uml.org.cn 文章 文库 视频 讲座 课程 咨询 认证

加火龙果微信

看讲座实录

火龙果讲堂
- 一线专家
- 案例回顾
- 经验分享

# 深度学习框架TensorFlow

钱兴会

学习为工作创造价值

# 机器学习



- 机器学习（Machine Learning）是一门专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构市值不断改善自身的性能的学科，简单地说，机器学习就是通过算法，使得机器能从大量的历史数据中学习规律，从而对新的样本做智能识别或预测未来。
- 机器学习在图像识别、语音识别、自然语言理解、天气预测、基因表达、内容推荐等很多方面的发展还存在着没有良好解决的问题。

# 特征的自学习

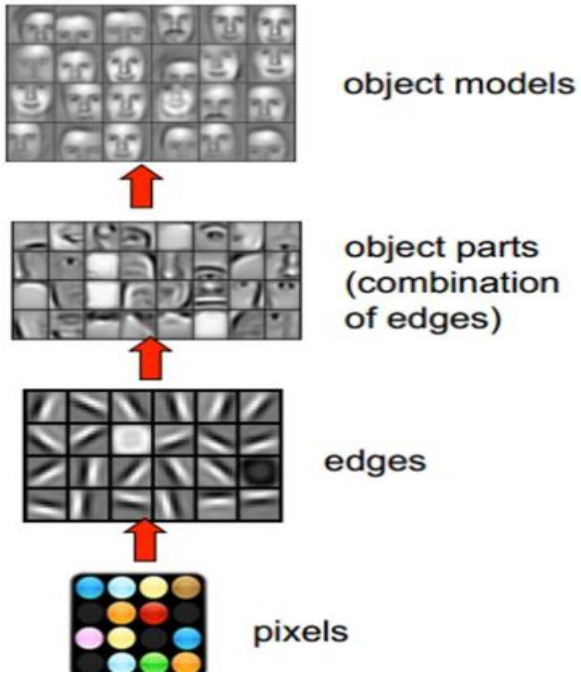


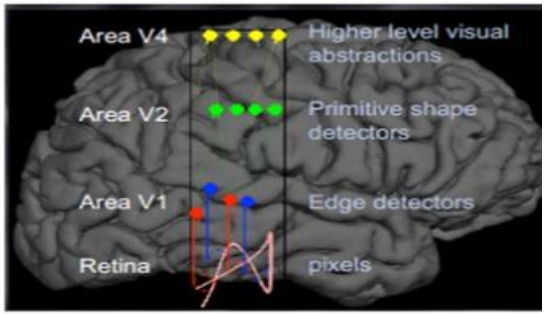
传统的模式识别方法:

通过传感器获取数据，然后经过预处理、特征提取、特征选择、再到推理、预测或识别。

特征提取与选择的好坏对最终算法的确定性齐了非常关键的作用。而特征的样式目前一般都是靠人工提取特征。而手工选取特征费时费力，需要专业知识，很大程度上靠经验和运气，那么机器能不能自动的学习特征呢？深度学习的出现就这个问题提出了一种解决方案。

由此可知人的视觉系统的信息处理是分级的，高层的特征是低层特征的组合，从低层到高层的特征表示越来越抽象，越来越能表现语义或者意图，抽象层面越高，存在的可能猜测就越少，就越利于分类。



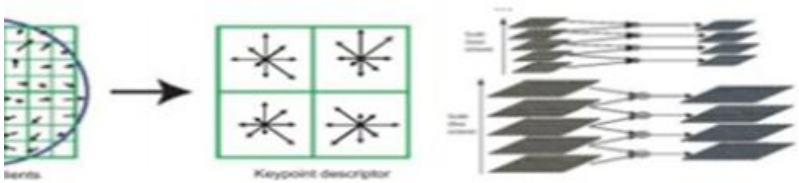
# 深度学习



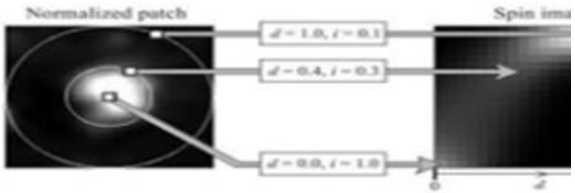
- 自2006年，深度学习（Deep Learning）已经成为机器学习研究中的一个新兴领域，通常也被叫做深层结构学习或分层学习。其动机在于建立、模拟人脑进行分析学习的神经网络，它模拟人脑的机制来解释数据，例如图像，声音和文本，深度学习是无监督学习的一种。
- 深度学习的概念源于人工神经网络的研究，含多隐层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，已发现数据的分布式特征表示。

# 基本概念

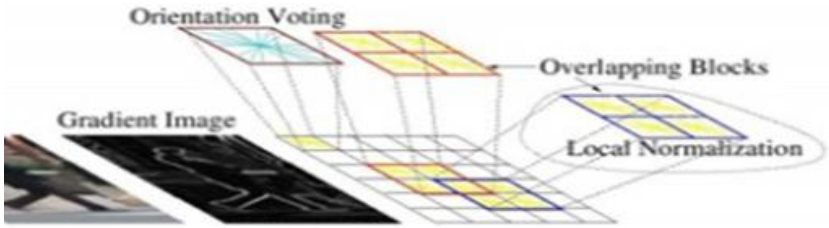




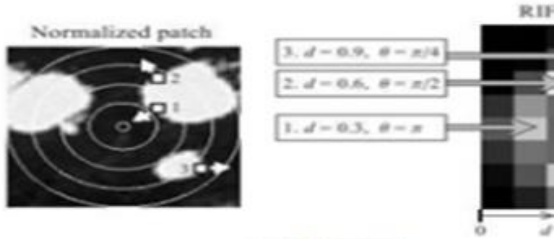

SIFT



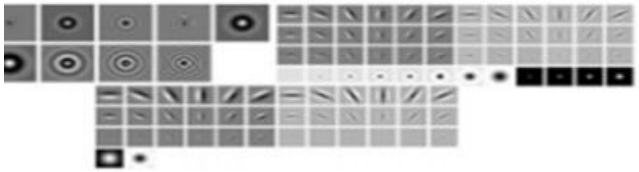

Spin image



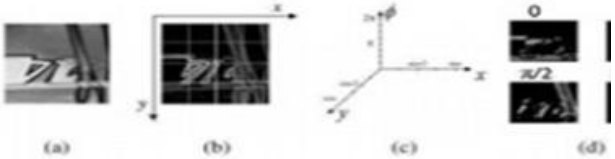

HoG




RIFT



Textons




GLOH

# 深度学习实质

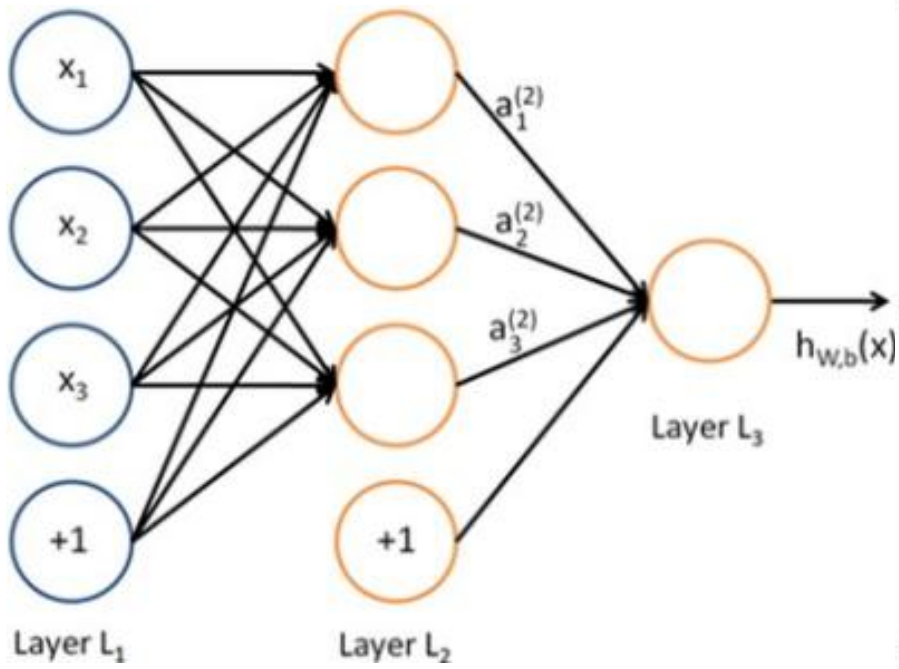


- 深度学习的实质
  - 通过构建具有很多隐层的机器学习模型和海量的训练数据，来学习更有用的特征，从而最终提升分类或预测的准确性。因此，“深度模型”是手段，“特征学习”是目的。
- 深度学习与浅层学习的区别
  - 强调了模型结构的深度，通常有5-10多层的隐层节点;
  - 明确突出了特征学习的重要性，通过逐层特征变换，将样本在原空间的特征表示变换到一个新特征空间，从而使分类或预测更加容易。与人工规则构造特征的方法相比，利用大数据来学习特征，更能够刻画数据的丰富内在信息。

# 深度学习与神经网络的异同



## 神经网络




## 深度学习

含多个隐层的深度学习模型

# 深度学习与神经网络的异同



- 相同点

  二者均采用分层结构，系统包括输入层、隐层（多层）、输出层组成的多层网络，只有相邻层节点之间有连接，同一层以及跨层节点之间相互无连接，每一层可以看作是一个logistic 回归模型。

- 不同点：采用不同的训练机制

  –神经网络：采用BP算法调整参数，即采用迭代算法来训练整个网络。随机设定初值，计算当前网络的输出，然后根据当前输出和样本真实标签之间的差去改变前面各层的参数，直到收敛；

  –深度学习：BP算法不适合深度神经网络，如果对所有层同时训练，时间复杂度会太高，如果每次训练一层，偏差逐层传递会出现过拟合。因此深度学习整体上是是一个分层训练机制。

# 深度学习的训练过程



- 自下而上的非监督学习：从底层开始，一层一层的往顶层训练，分别得到各层参数。

  采用无标签数据分层训练各层参数（可以看作是特征学习的过程）。

- 自上而下的监督学习

  基于第一步的得到的各层参数进一步调整整个多层模型的参数，这一步是一个有监督的训练过程。

# 深度学习的几种常用模型



- Auto Encoder（自动编码器）
- Sparse Coding （稀疏编码）
- Restricted Boltzmann Machine（限制玻尔兹曼机）
- Deep Belief Networks （深度信任网络）
- Convolutional Neural Networks （卷积神经网络）

# Convolutional Neural Networks(CNN)

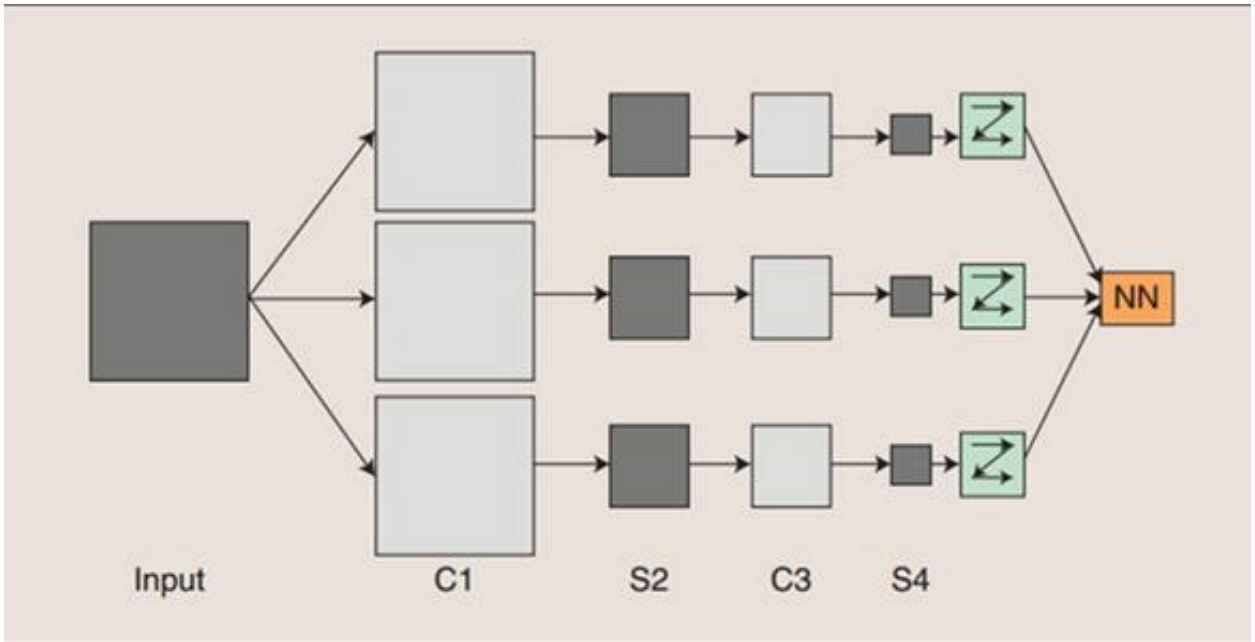


卷积神经网络是人工神经网络的一种，已成为当前语音分析和图像识别领域的研究热点。它的权值共享网络结构使之更类似于生物神经网络，降低了网络模型的复杂度，减少了权值的数量。该优点在网络的输入是多维图像时表现的更为明显，使图像可以直接作为网络的输入，避免了传统识别算法中复杂的特征提取和数据重建过程。卷积网络是为识别二维形状而特殊设计的一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者共他形式的变形具有高度不变性。

# 卷积神经网络原理图



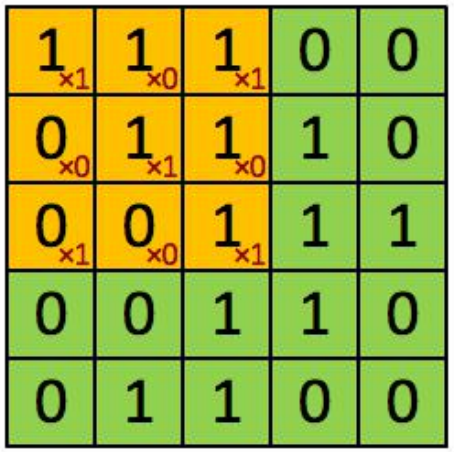
如图所示，输入图像（Input）通过和三个可训练的卷积核和可加偏置进行卷积，卷积后在C1层产生三个特征映射图（Feature map）然后，C1层的Feature map在经过子采样（Subsampling）后，加权值，加偏置，再通过一个Sigmoid函数得到三个S2层的特征映射图。



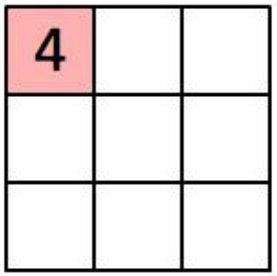
# CNN的Convolution过程



如图，原图像是5*5大小，有25个神经元，用一个3*3的卷积核对它进行卷积，得到了如右图所示的卷积后的Feature map。该特征图大小为3*3。




Image




Convolved Feature

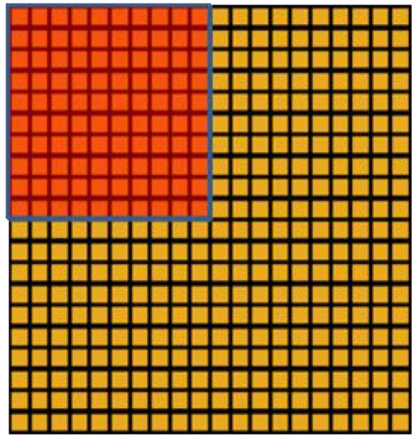
假设一种卷积核只提取出图像的一种特征，所以一般要多个卷积核来提取不同的特征，所以每一层一般都会有多张Feature map。

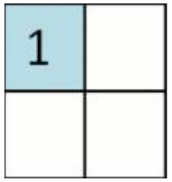
同一张Feature map上的神经元共用一个卷积核，这大大减少了网络参数的个数。

# CNN的Pooling过程



图像具有一种“静态性（stationarity）”的属性，可以对图像某一个区域上的特征取平均值 (或最大值)。这种聚合的操作就叫做池化 (pooling)。






Convolved feature

Pooled feature

如果人们选择图像中的连续范围作为池化区域，并且只是池化相同(重复)的隐藏单元产生的特征，那么，这些池化单元就具有平移不变性 (translation invariant)。这就意味着即使图像经历了一个小的平移之后，依然会产生相同的 (池化的) 特征。

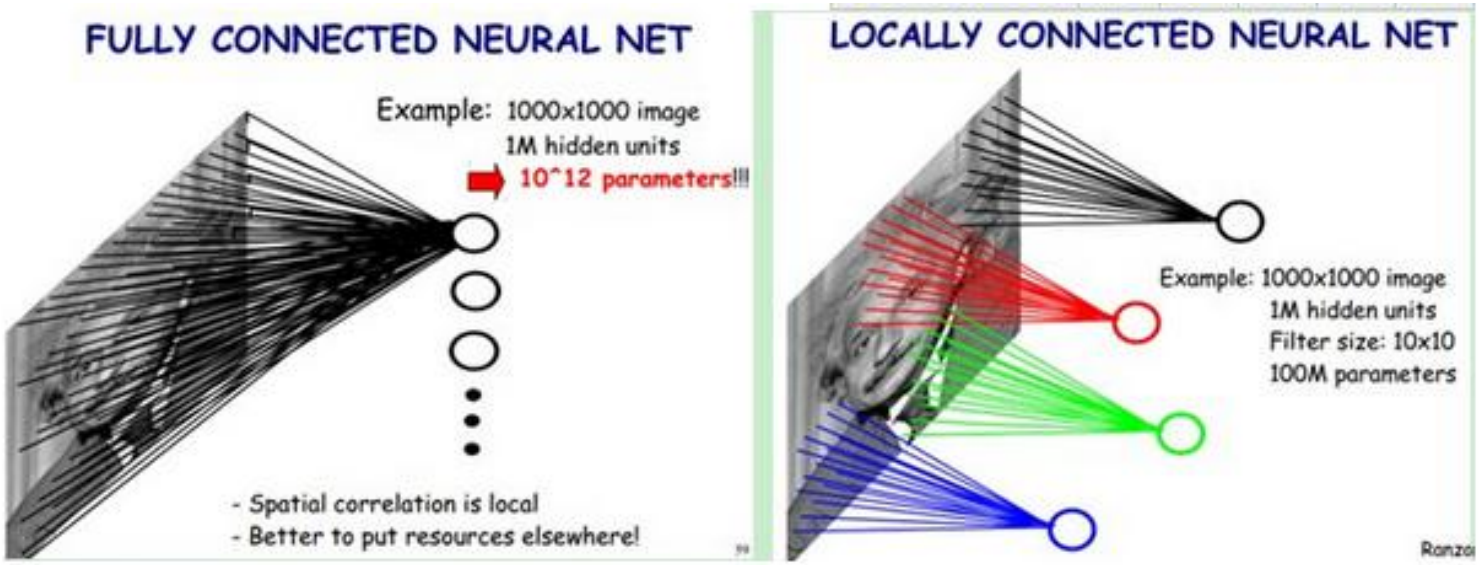
# CNN的优点



**参数减少与权值共享**

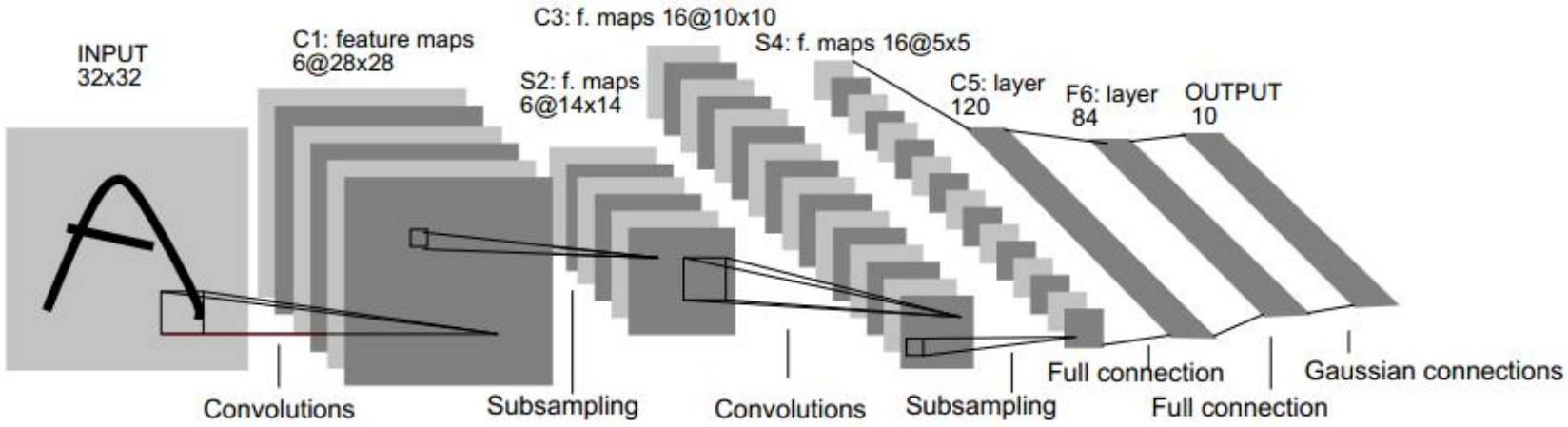
如下图所示，如果我们有1000x1000像素的图像，有1百万个隐层神经元，那么他们全连接的话（每个隐层神经元都连接图像的每一个像素点），就有个连接，也就是1000x1000x1000个权值参数。

局部连接网络，每一个节点与上层节点同位置附近10x10的窗口相连接，则1百万个隐层神经元就只有 $10^6 \times 10^8$，即10^8个参数。其权值连接个数比原来减少了四个数量级。



# 经典例子：文字识别系统LeNet-5






1. 输入图像是32x32的大小,卷积核的大小是5x5的，则C1层的大小是28x28。这里设定有6个不同的C1层，每一个C1层内的权值是相同的。

2. S2层是一个下采样层，由4个点下采样为1个点，也就是4个数的加权平均，加权系数也需要通过学习得到。这个过程也叫做Pool。

3.我们很容易得到C3层的大小为10x10，不过，C3层有16个10x10网络! 我们只需要按照一定的规则来组合S2的特征图。具体的组合规则在 LeNet-5 系统中给出了下面的表格：

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | X | | | | X | X | X | | | X | X | X | X | | X | X |
| 1 | X | X | | | | X | X | X | | | X | X | X | X | | X |
| 2 | X | X | X | | | | X | X | X | | | X | | X | X | X |
| 3 | | X | X | X | | | X | X | X | X | | | X | | X | X |
| 4 | | | X | X | X | | | X | X | X | X | | X | X | | X |
| 5 | | | | X | X | X | | | X | X | X | X | | X | X | X |

4. S4 层是在C3层基础上进行下采样，前面已述。在后面的层中每一层节点个数比较少，都是全连接层，这里不再赘述。

**小结：**

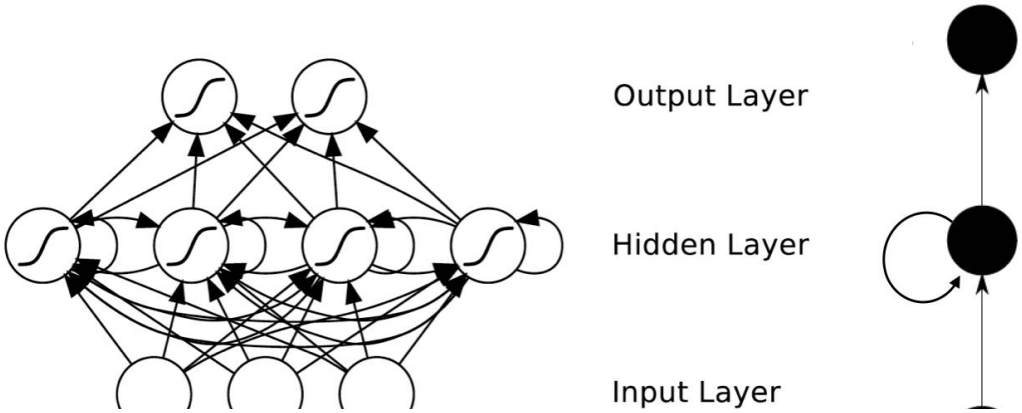
经过计算，LeNet-5系统总共需要大约13万个参数，这与前面提到的全连接系统每个隐藏层就需要百万个参数有着天壤之别，极大地减少了计算量。

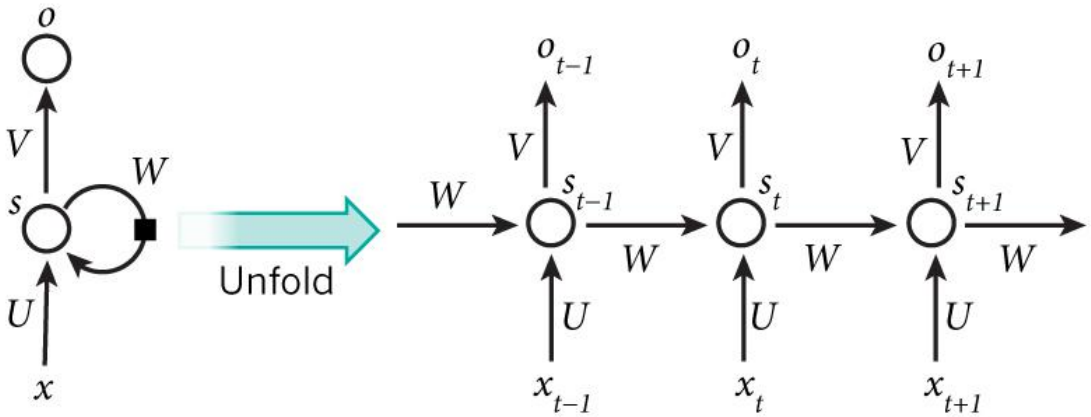
在以上的识别系统中，每个特征图提取后都紧跟着一个用来求局部平均与二次提取的亚取样层。这种特有的两次特征提取结构使得网络对输入样本有较高的畸变容忍能力。也就是说，卷积神经网络通过局部感受野、共享权值和亚取样来保证图像对位移、缩放、扭曲的鲁棒性。

# Recurrent neural networks



- RNNs的目的使用来处理序列数据,应用领域：NLP，如词向量表达、语句合法性检查、词性标注等



# LSTMs

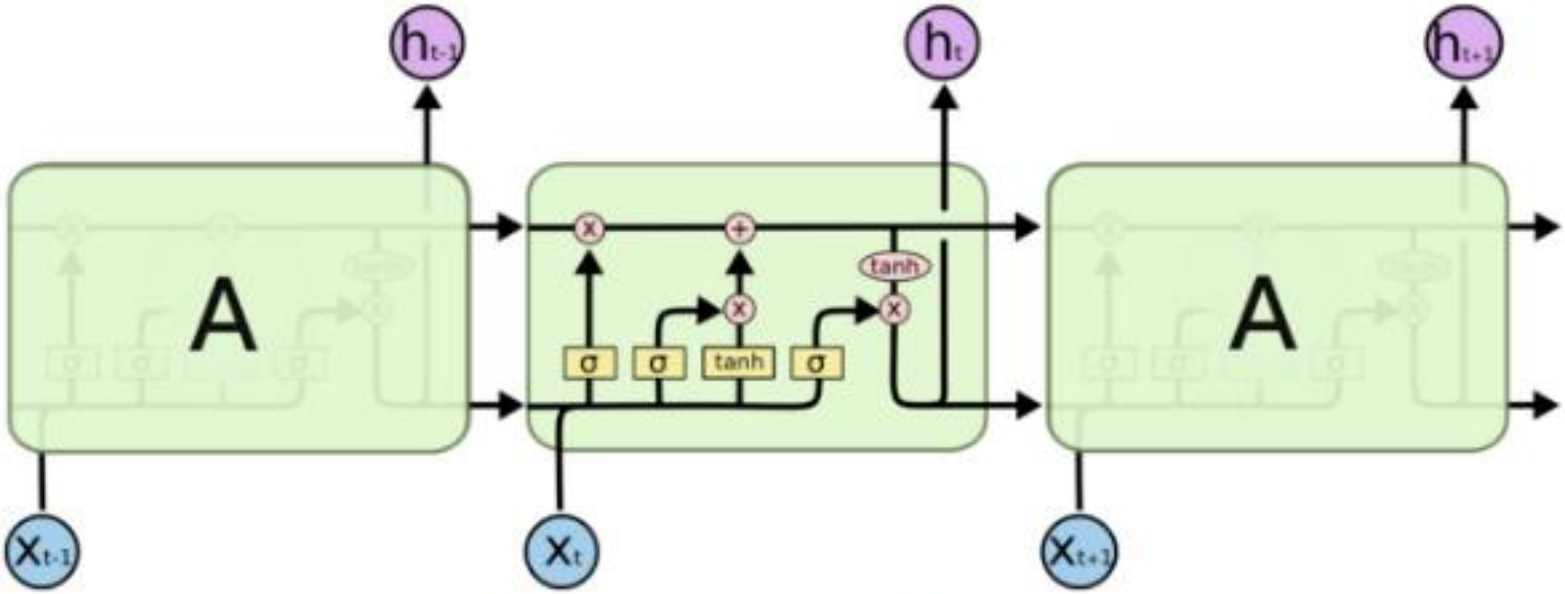


- 长短期内存网络(Long Short Term Memory networks)是一种特殊的RNN类型，可以学习长期依赖关系。
- LSTMs 刻意的设计去避免长期依赖问题。记住长期的信息在实践中RNN几乎默认的行为，但是却需要很大的代价去学习这种能力。

- LSTM同样也是链式结构，但是重复的模型拥有不同的结构，它与单个的神经网层不同，它有四个， 使用非常特别方式进行交互。

# LSTMs






The repeating module in an LSTM contains four interacting layers.

# TensorFlow 是什么



- 是一个深度学习库，由 Google 开源，可以对定义在Tensor(张量)上的函数自动求导。
- Tensor(张量)意味着 N 维数组，Flow(流)意味着基于数据流图的计算，TensorFlow即为张量从图的一端流动到另一端
- 亮点是支持异构设备分布式计算，它能够在各个平台上自动运行模型，从电话、单个CPU / GPU到成百上千GPU卡组成的分布式系统
- 支持CNN、RNN和LSTM算法，是目前在 Image，NLP 最流行的深度神经网络模型

# 为什么需要 TensorFlow 等库



- 解决深度学习模型训练时的2个问题
  - 1. 较大的数字或者张量在一起相乘百万次的处理，使得整个模型代价非常大。
  - 2. 手动求导耗时非常久

# TensorFlow 的优点

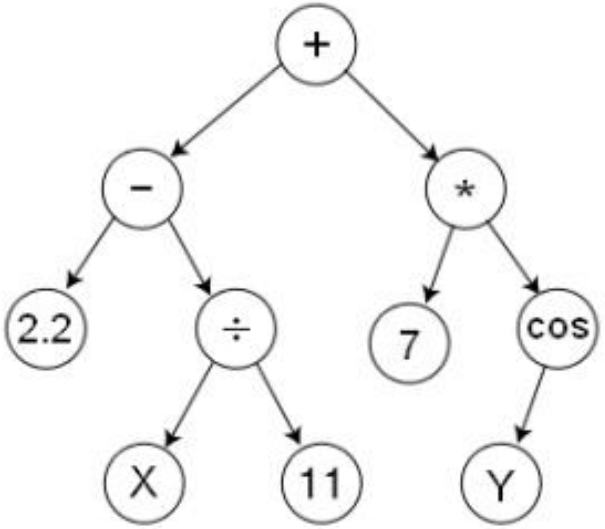


- 第一，基于Python，写的很快并且具有可读性。
- 第二，在多GPU系统上的运行更为顺畅。
- 第三，代码编译效率较高。
- 第四，社区发展的非常迅速并且活跃。
- 第五，能够生成显示网络拓扑结构和性能的可视化图。

# TensorFlow 的工作原理



- TensorFlow是用数据流图(data flow graphs)技术来进行数值计算的。
- 数据流图是描述有向图中的数值计算过程。
- 有向图中，节点通常代表数学运算，边表示节点之间的某种联系，它负责传输多维数据(Tensors)。
- 节点可以被分配到多个计算设备上，可以异步和并行地执行操作。因为是有向图，所以只有等到之前的入度节点们的计算状态完成后，当前节点才能执行操作。




$$\left(2.2-\left(\frac{X}{11}\right)\right)+\left(7*\cos(Y)\right)$$

# TensorFlow基本术语



- Devices
  - 一块可以用来运算并且拥有自己的地址空间的硬件，比如GPU和CPU
- eval
  - Tensor 的一个方法，返回 Tensor 的值。触发任意一个图计算都需要计算出这个值。只能在一个已经启动的会话的图中才能调用该 Tensor 值
- Feed
  - TensorFlow 的一个概念：把一个 Tensor 直接连接到一个会话图表中的任意节点。feed 不是在构建图(graph)的时候创建，而是在触发图的执行操作时去申请。一个 feed 临时替代一个带有 Tensor 值的节点。把feed数据作为run( )方法和eval( )方法的参数来初始化运算。方法运行结束后，替换的 feed 就会消失，而最初的节点定义仍然还在。可以通过tf.placeholder( )把特定的节点指定为 feed 节点来创建它们。
- Fetch
  - TensorFlow中的一个概念：为了取回运算操作的输出结果。取回的申请发生在触发执行图操作的时候，而不是发生在建立图的时候。如果要取回一个或多个节点（node）的 Tensor 值，可以通过在 Session 对象上调用run( )方法并将待取回节点（node）的列表作为参数来执行图表(graph)。
- Graph
  - 把运算任务描述成一个直接的无环图形（DAG），图表中的节点（node）代表必须要实现的一些操作。图中的边代表数据或者可控的依赖。GratheDef 是系统中描述一个图表的协议(api)，它由一个 NodeDefs 集合组成。一个GraphDef可以转化成一个更容易操作的图表对象
- IndexedSlices
  - 在 Python API 中，TensorFlow 仅仅在第一维上对 Tensor 有所体现。如果一个 Tensor 有k维，那么一个 IndexedSlices 实例在逻辑上代表一个沿着这个 Tensor 第一维的(k-1)维切片的集合。切片的索引被连续储存在一个单独的一维向量中，而对应的切片则被拼接成一个单独的k维 Tensor。如果 sparsity 不是受限于第一维空间，请用 SparseTensor。

# TensorFlow基本术语



- Node
  - 图中的一个元素。 把启动一个特定操作的方式称为特定运算图表的一个节点，包括任何用来配置这个操作的属性的值。对于那些多形态的操作，这些属性包括能完全决定这个节点（Node）签名的充分信息
- Operation
  - 在 TensorFlow 的运行时中，它是一种类似 add 或 matmul 或 concat的运算。可以用how to add an op中的方法来向运行时添加新的操作。
  - 在 Python 的API中，它是图中的一个节点。在tf.Operation类中列举出了这些操作。一个操作(Operation)的 type 属性决定这个节点（node）的操作类型，比如add和matmul
- Run
  - 在一个运行的图中执行某种操作的行为。要求图必须运行在会话中。
  - 在 Python 的 API 中，它是 Session 类的一个方法tf.Session.run。可以通过 Tensors 来订阅或获取run( )操作。
  - 在C++的API中，它是tensorflow::Session类 的一个方法。
  - Session
  - 启动图的第一步是创建一个 Session 对象。Session 提供在图中执行操作的一些方法。
  - 在 Python API中，使用tf.Session。
  - 在 C++ 的API中，tensorflow::Session是用来创建一个图并运行操作的类。
- Shape
  - Tensor 的维度和它们的大小。
  - 在一个已经启动的图中，它表示流动在节点（node）之间的 Tensor 的属性。一些操作对 shape 有比较强的要求，如果没有 Shape 属性则会报告错误。
  - 在 Python API中，用创建图的 API 来说明 Tensor 的 Shape 属性。Tensor 的Shape 属性要么只有部分已知，要么全部未知。详见tf.TensroShape
  - 在C++中，Shape 类用来表示 Tensor 的维度。tensorflow::TensorShape。

# TensorFlow基本术语



- SparseTensor
  - 在 Python API 中，它用来表示在 TensorFlow 中稀疏散落在任意地方的 Tensor 。SparseTensor 以字典-值格式来储存那些沿着索引的非空值。换言之，m个非空值，就包含一个长度为m的值向量和一个由m列索引(indices)组成的矩阵。为了提升效率，SparseTensor 需要将 indice（索引）按维度的增加来按序存储，比如行主序。如果稀疏值仅沿着第一维度，就用 IndexedSlices。
- Tensor
  - Tensor是一种特定的多维数组。比如，一个浮点型的四维数组表示一小批由[batch,height，width，channel]组成的图片。
  - 在一个运行的图(graph)中，它是一种流动在节点（node）之间的数据。 在 Python 中，Tensor 类表示添加到图的操作中的输入和输出，见tf.Tensor，这样的类不持有数据。
  - 在C++中，Tensor是方法Session::Run( )的返回值，见tensorflow::Tensor，这样的 Tensor 持有数据。

# TensorFlow 基本用法



- 你需要理解在TensorFlow中，是如何：
  - 将计算流程表示成图；
  - 通过Sessions来执行图计算；
  - 将数据表示为tensors；
  - 使用Variables来保持状态信息；
  - 分别使用feeds和fetches来填充数据和抓取任意的操作结果；

# TensorFlow和普通的Numpy的对比



## Simple Numpy Recap

```
In [23]: import numpy as np

In [24]: a = np.zeros((2,2)); b = np.ones((2,2))

In [25]: np.sum(b, axis=1)
Out[25]: array([ 2.,  2.])

In [26]: a.shape
Out[26]: (2, 2)

In [27]: np.reshape(a, (1,4))
Out[27]: array([[ 0.,  0.,  0.,  0.]])
```

## Repeat in TensorFlow

```
In [31]: import tensorflow as tf

In [32]: tf.InteractiveSession()

In [33]: a = tf.zeros((2,2)); b = tf.ones((2,2))

In [34]: tf.reduce_sum(b, reduction_indices=1).eval()
Out[34]: array([ 2.,  2.], dtype=float32)

In [35]: a.get_shape()
Out[35]: TensorShape([Dimension(2), Dimension(2)])

In [36]: tf.reshape(a, (1, 4)).eval()
Out[36]: array([[ 0.,  0.,  0.,  0.]], dtype=float32)
```

*More on Session soon*

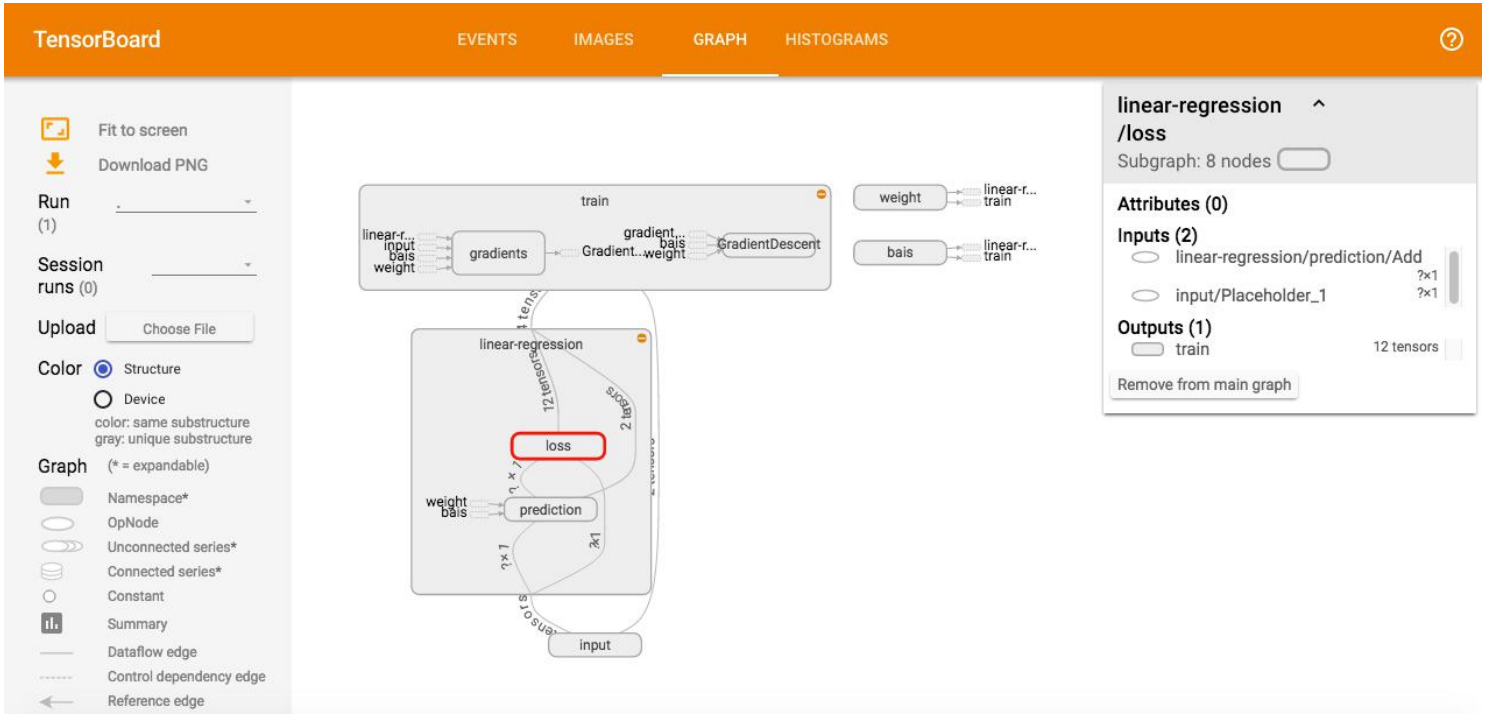
*More on .eval() in a few slides*

*TensorShape behaves like a python tuple.*

# TensorFlow工具



- TensorBoard



# TensorFlow演示

Q&A

| 讲座 | 9月10日论C语言编程的面向对象<br>宋宝华 ， 《Linux设备驱动开发详解》的作者 |
| --- | --- |

北京公开课 9月25-26日

# 人工智能，机器学习和深度学习

| 更多... | uml.org.cn 文章, 文挡, 课程 |
| --- | --- |