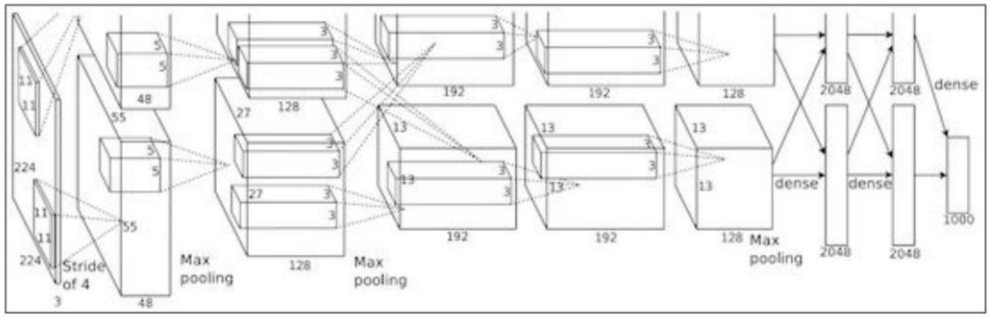


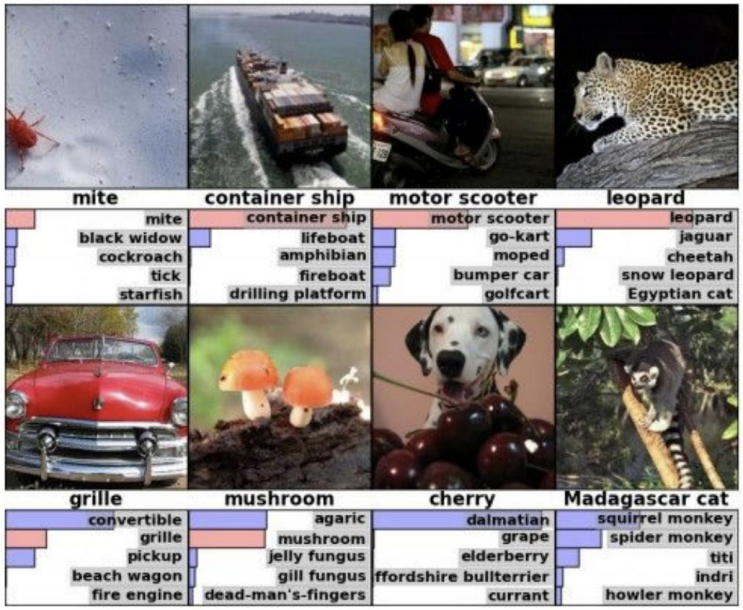
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

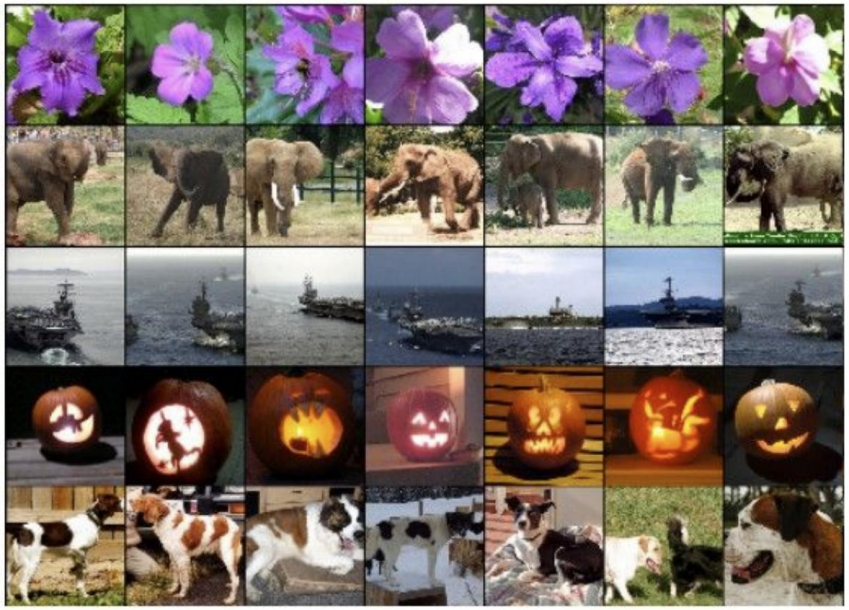
CNN的历史

Fast-forward to today: ConvNets are everywhere

Classification



Retrieval



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

CNN的历史

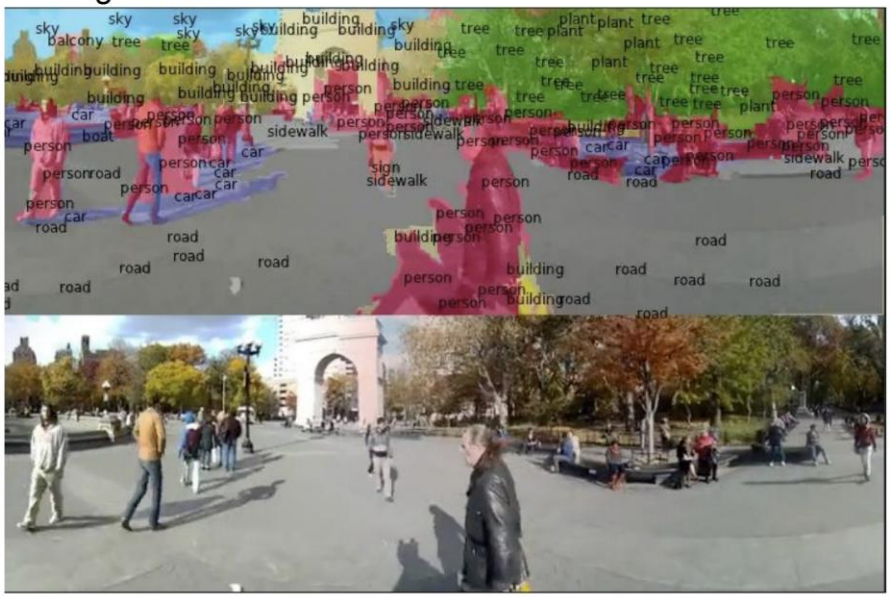
Fast-forward to today: ConvNets are everywhere

Detection

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

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

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

CNN的历史

No errors

A white teddy bear sitting in the grass

Minor errors

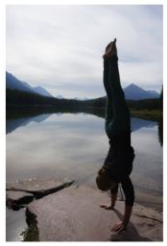
A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor



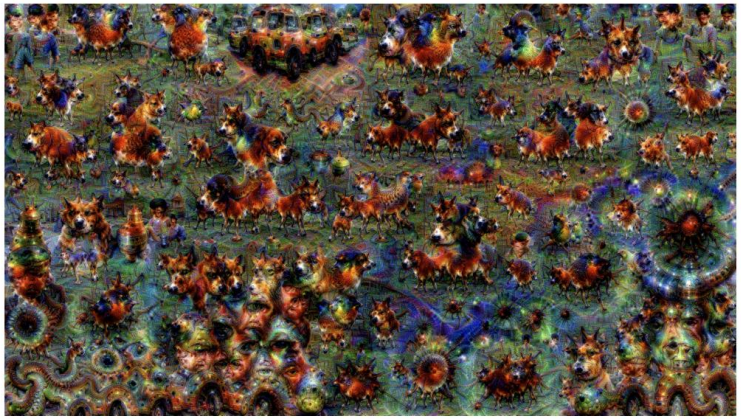
A woman standing on a beach holding a surfboard

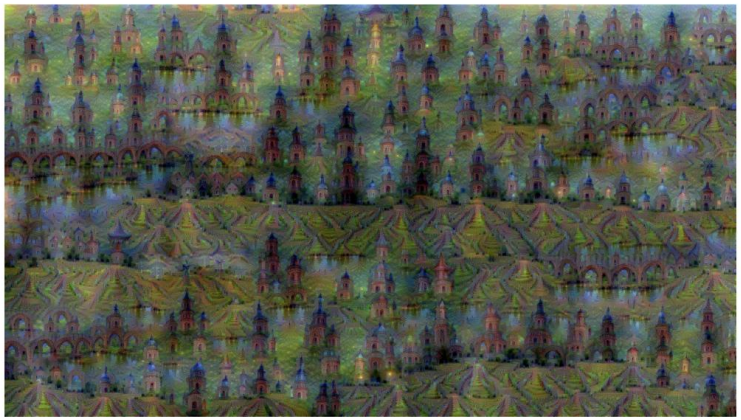
Image Captioning

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

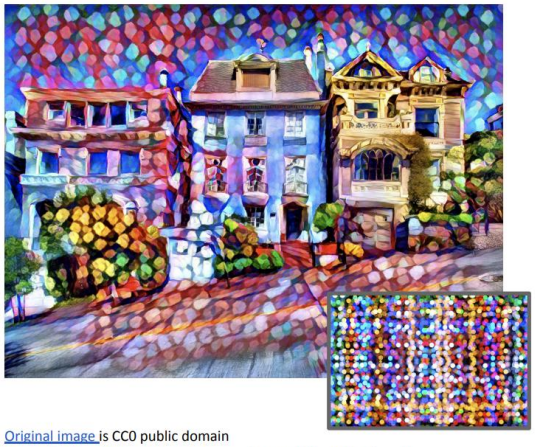
Captions generated by Justin Johnson using Neuraltalk2

CNN的历史

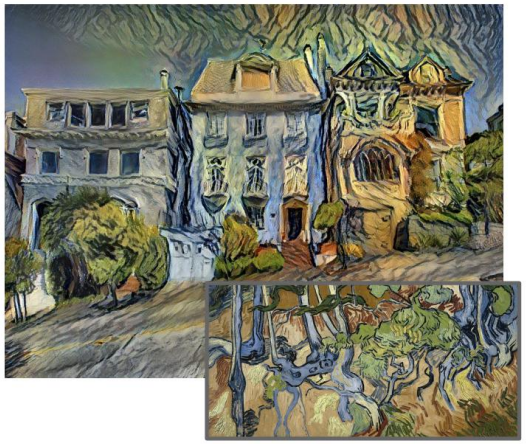




Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

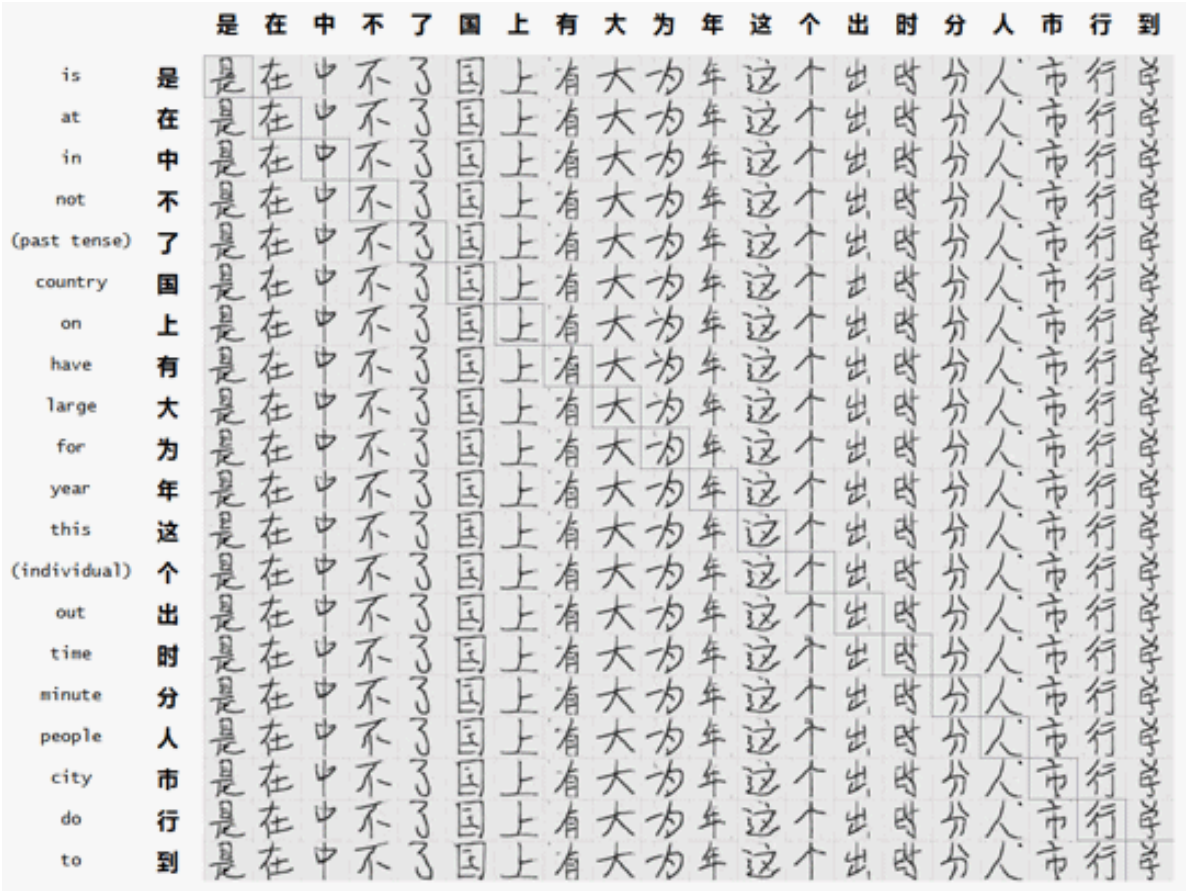


Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission



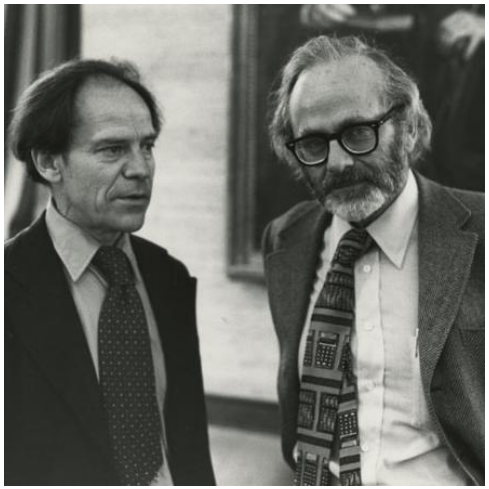
Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

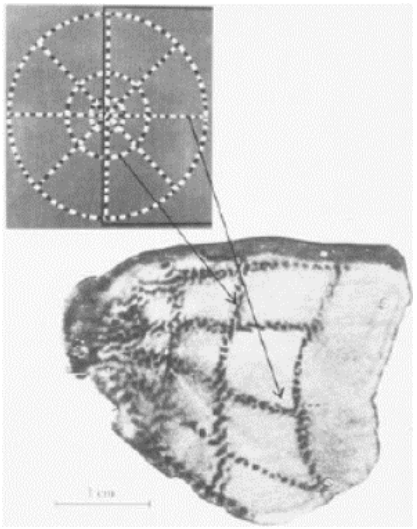
CNN的历史

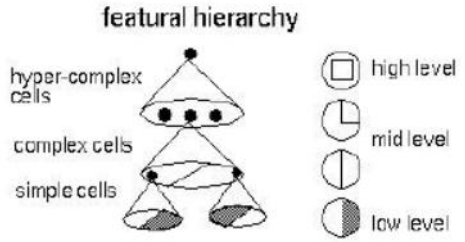

是 在 中 不 了 国 上 有 大 为 年 这 个 出 时 分 人 市 行 到
is 是
at 在
in 中
not 不
(past tense) 了
country 国
on 上
have 有
large 大
for 为
year 年
this 这
(individual) 个
out 出
time 时
minute 分
people 人
city 市
do 行
to 到

CNN的生物学基础: Hubel & Wiesel

- 猫的大脑中第一层的视觉神经元（V1），只处理局部视觉中的简单基础结构信息（如某一方向的直线、点），且一种神经元只对一种特定的基础结构做出反应。
- 而且V1神经元是会保留拓扑结构信息的，即相邻的神经元的感受野在图像中也相邻。
- 视觉神经元是有层次的。

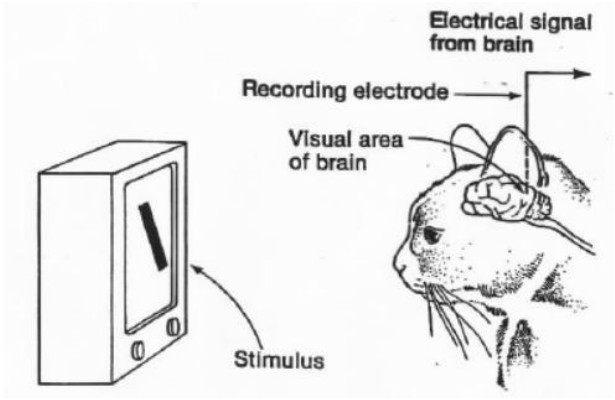


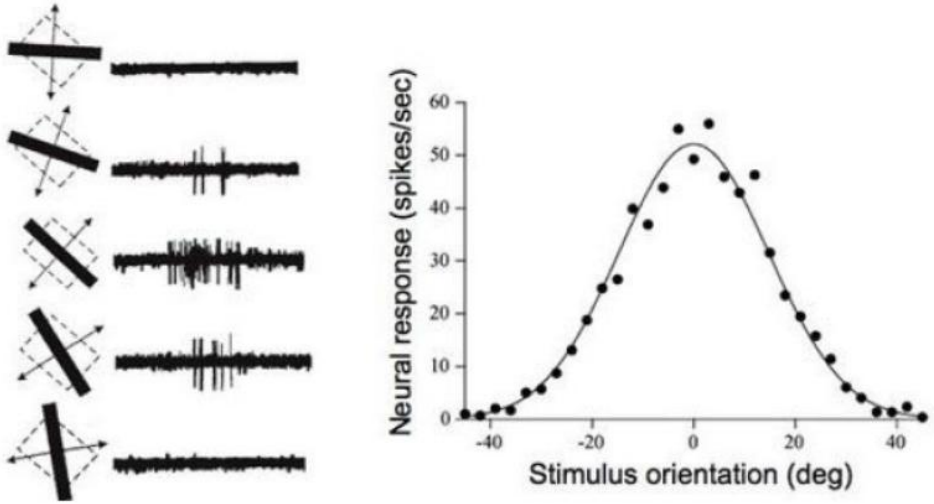




Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." *The Journal of physiology* 160.1 (1962): 106-154.

CNN的生物学基础



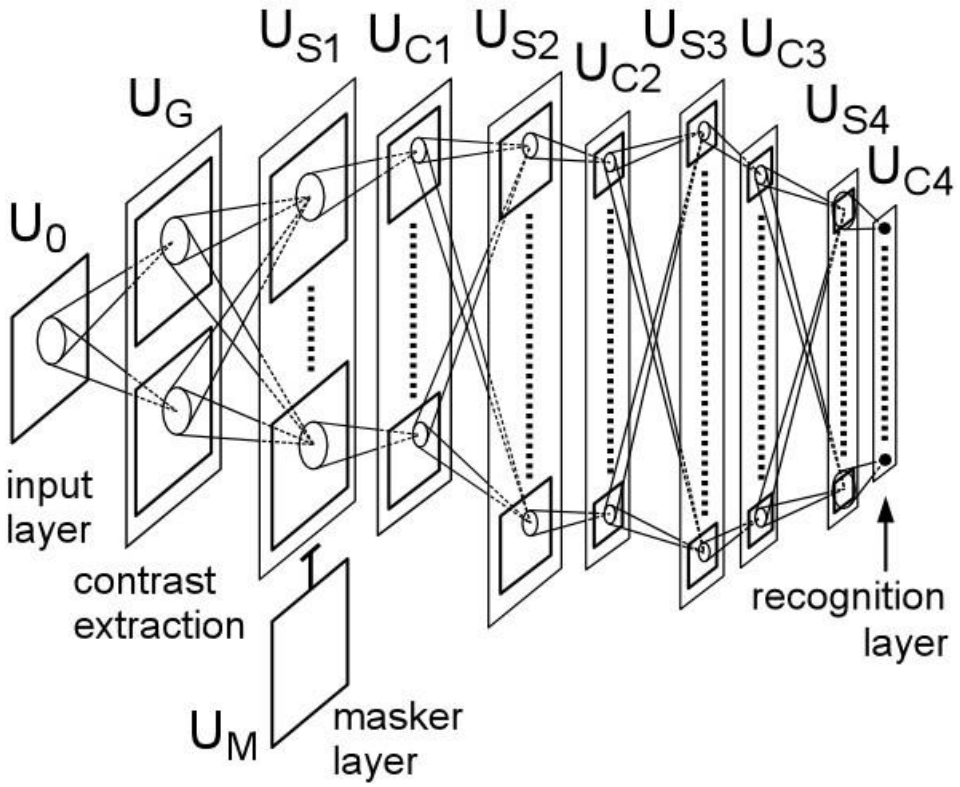


猫的大脑中第一层的视觉神经元（V1），只处理**局部视觉**中的**简单基础结构信息**（如某一方向的直线、点），且一种神经元只对一种特定的基础结构做出反应。

1980： Neurocognitron

Fukushima 1980: Neurocognitron

(“三明治” 结构, 局部感受野, 池化层)

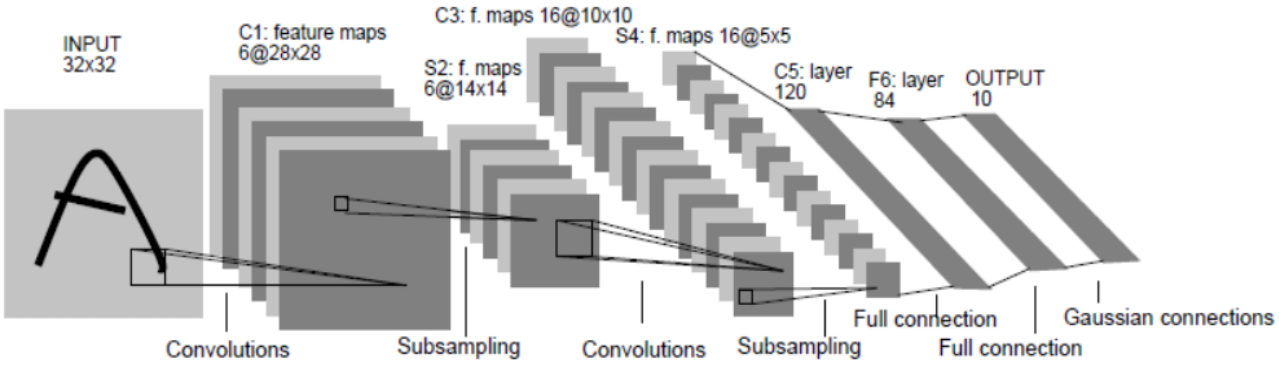


2020年富兰克林学院鲍尔奖之科学成就奖

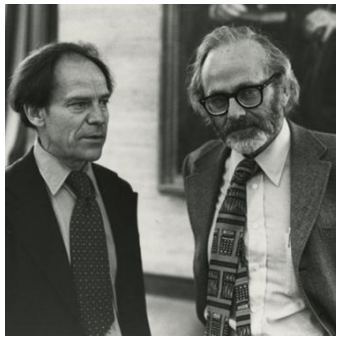
获奖理由：他将神经科学和工程学的原理联系起来，发明了一种具有视觉模式识别能力的神经网络，从而推动了当今人工智能的兴起。

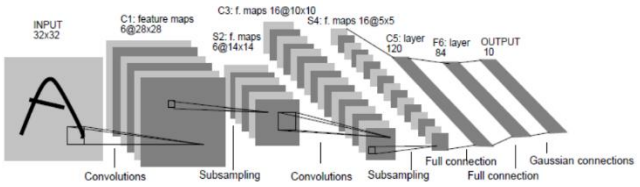
1998: 卷积神经网络（CNN）

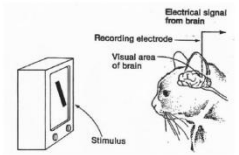
- Lecun 1998: LeNet-5 (工业界的应用：手写数字识别）

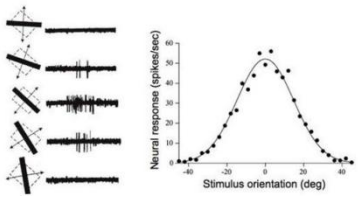


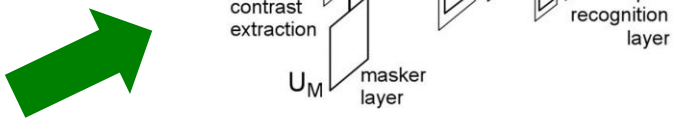
卷积神经网络：受生物神经机理启发的模型







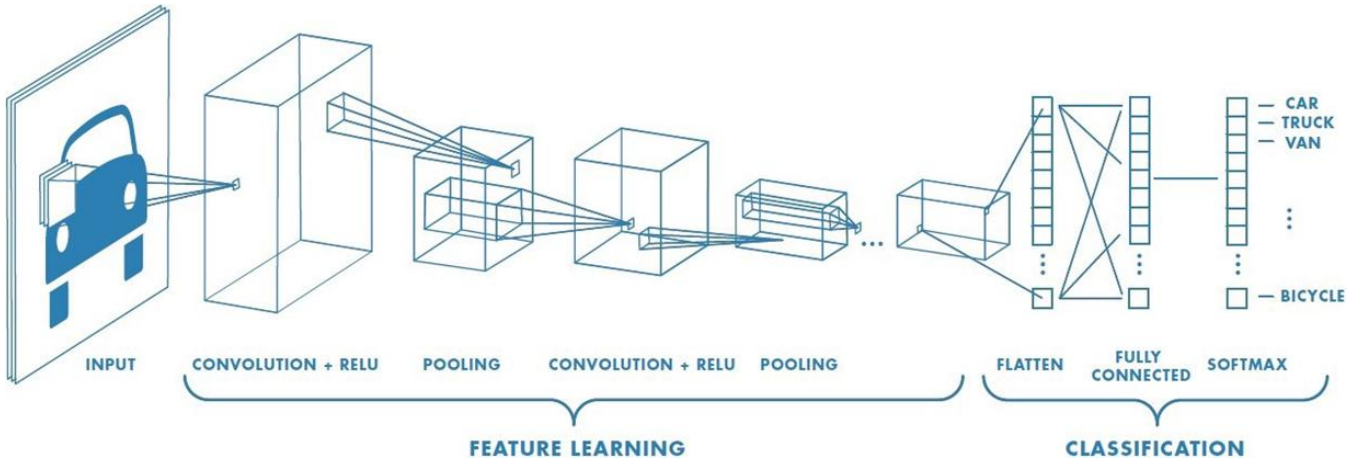




CNN的结构元件

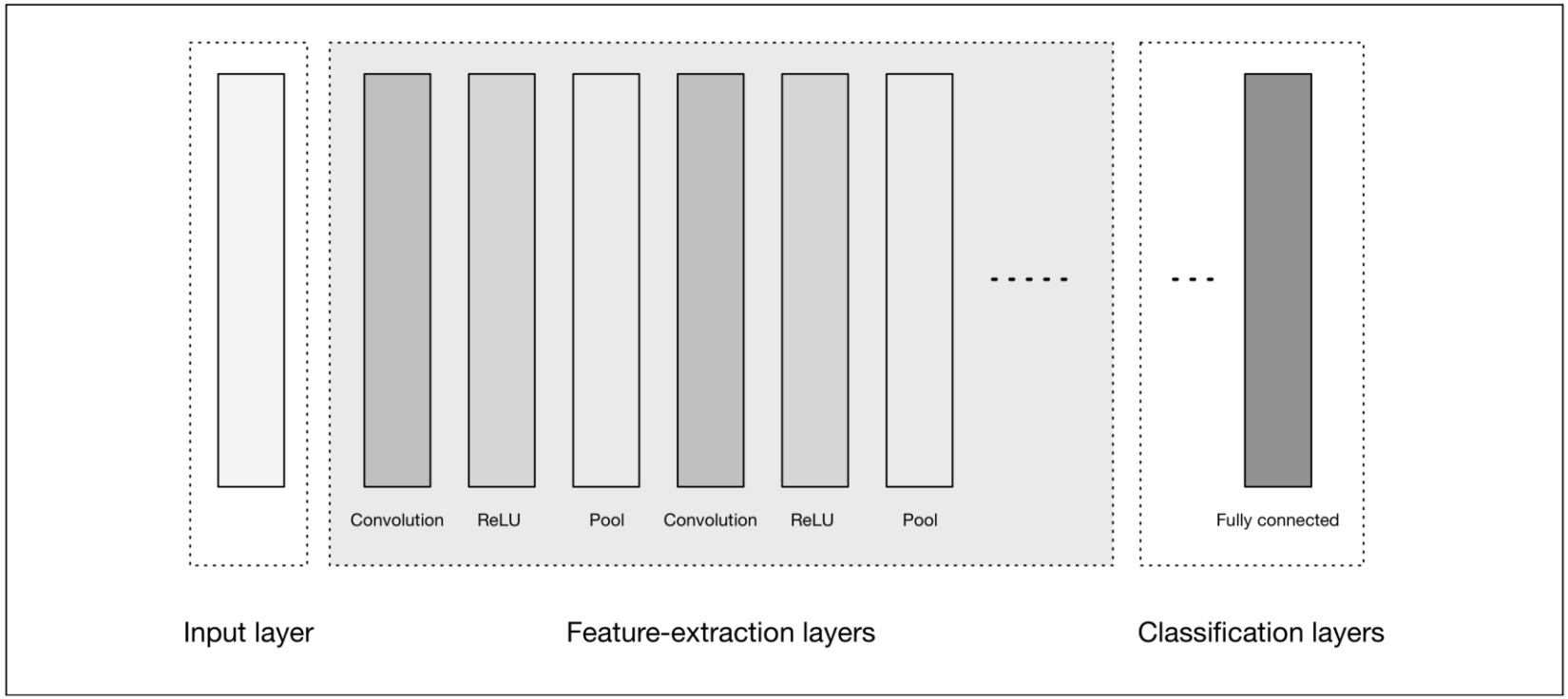
- 卷积的定义
- 卷积层
- 激活层
- 池化层
- 全连接层
- 损失函数

CNNs and Computer Vision



- 前向神经网络
- 使用BP算法调节参数
- 面向计算机视觉问题开发

High-level General CNN Architecture

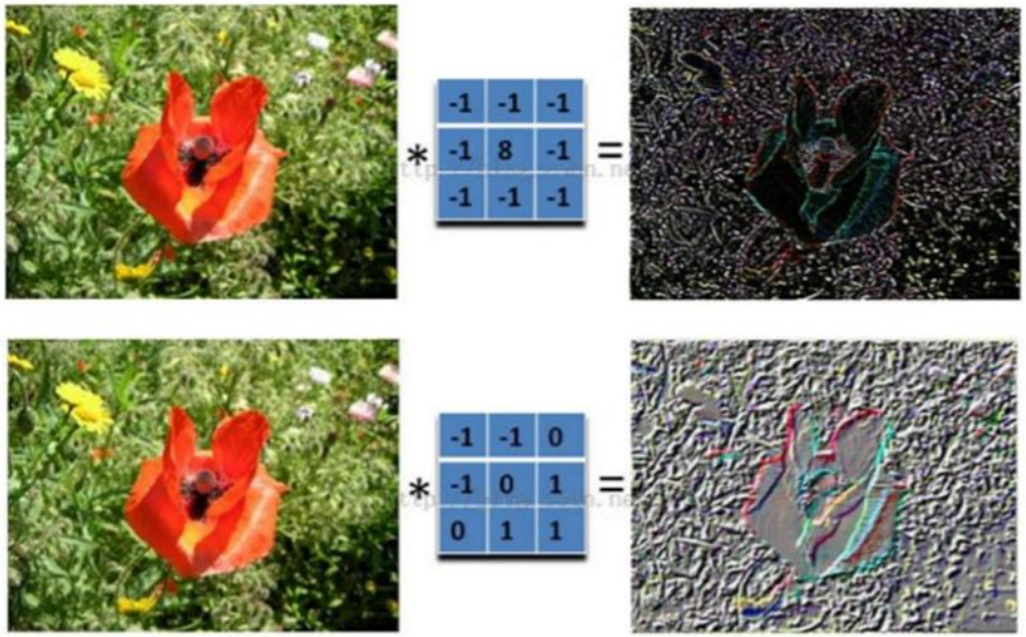


1. Input layer
2. Feature-extraction (learning) layers
3. Classification layers

卷积的定义

- 在泛函分析中，卷积(Convolution)是通过两个函数f 和g 生成第三个函数的一种数学算子，表征函数f 与g经过翻转和平移的重叠部分的面积。如果将参加卷积的一个函数看作区间的指示函数，卷积还可以被看作是“滑动平均”的推广。
- 卷积是在信号与线性系统的基础上或背景中出现的，讨论的就是信号经过一个线性系统以后发生的变化（就是输入，输出和所经过的所谓系统，这三者之间的数学关系）。
- 实际上，线性系统的传递函数和输入信号，在数学上的形式就是所谓的卷积关系(一维卷积)。而数字图像处理中的卷积关系是二维卷积。

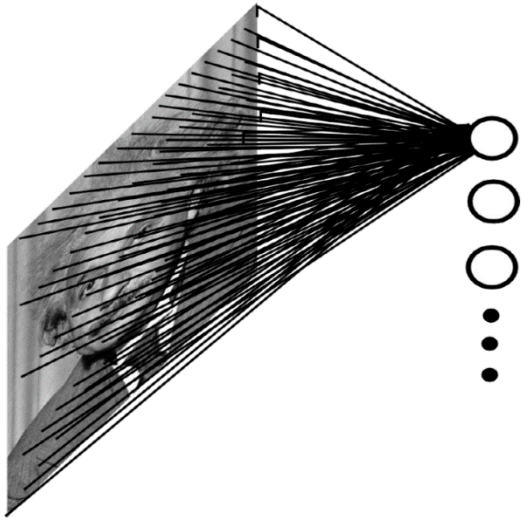
图像上的卷积

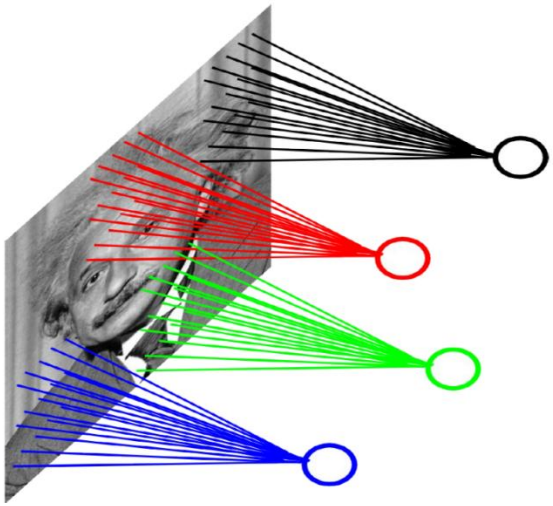


- 左边是图像输入，中间部分就是滤波器filter（带着一组固定权重的神经元），
- 不同的滤波器会得到不同的输出数据，比如颜色深浅、轮廓。
- 如果想提取图像的不同特征，则用不同的滤波器，提取想要的关于图像的特定信息：颜色深浅或轮廓

Locally connected and Share Weights

- Example: 1000*1000 image
  - Fully-connected, 1000,000 hidden units
  - 1000 *1000*1000,000= 10^{12} parameters
  - Locally-connected, 1000,000 hidden units 10x10 fields = 10^{8} parameters
  - Local connections capture local dependencies
  - 1000,000 hidden units share the same weights
  - The number of parameters are decreased from 10^{8} to 100!

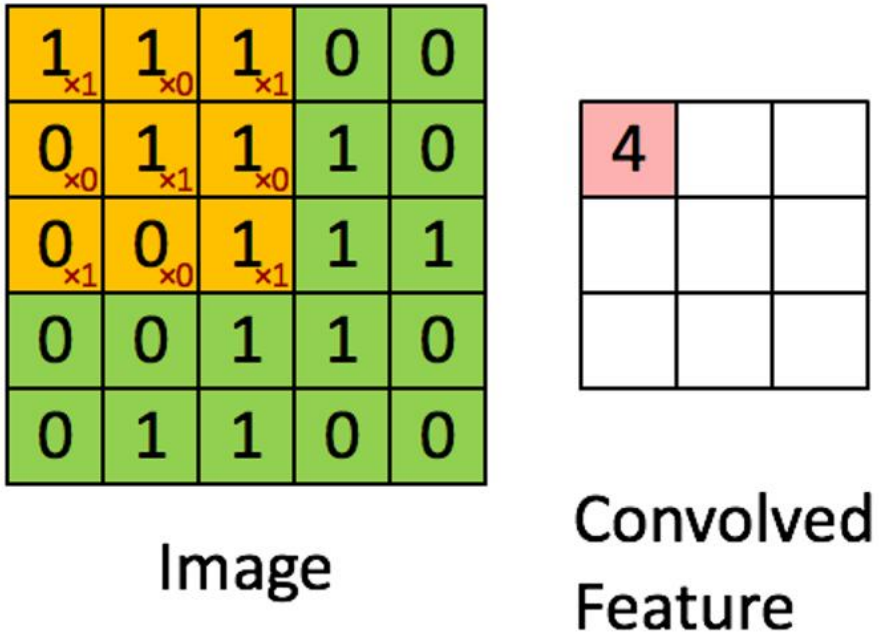




CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长S在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

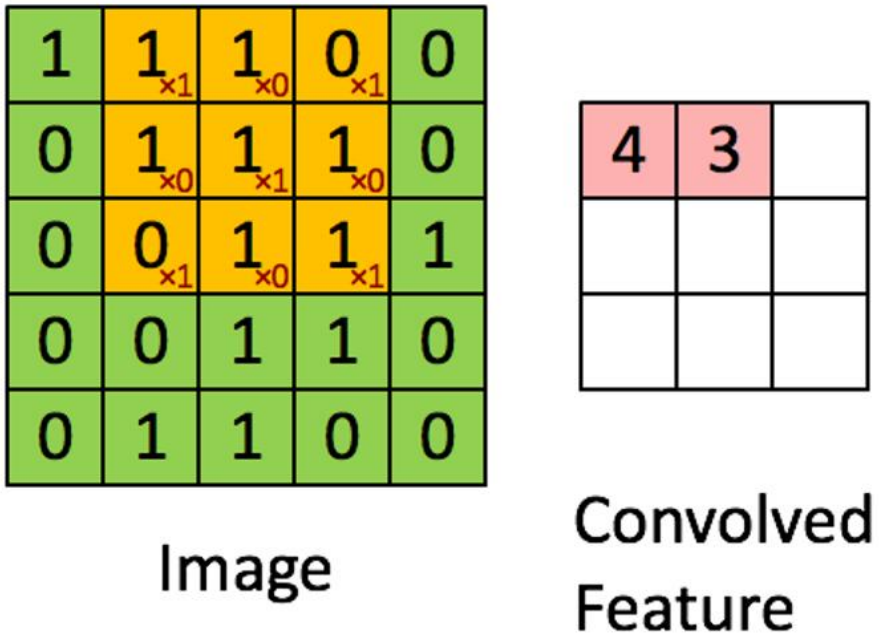
- 右图给出了H=W=5, $C_1=1$, $C_2=1$, F=3, {C}={1}, S=1, P=0情况下的例子。



CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长S在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

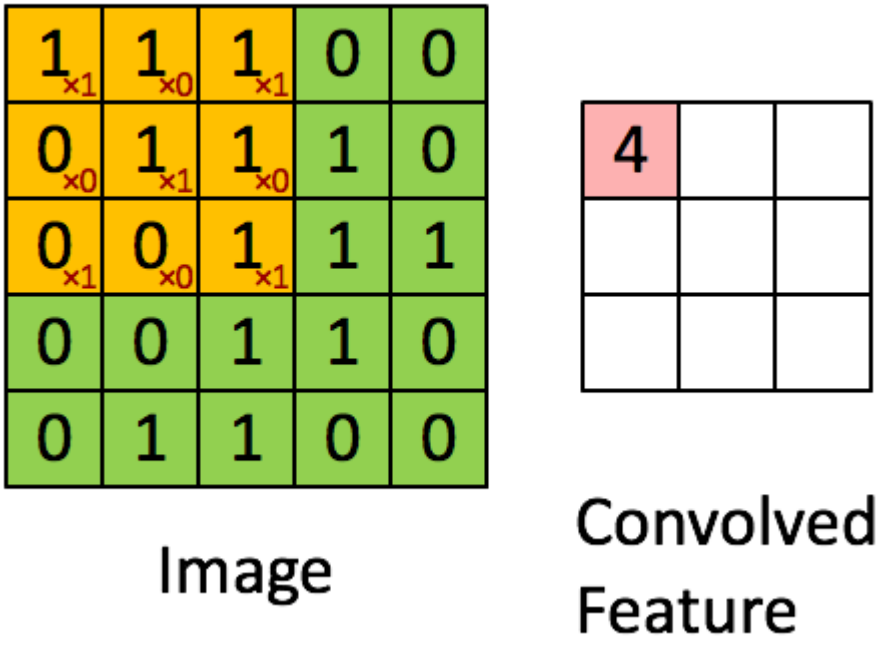
- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0情况下的例子。



CNN的结构元件：卷积层

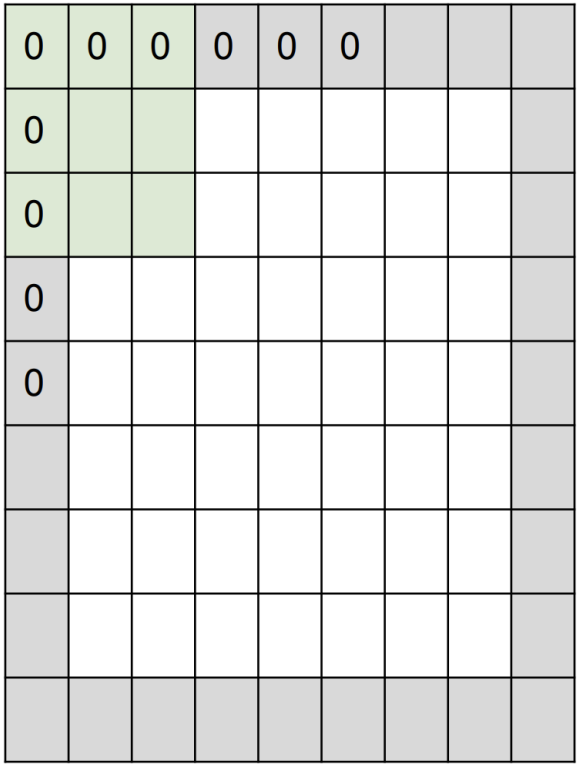
- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长S在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。



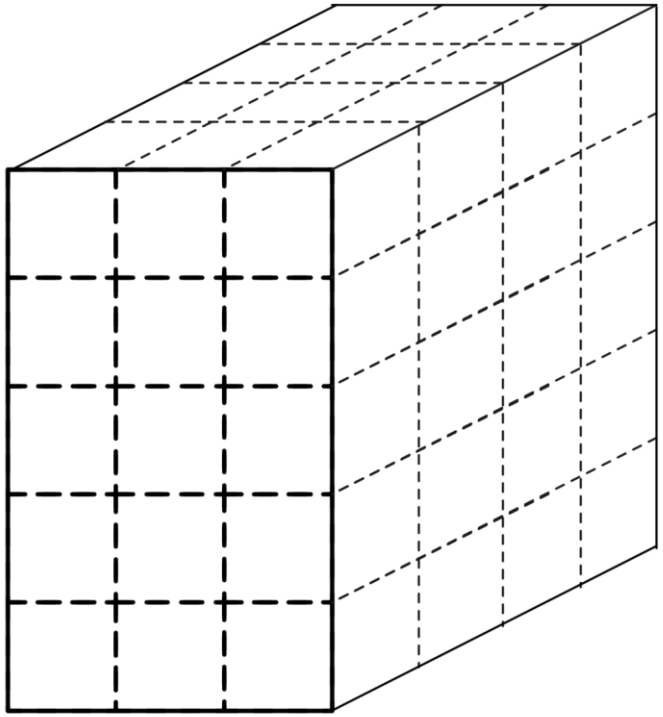
CNN的结构元件：卷积层

- 实际操作在边界填充零值（Zero Padding）
- 比如输入7*7，3*3滤波器，步长为1
- 不填充/填充1个像素边界，输出是什么？



- 5*5 和 7*7 的输出！
- 通常卷积层的步长为1，滤波器大小F*F
- 为了保持空间尺寸不变，需要填充零值（F-1）/2
- F = 3 =》填充宽度1的零值边界
- F = 5 =》填充宽度3的零值边界

Input Layer 3D Volume



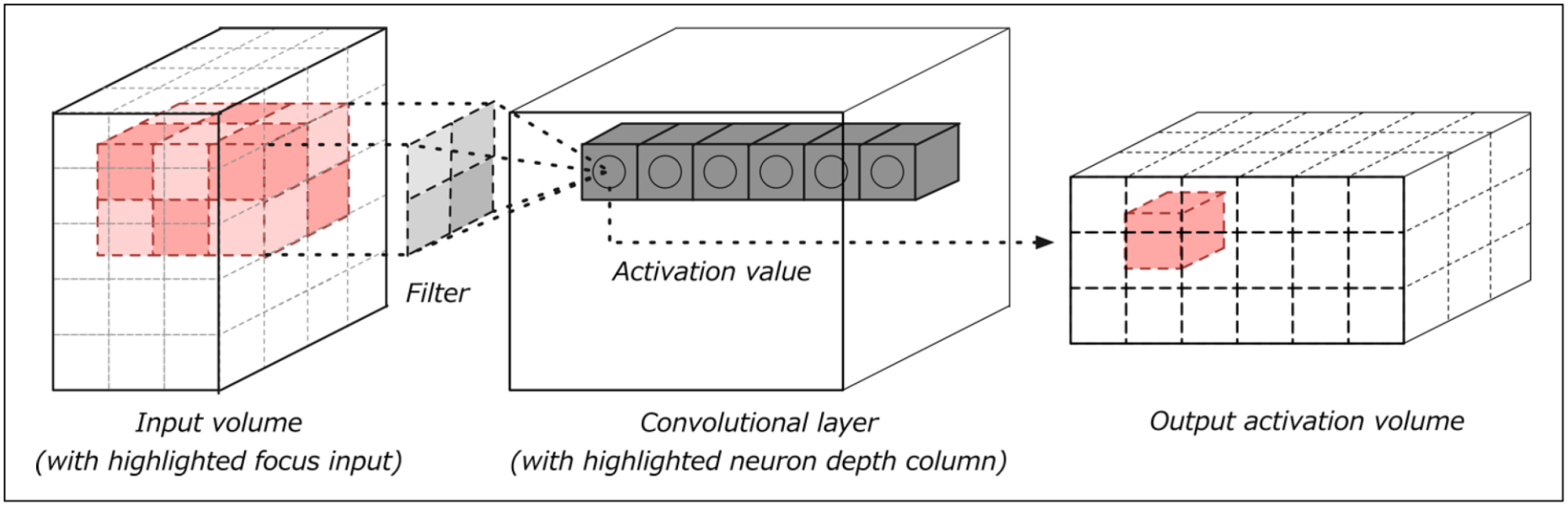
- Width
- Height
- Depth

These attributes of the input match up to an image structure for which we have:

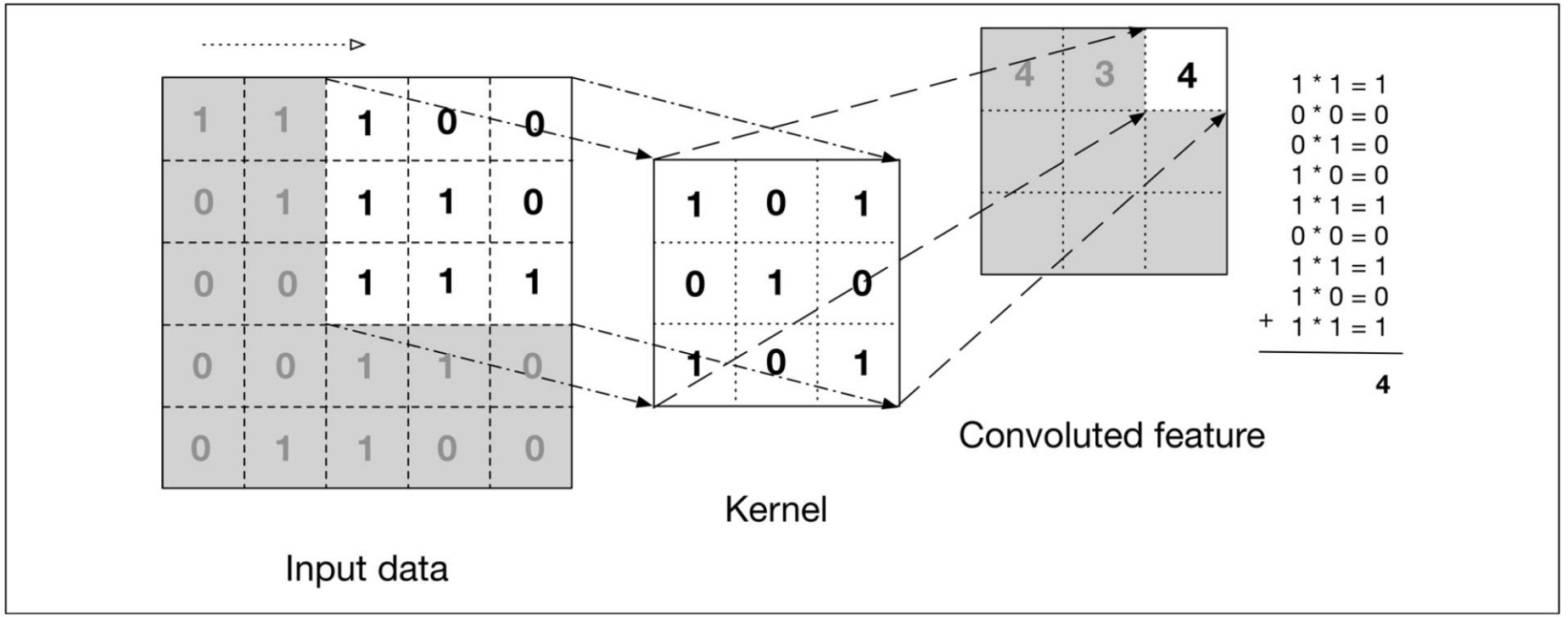
- Image width in pixels
- Image height in pixels
- RGB channels as the depth

Input layers are where we load and store the raw input data of the image for processing in the network. This input data specifies the width, height, and number of channels. Typically, the number of channels is three, for the RGB values for each pixel.

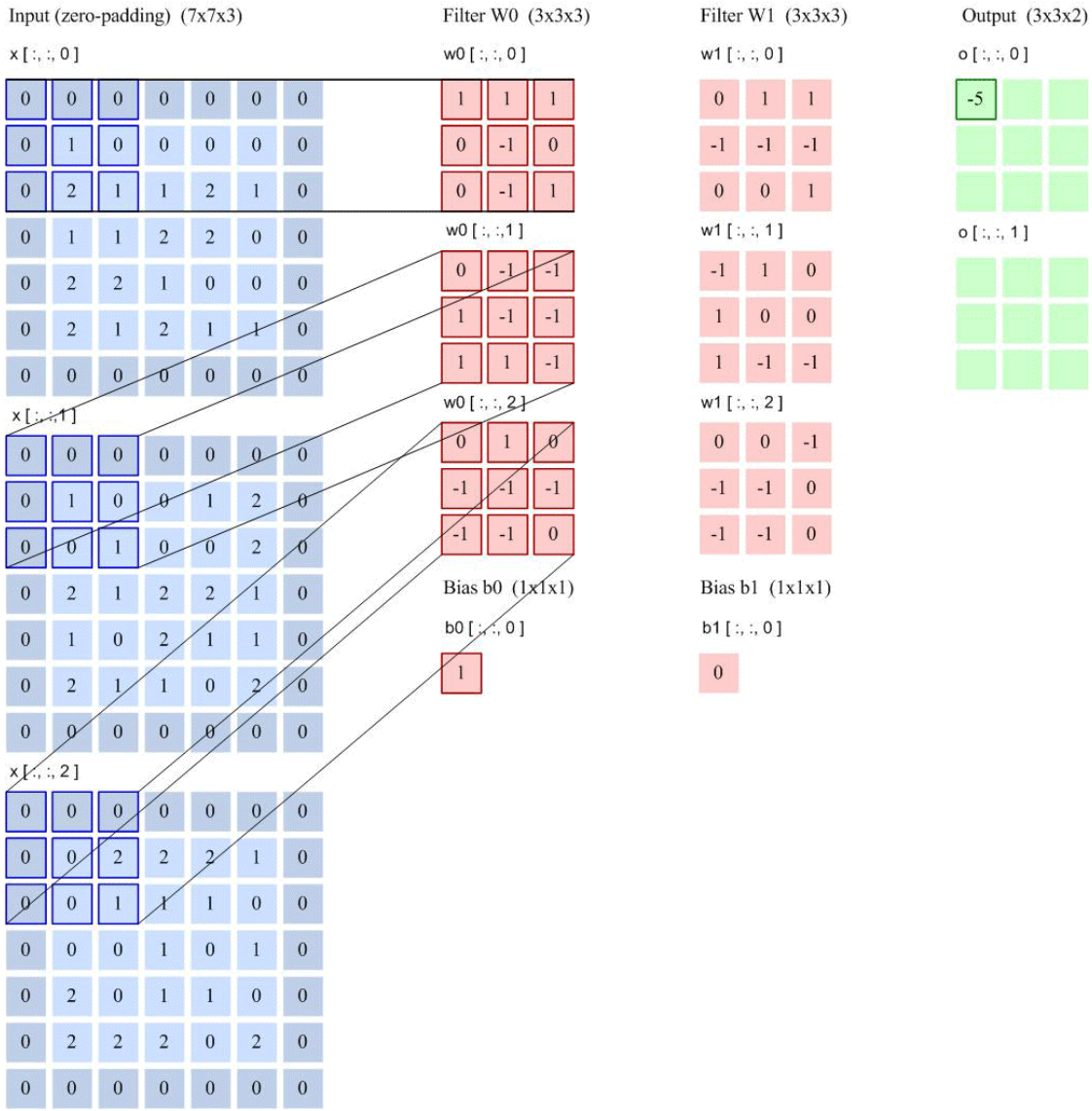
Convolution Layer with Input & Output Volumes



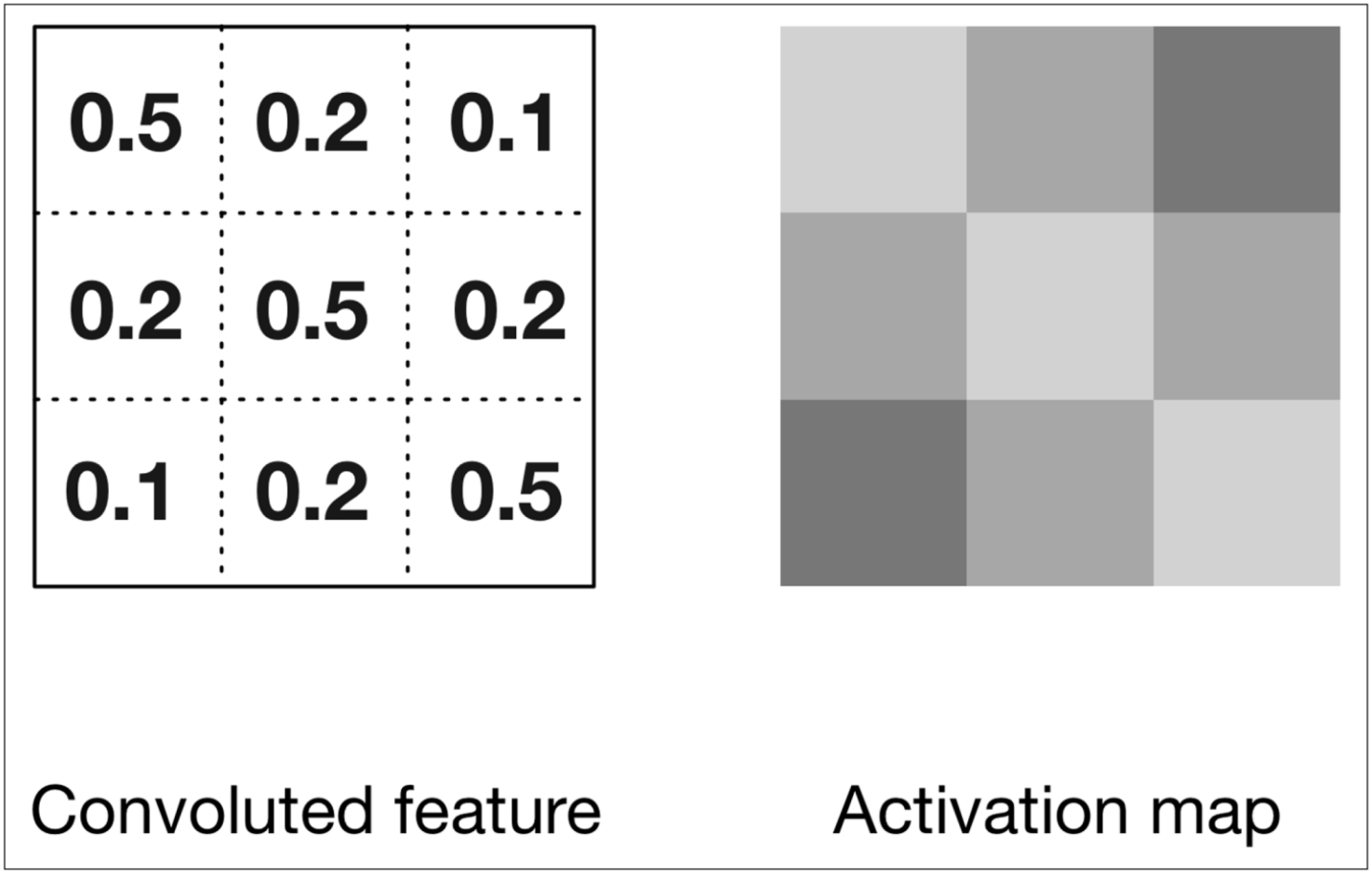
The Convolution Operation



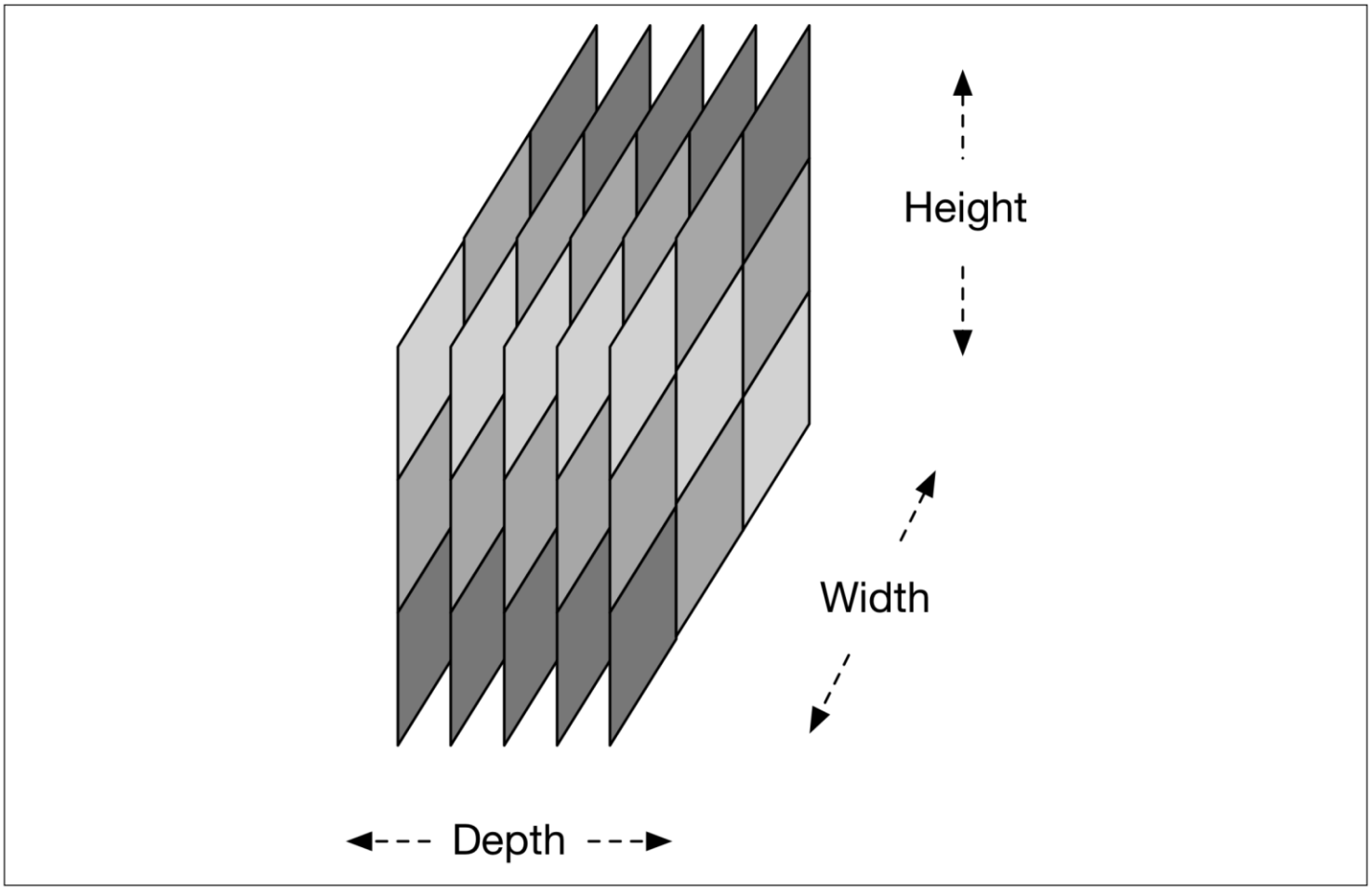
The Convolution Operation



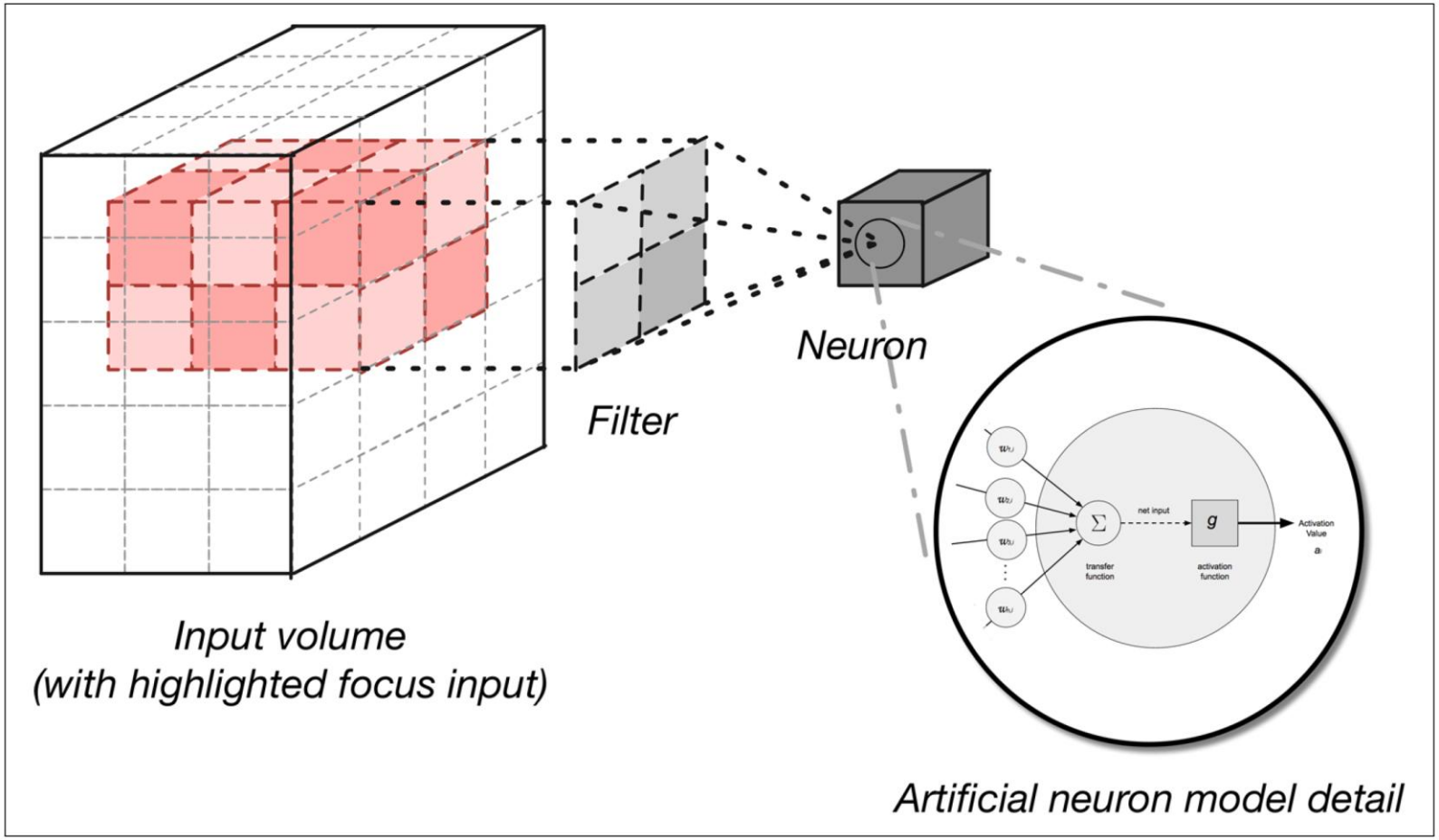
Convolution and Activation Map



Activation Volume Output of Convolutional Layer

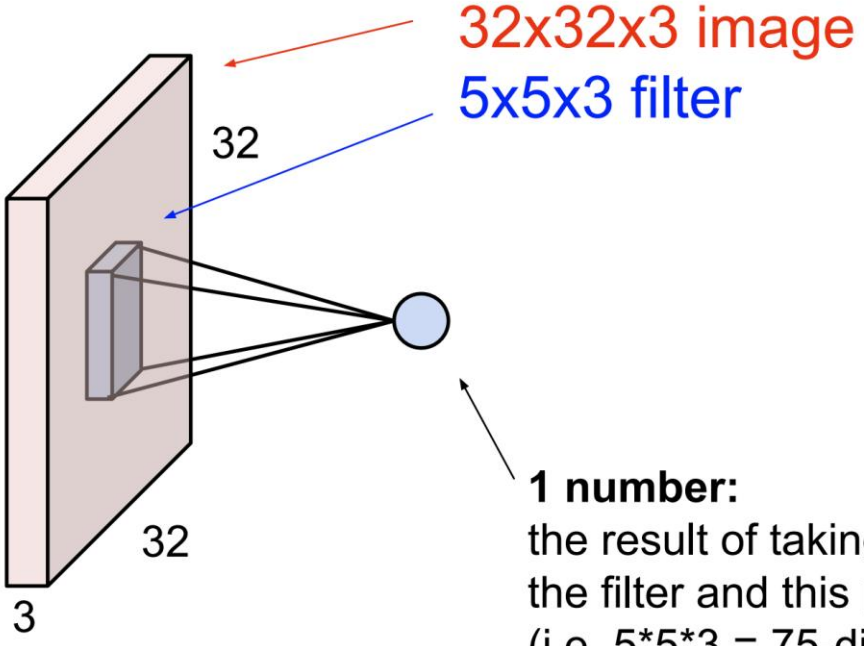


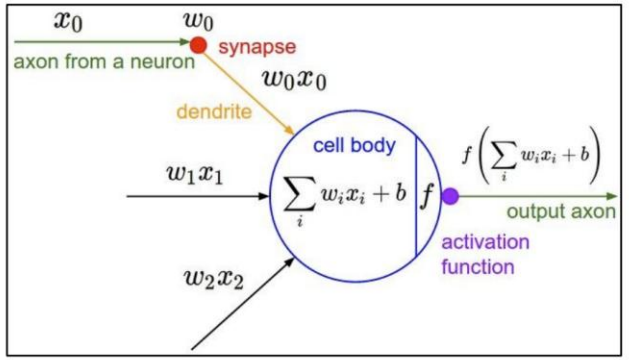
Generating an Activation Output Volume



CNN的结构元件：卷积层

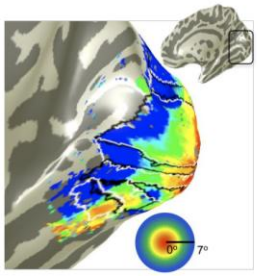
The brain/neuron view of CONV Layer





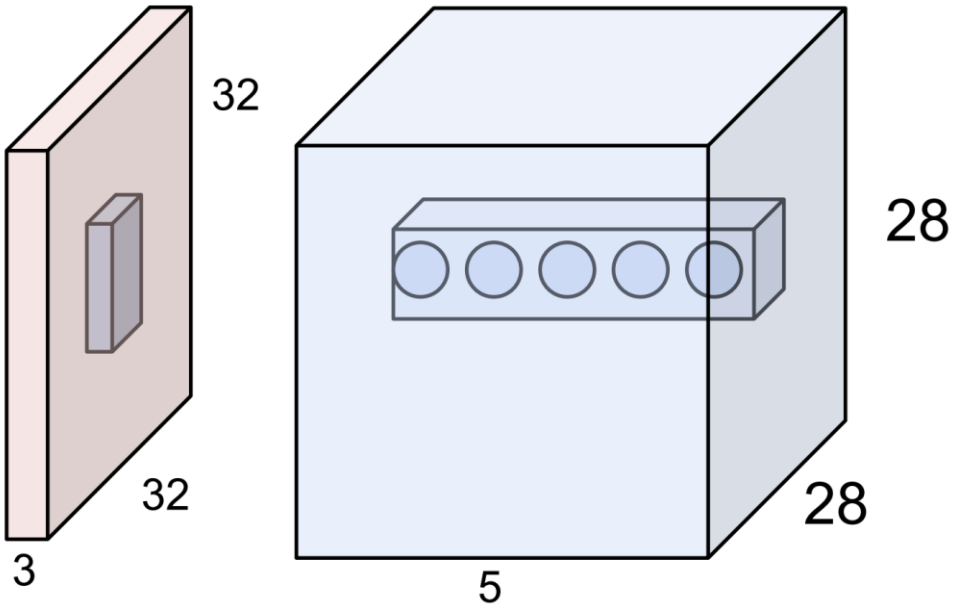
It's just a neuron with local connectivity...

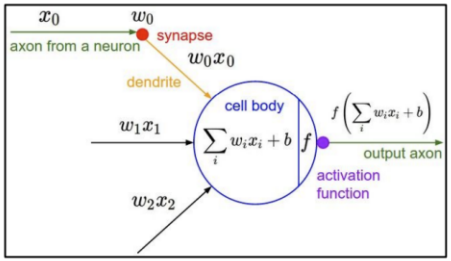
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



CNN的结构元件：卷积层

The brain/neuron view of CONV Layer





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

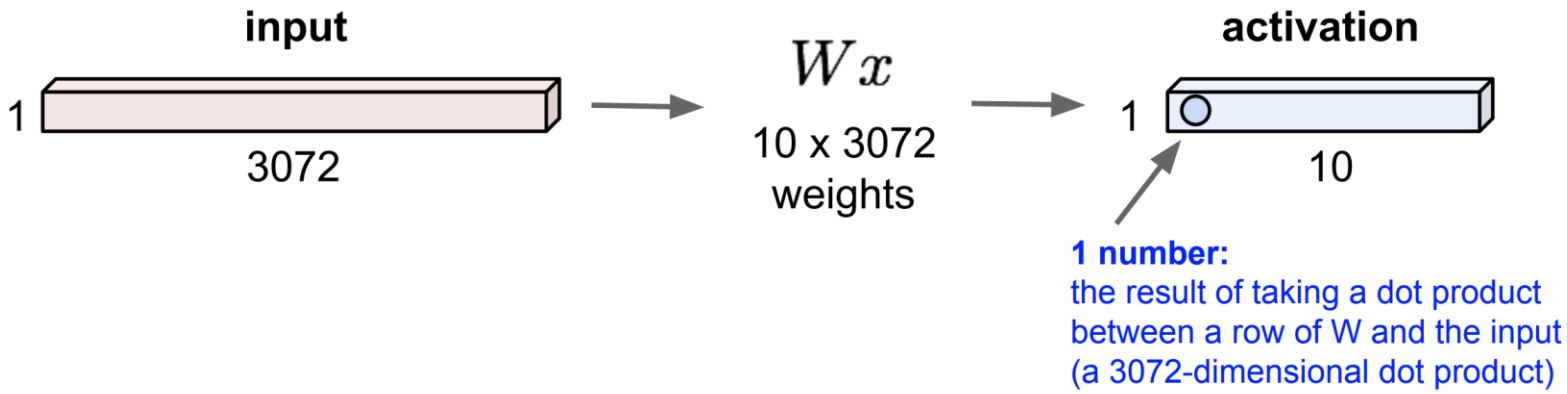
There will be 5 different neurons all looking at the same region in the input volume

CNN的结构元件：卷积层

Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume

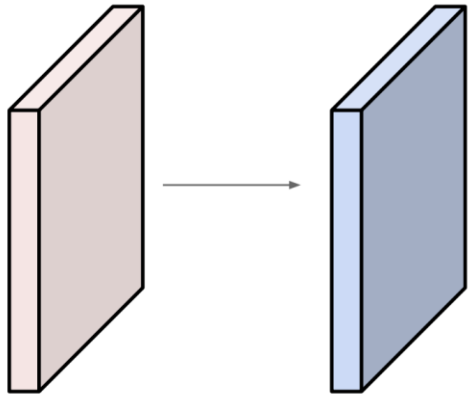


CNN的结构元件：卷积层总结

- O = convolution(I , filters)
  - O = {convolution(I , filter), for filter in filters}
  - 每个滤波器生成输出的一个通道，滤波器数=输出通道数
- 输入I：H*W*C_1，其中H,W分别是输入的高度和宽度，C_1是输入的通道数（深度）。
- 滤波器组filters：C_2*滤波器，C_2是滤波器的数目。
  - 滤波器：F是局部感受野的边长， bias是偏移量 (通常为1)，P是填充宽度，S是步长
- 输出O：H'*W'*C_2，由输入和滤波器组唯一确定。
  - H'=(H-F+2*P)/S+1，W'=(W-F2*P)/S+1

CNN的结构元件：卷积层总结

Examples time:

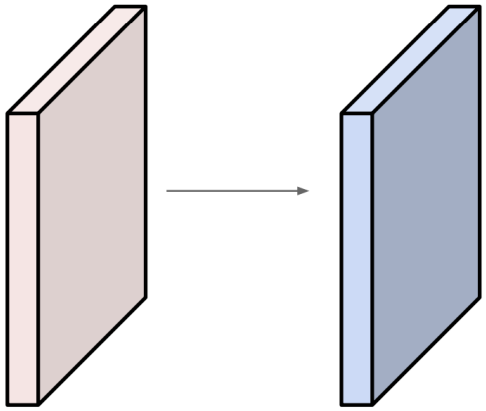


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

CNN的结构元件：卷积层总结

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

CNN卷积层PyTorch实现

Example: CONV layer in PyTorch

Conv layer needs 4 hyperparameters:

- 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

CNN的结构元件：池化的意义

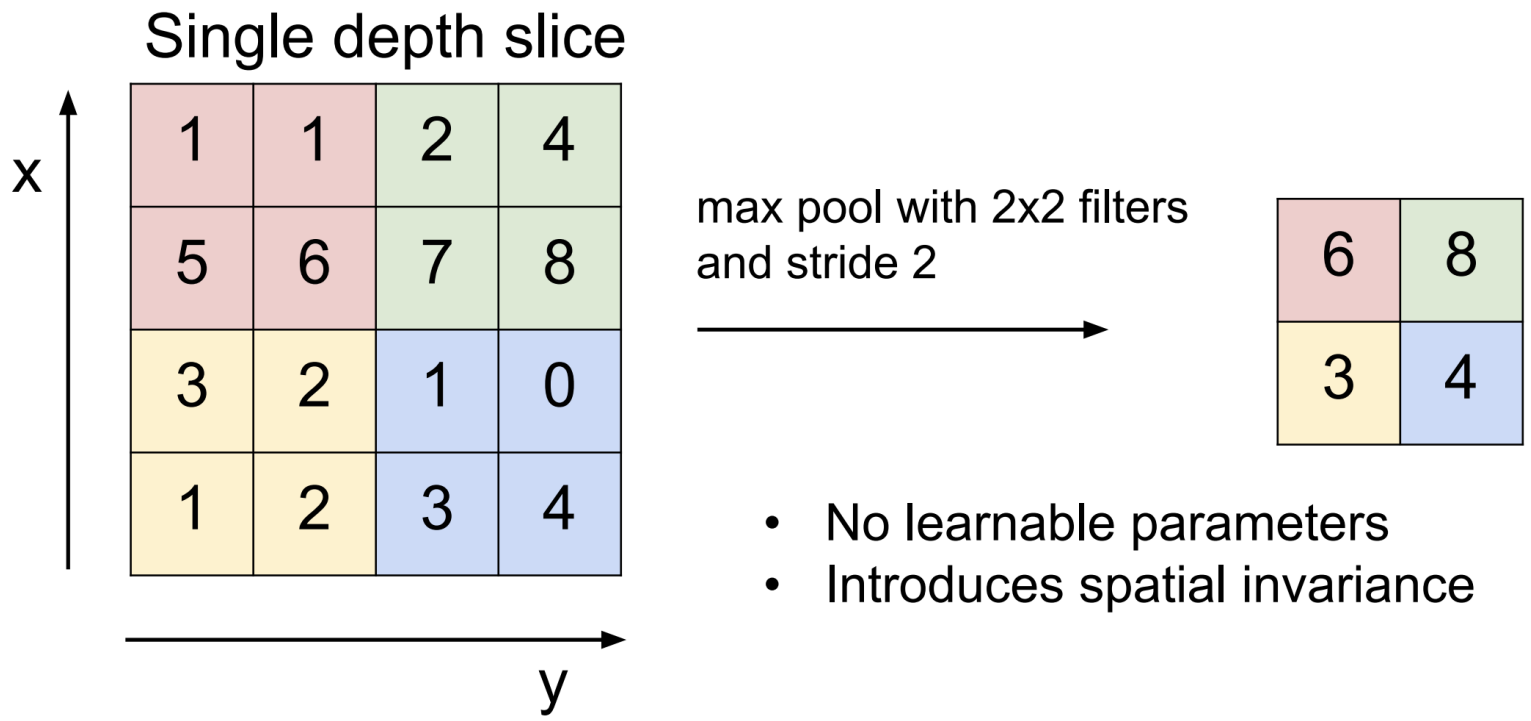
- 在卷积神经网络中，经常会进行池化操作，池化层往往在卷积层后面，通过池化来降低卷积层输出的特征向量，同时改善结果（不易出现过拟合）。

- 图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用。因此，为了描述大的图像，一个很自然的想法就是对不同位置的特征进行聚合统计。

- 例如，人们可以计算图像一个区域上的某个特定特征的平均值（或最大值）来代表这个区域的特征。这就是池化操作，通过池化我们可以降低特征的维度。

CNN的结构元件：池化层

- 池化层：F*F+S，其中F是窗口长度，S是移动步长。
- 常见的CNN都是使用最大化池化层，即输出窗口中的最大值。
- 均值池化层(输出窗口内的均值)，最近常用于替代全连接层。
- 下图是F=2,S=2最大化池化的例子：



CNN池化层PyTorch实现

MAXPOOL2D

CLASS torch.nn.MaxPool2d(*kernel_size*, *stride=None*, *padding=0*, *dilation=1*, *return_indices=False*, *ceil_mode=False*) [SOURCE]

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

In the simplest case, the output value of the layer with input size (N, C, H, W), output (N, C, H_{out}, W_{out}) and `kernel_size` (kH, kW) can be precisely described as:

$$out(N_i, C_j, h, w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} \text{input}(N_i, C_j, \text{stride}[0] \times h + m, \text{stride}[1] \times w + n)$$

If `padding` is non-zero, then the input is implicitly padded with negative infinity on both sides for `padding` number of points. `dilation` controls the spacing between the kernel points. It is harder to describe, but this link has a nice visualization of what `dilation` does.

• NOTE

When ceil_mode=True, sliding windows are allowed to go off-bounds if they start within the left padding or the input. Sliding windows that would start in the right padded region are ignored.

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

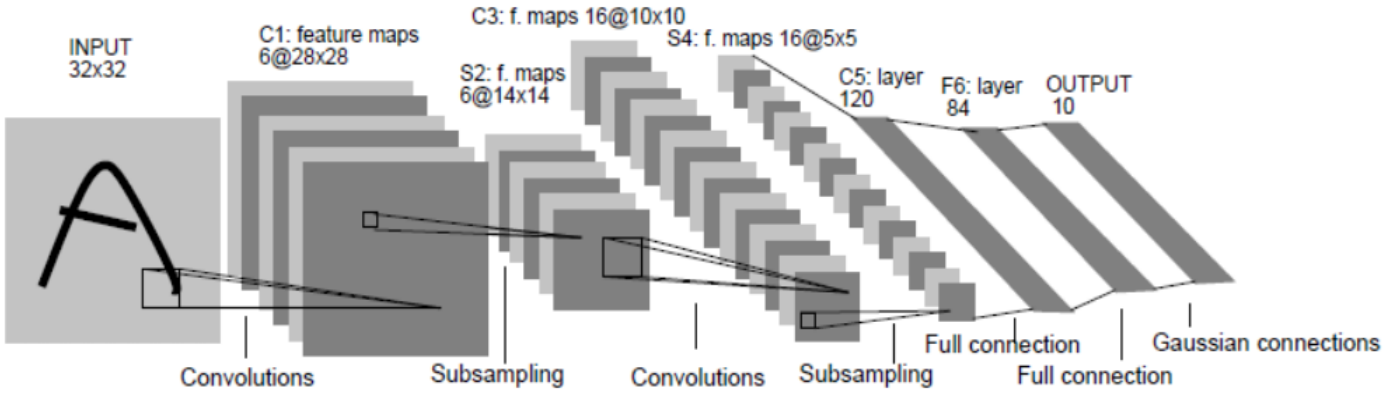
- a single `int` – in which case the same value is used for the height and width dimension
- a `tuple` of two ints – in which case, the first *int* is used for the height dimension, and the second *int* for the width dimension

CNN的结构元件：全连接层&损失函数

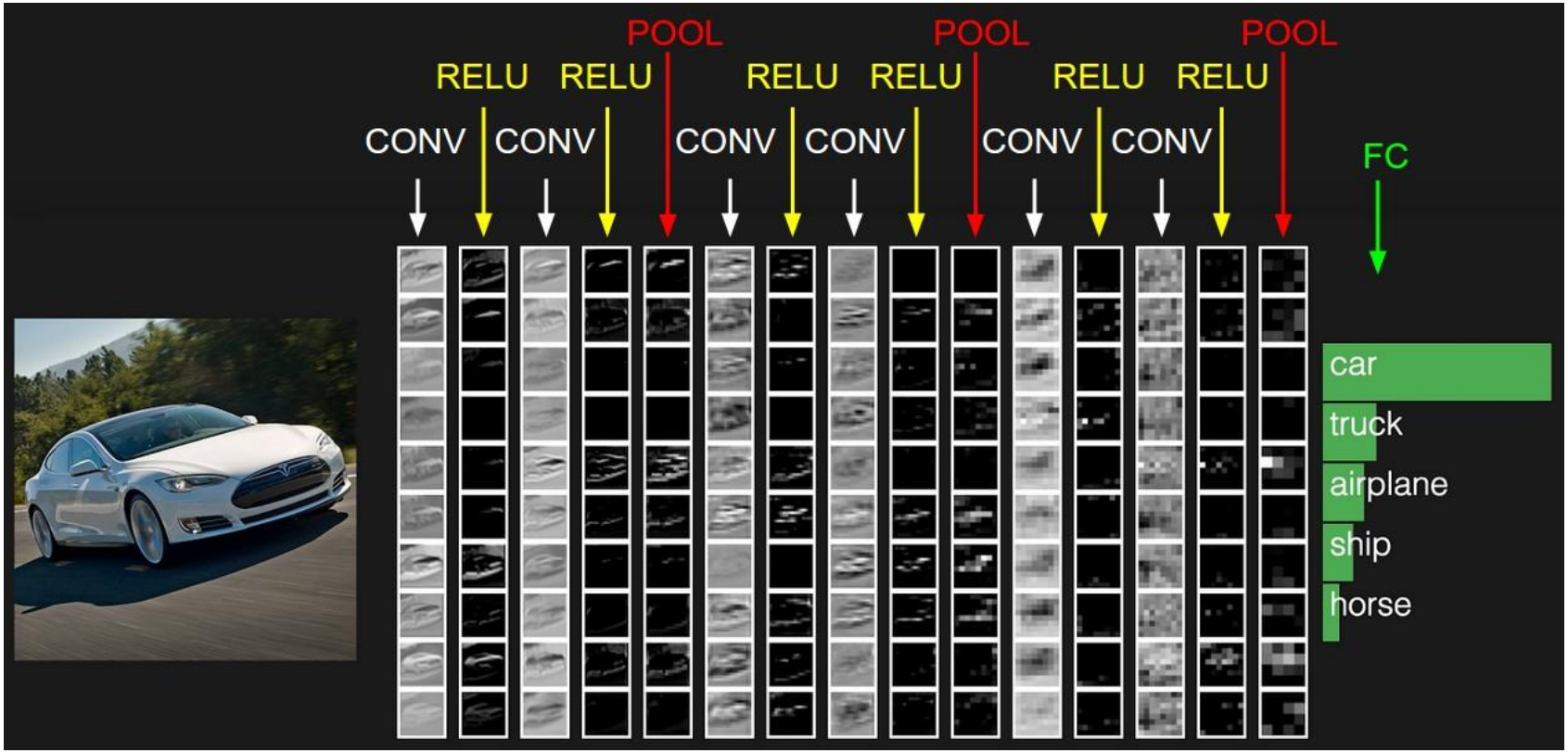
- 全连接层：常见的前向传播网络，每个神经网络节点与前一层的所有节点连接。
  - 全连接层可以看做F=H=W, P=0的卷积层。
  - 大型网络中常用均值池化层替代全连接层(减少参数个数)
- 损失函数：常见的有交叉熵，平均平方误差等。

CNN的典型结构

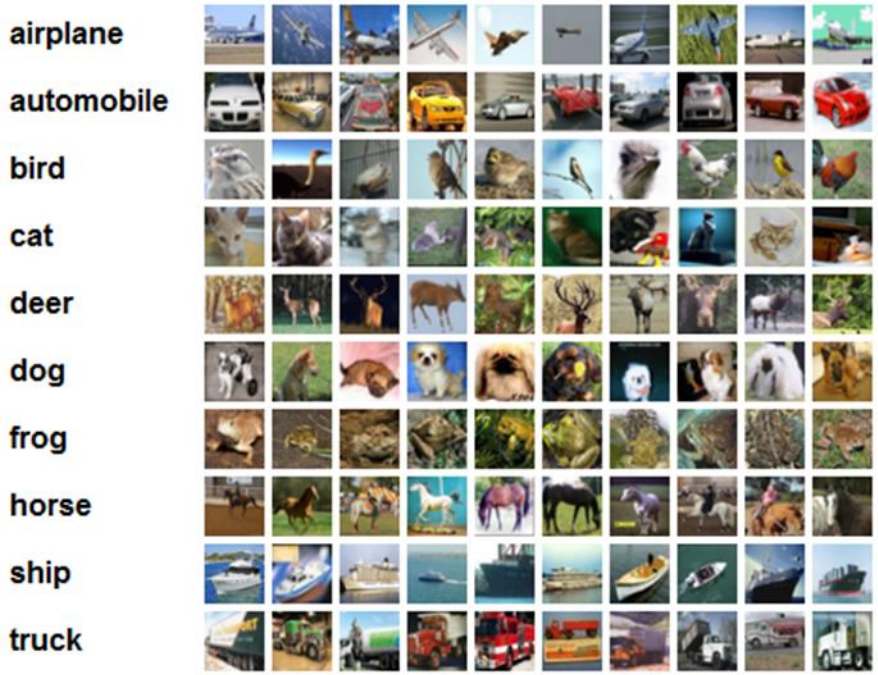
- 每层卷积层之后都接激活层
- 若干层卷积层之后接池化层
- 最后使用全连接层+损失函数
- **LeNet, AlexNet, VGGNet都符合这一范式**



CNN的典型结构


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

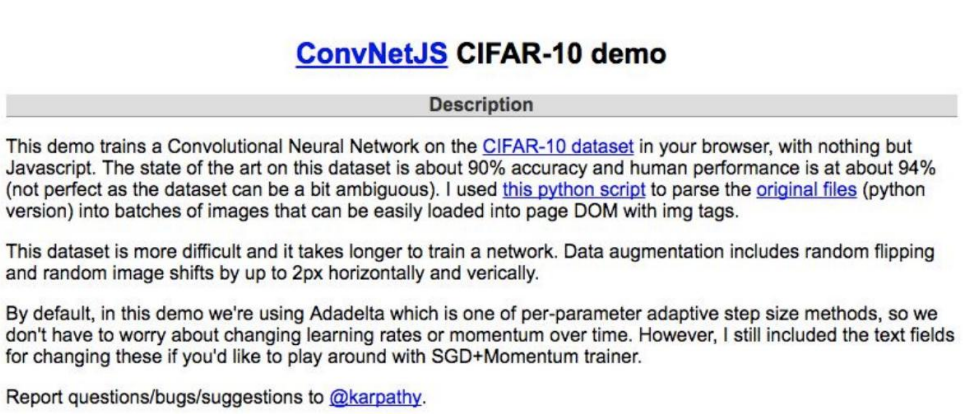
CNN应用案例

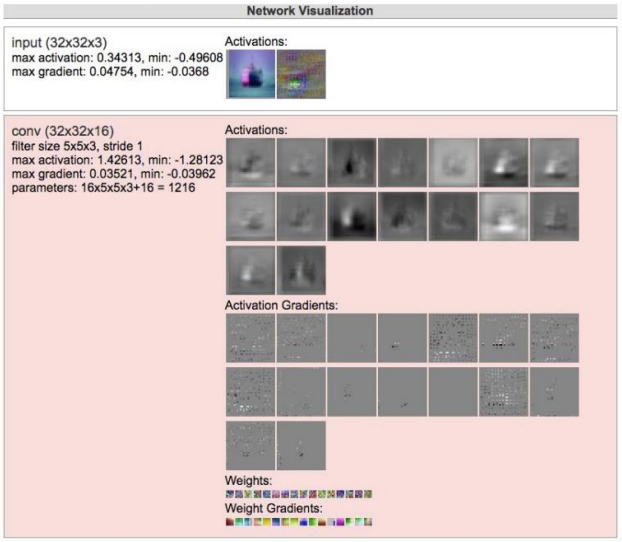


The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

CNN应用案例

[ConvNetJS demo: training on CIFAR-10]

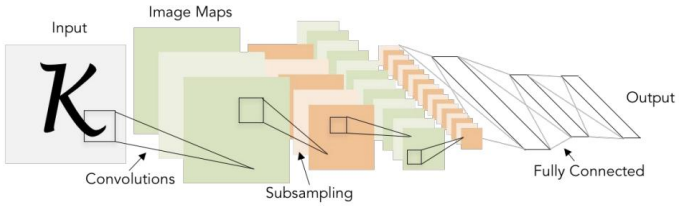


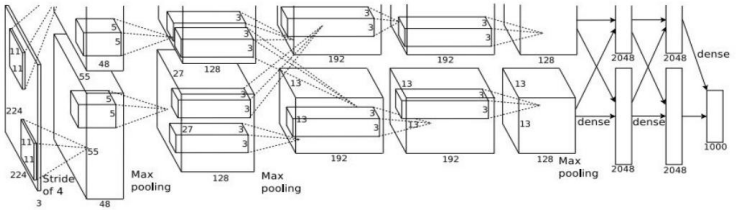


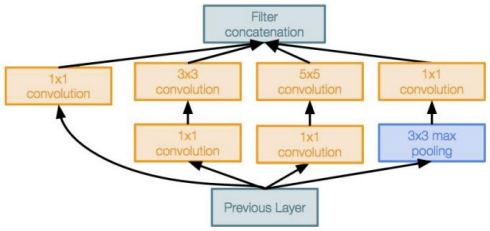
http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

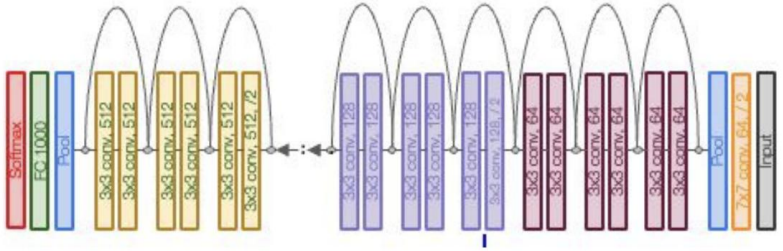
更多的CNN网络结构

- AlexNet
- VGG
- GoogLeNet
- ResNet
- DenseNet
- …









ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton,

Advances in Neural Information Processing Systems 2012

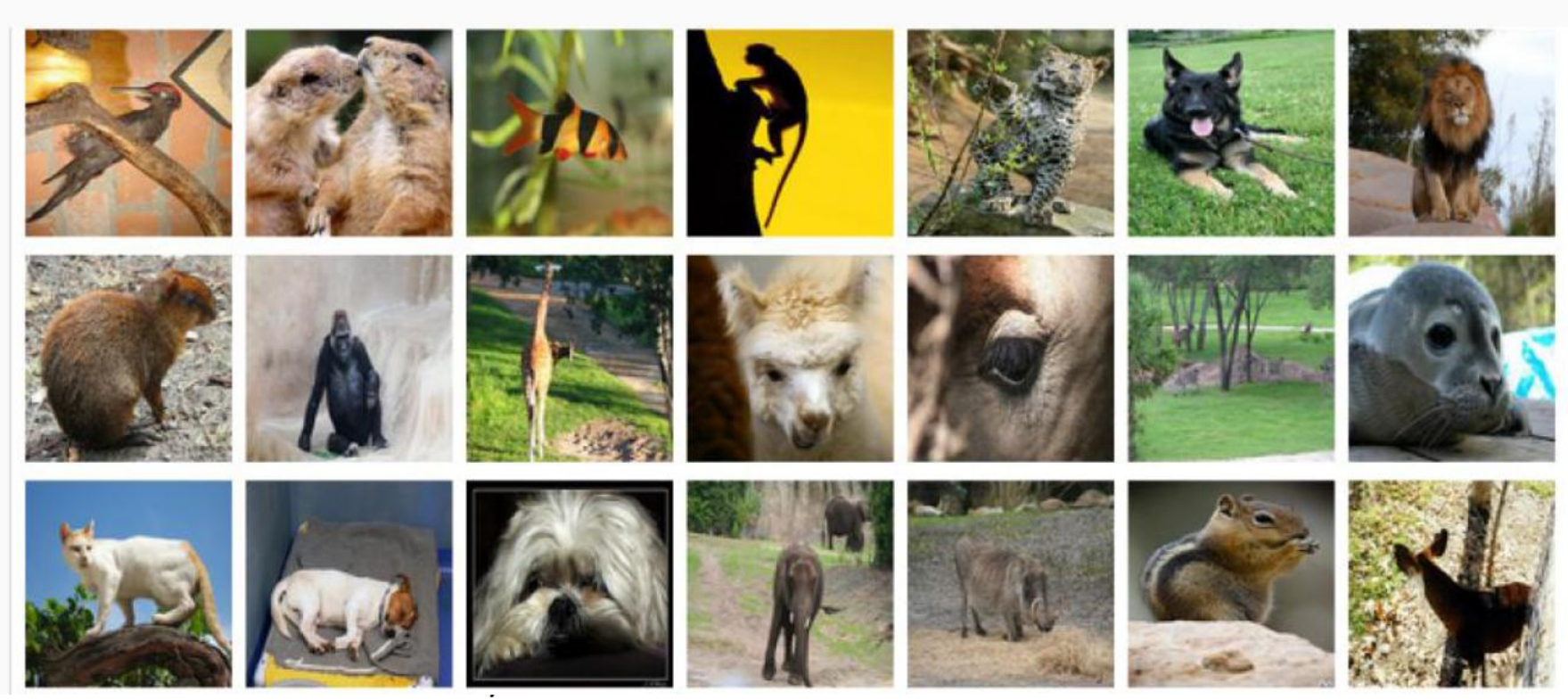
ImageNet

- 15M images
- 22K categories
- Images collected from Web
- Human labelers (Amazon's Mechanical Turk crowd-sourcing)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
  - 1K categories
  - 1.2M training images (~1000 per category)
  - 50,000 validation images
  - 150,000 testing images
- RGB images
- Variable-resolution, but this architecture scales them to 256x256 size

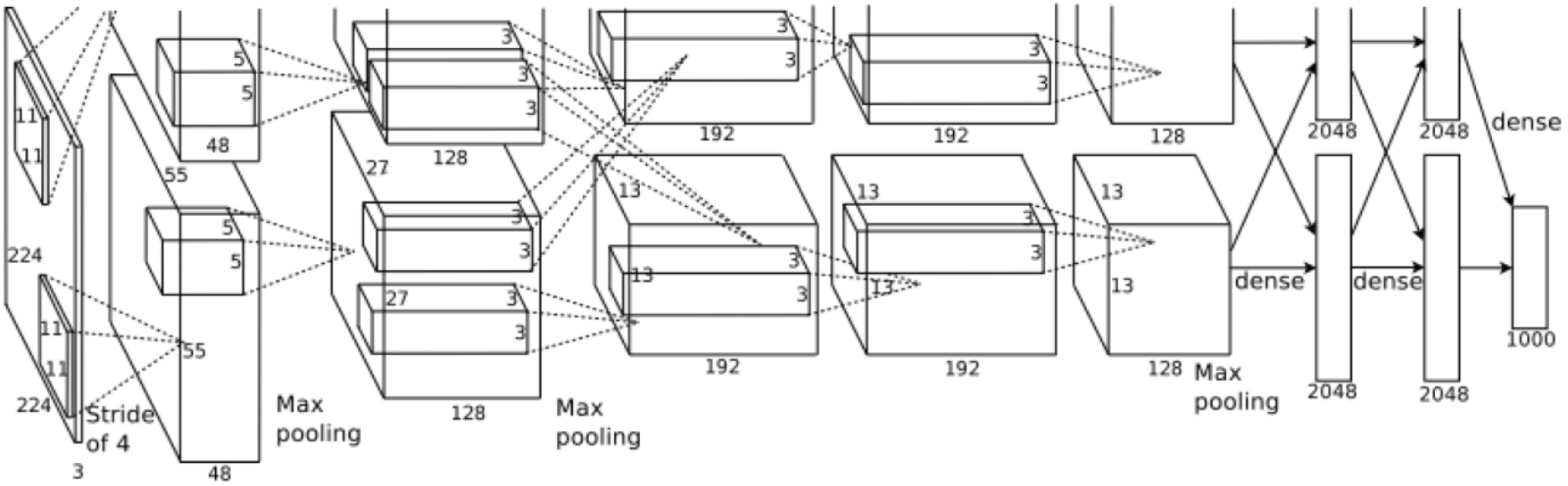
ImageNet

Classification goals:

- Make 1 guess about the label (Top-1 error)
- make 5 guesses about the label (Top-5 error)



The CNN Architecture



The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in the kernel map. 224/4=56

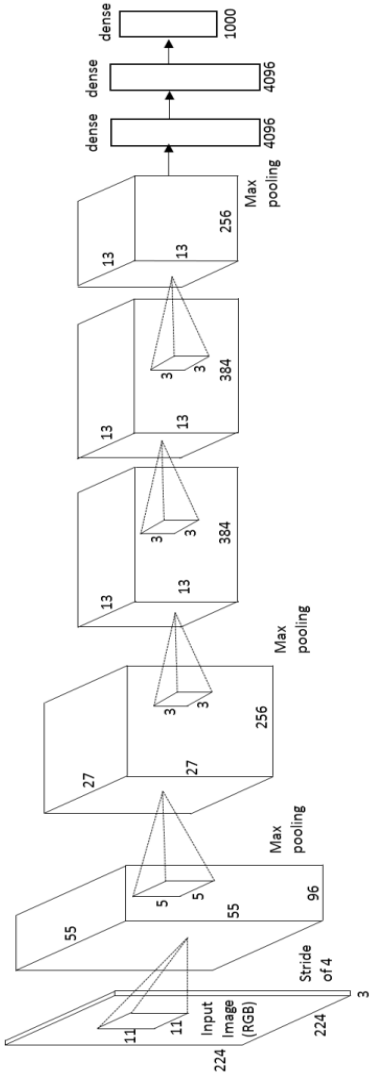
The pooling layer: form of non-linear down-sampling. Max-pooling partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum value

Experimental Setting

- Trained with stochastic gradient descent
- on two NVIDIA GTX 580 3GB GPUs
- for about a week

- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 5 convolutional layer, 3 fully connected layer
- Final feature layer: 4096-dimensional

AlexNet

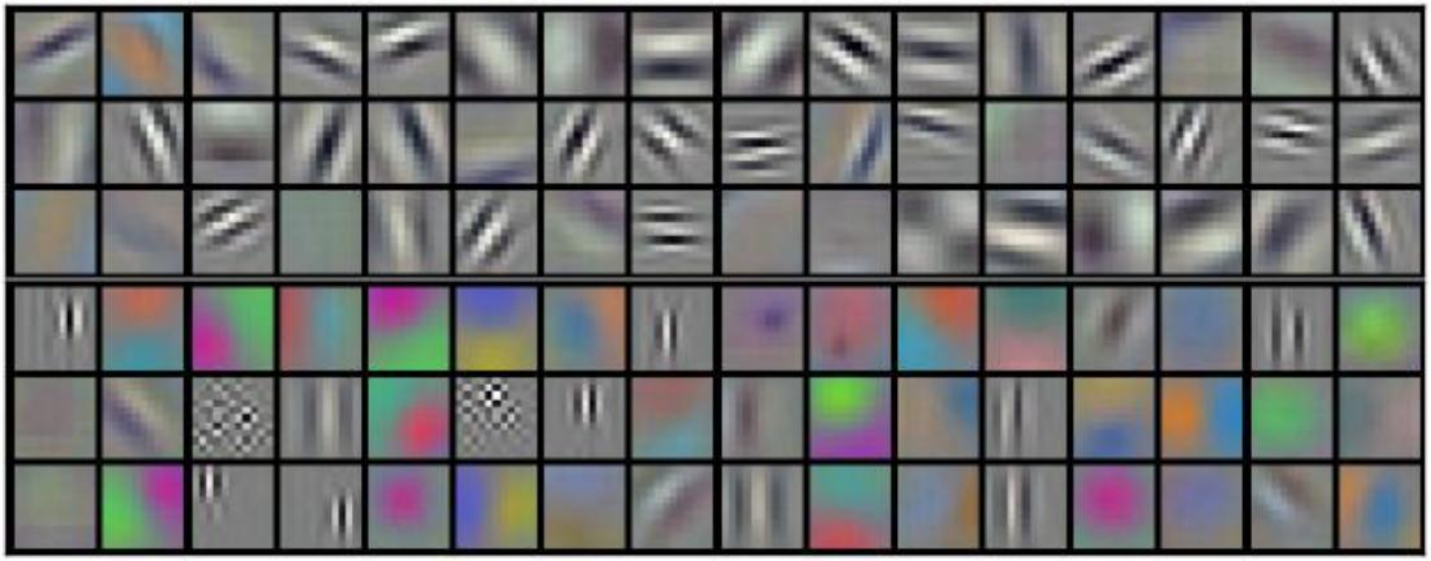



```python
class AlexNet(nn.Module):

    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), 256 * 6 * 6)
        x = self.classifier(x)
        return x
```

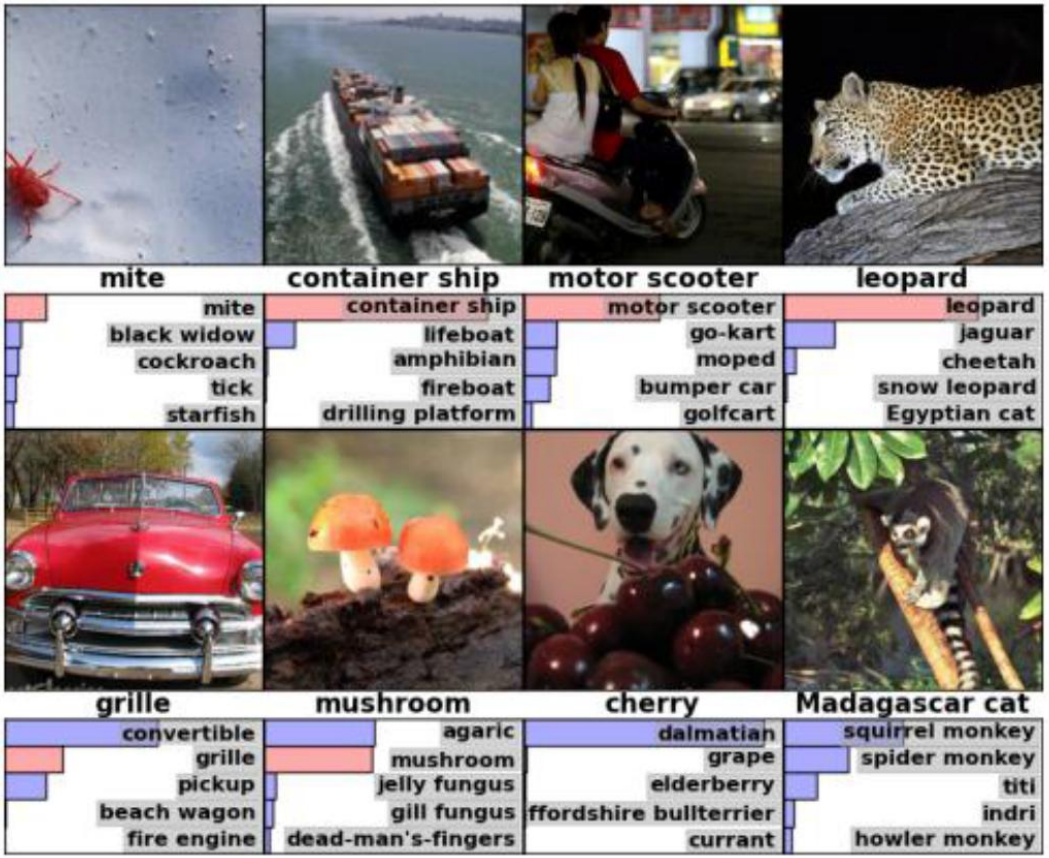
The First Convolutional Layer



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images.

The top 48 kernels were learned on GPU1 while the bottom 48 kernels were learned on GPU2

Experimental Results



Results on the test data:

top-1 error rate: 37.5%

top-5 error rate: 17.0%

CNN VS 全连接网络

- 局部感受野
  - 保留图像的拓扑结构
  - 提取图像的基础特征
- 权值共享
  - 减少可训练参数→防止过拟合
  - 基础特征适用于图像的所有位置→平移、扭曲不变性
- 降采样
  - 提取层次化特征
  - 减少具体位置对于最终结果的影响→平移、扭曲不变性

谢谢！ 下周见！