# 卷积神经网络

## 基本概念

常用于分析视觉图像，接受多个特征图作为输入，而不是向量。

卷积神经网络通常由五个部分组成：

1. **Input layer**

去均值：把输入数据的各个维度都中心化为0

归一化：减少特征取值范围的差异

PCA/白化：降维，各特征轴上的数据归一化

1. **CONV layer**

每个神经元看作是一个filter，receptive field滑动，filter对局部数据计算。重点是理解filter的移动和计算方式，这又涉及到stride和padding的问题。

一组固定的窗口和不同窗口内的数据做内积的过程，称为卷积。

1. **RELU layer**

把卷积层的输出结果做非线性的映射。CNN采用的激活函数一般是RELU，它的特点是收敛快且求梯度简单，sigmoid效果不好。

1. **Pooling layer**

夹在连续的卷积层中间，压缩数据和参数的量，一定程度上减少过拟合。如果输入的是图像，那么池化层的作用就是压缩图像，保留重要的特征。

方法上多用Max pooling，在每个特定的小窗口上选取最大的数输出。

1. **FC layer**

一般全连接层在卷积神经网络的尾部，和普通神经网络的连接方式相同

## 一些需要注意的点

### 训练算法

* 损失函数定义，衡量预测值和真实值之间的差异
* 找到最小化损失函数的参数w和b，CNN用的算法是SGD（随机梯度下降）

随机梯度下降是随机选取一些样本，近似所有的样本，用以调整参数，有些时候梯度下降会碰到鞍点或者一些局部最优解，依靠SGD可以走出这些点，而且只要超参数合理，大方向仍然是向全局最优解进行训练的。

### Fine-tuning

Fine-tuning就是使用已使用于其它目标、预训练好模型的权重或是部分权重，作为初始值开始训练。

* 复用相同层的权重，新定义层随机选取权重初始值
* 调大新定义层的学习率，调小复用层学习率

### 常用框架

Caffe，pytorch， tensorflow

# Keras

由python编写的开源人工神经网络库，支持卷积神经网络和循环神经网络。

同时它提供了mnist数据集，在构建简单的神经网络这方面较为简便。

使用它的步骤大致如下：

1. 安装库，导入库和需要的模块
2. 加载图像数据，预处理输入数据和类标签
3. 构建CNN
4. 编译模型，拟合模型
5. 评估模型

# 学习到的注意事项

对于mnist手写数据集，其像素值范围是0到255，通过将数据类型转换为float32并除以255，能将数据规范化到（0，1）之间。

Adam梯度下降优化算法：它的名字来源于“Adaptive Moment Estimation”，即自适应矩估计。其特点是计算梯度的一阶矩估计（动量）和二阶矩估计（平方梯度）来确定每个参数的学习率。