1.收集视频数据

利用python爬虫得到视频数据。Python爬虫：通过程序模拟浏览器请求站点的行为，把站点返回的HTML代码/JSON数据/二进制数据（图片、视频） 爬到本地，进而提取自己需要的数据，存放起来使用。代码：create\_db()创建数据库，save\_db()将数据保存到本地。Map(a,b):a需要异步执行的函数，b可迭代对象，为a每次运行提供参数。

2.视频转音频，作为卷积神经网络的输入

利用ffmpeg对视频进行处理，提取音频文件。对音频进行傅里叶变换，得到各个时间点上的频谱图。再利用连续时间段内的若干频谱图得到声谱图（连续的若干频谱），声谱图作为卷积神经网络的输入。声谱图是二维数据，卷积神经网络主要应用在二维数据识别网络，因此不必再经过转换。另外，声谱图比音频本身小很多，训练快速。

3.用cool edit去除人声

人声的声像特点是声像位置在整个声场的中央，且音频集中在中频和高频。因此，可以利用人声和环境音的区别，将二者分离。

4.差频法得到人声，作为卷积神经网络的输出

通过2、3两步，得到了原声和去除人声的音频。之后通过差频法反向补偿得到降噪后的语音。通过二者频谱图的差异，得到纯净的人声作为卷积神经网络的输出。

5.训练模型

卷积神经网络的训练过程分为两个阶段。第一个阶段是数据由低层次向高层次传播的阶段，即前向传播阶段。另外一个阶段是，当前向传播得出的结果与预期不相符时，将误差从高层次向底层次进行传播训练的阶段，即反向传播阶段。

训练过程为：

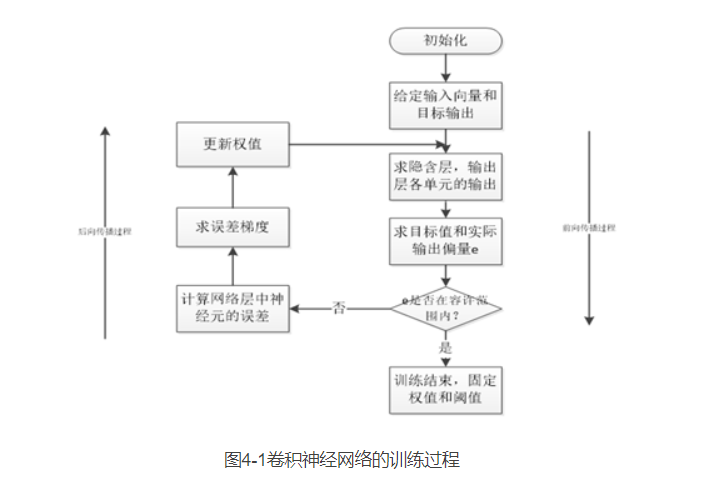
1、网络进行权值的初始化；

2、输入数据经过卷积层、下采样层、全连接层的向前传播得到输出值；

3、求出网络的输出值与目标值之间的误差；

4、当误差大于我们的期望值时，将误差传回网络中，依次求得全连接层，下采样层，卷积层的误差。各层的误差可以理解为对于网络的总误差，网络应承担多少；当误差等于或小于我们的期望值时，结束训练。

5、根据求得误差进行权值更新。然后在进入到第二步。



\*卷积神经网络原理：

• 数据输入层/ Input layer  
• 卷积计算层/ CONV layer  
• ReLU激励层 / ReLU layer  
• 池化层 / Pooling layer  
• 全连接层 / FC layer

数据输入层

去均值、归一化、PCA/白化

卷积计算层

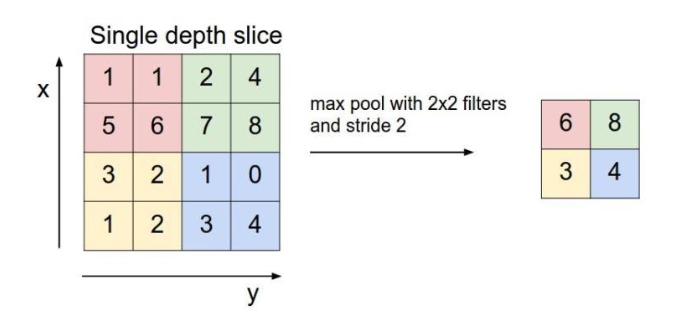
• 局部关联。每个神经元看做一个滤波器(filter)  
• 窗口(receptive field)滑动， filter对局部数据计算

激励层

把卷积层输出结果做非线性映射。CNN采用的激励函数一般为ReLU(The Rectified Linear Unit/修正线性单元)，它的特点是收敛快，求梯度简单，但较脆弱。

池化层

池化层夹在连续的卷积层中间， 用于压缩数据和参数的量，减小过拟合。具体作用：特征不变性、特征降维和防止过拟合。用到方法：max pooling



全连接层

两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部。

最小平方差损失函数

损失函数基本都是由真实值和预测值两部分组成，正确的损失函数，可以起到让预测值一直逼近真实值的效果，当预测值和真实值相等时，loss值最小。反向传播是一个不断更新权重w和偏差b的过程，当loss函数值为0时，表示预测值达到真实值，这时候权重和偏差不再需要更新。

卷积神经网络优化

**改进网络结构**

首先通过1x1的卷积层压缩通道数，然后使用NxN的卷积层计算，最后又用1x1的卷积层恢复到原来的通道数。这种方法大量减少了参数数量，提高了准确率。