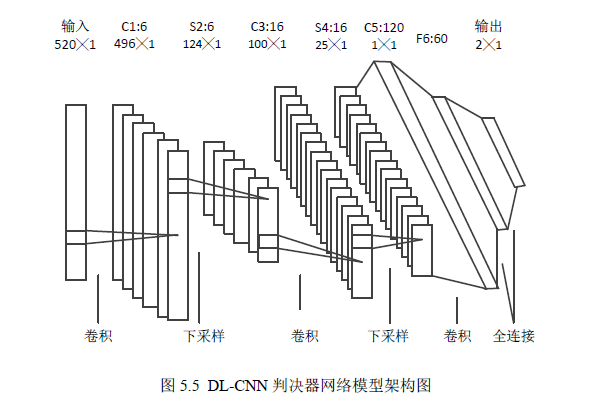
对卷积神经网络(CNN)的简单理解

卷积神经网络是一个多层的神经网络，在典型的CNN 中，开始几层都是卷积层和下采样层的交替，然后是几个全连接层，最后是输出层。



1.输入：520\*1

2. C1层：特征提取层，每个神经元的输入与前一层的局部感受野相连，并提取该局部的特征，一旦该局部特征被提取后，它与其他特征间的位置关系也随之确定下来。

然后注意C1 层的时候是用了6种 25\*1的卷积核，每一种卷积核都能提取一张特征图， 也就是说有多少种不同的卷积核，就会多少个不同的特征图， 因为不同的卷积核可以进行不同的特征提取，当然这里的特征图的数量是根据你自己的需求而定的，一般图像很大的话会需要很多的特征图， 而对于上图中520\*1的输入，6个特征图就比较合适。

到这里你会问，为什么需要设计多种的卷积核，全连接的不行么？ 这就是卷积神经网络最重要的特点：权值共享来减少网络参数，这时候就可以通过输入的大小以及卷积核的大小来确定隐含层的神经元的个数了，公式为：520-25+1=496个隐层神经元，也就是每一个特征图的大小就是496。那么C1一共有 （25\*1＋1）\*6=156个参数，其中25\*1为卷积核的大小，1为一个bias参数， 6为6种卷积核。

3.S2层，利用局部相关性的原理，对信号进行子抽样，减少数据处理量同时保留有用信息。该层采用的下采样倍数为4，下采样方法为首先将C1 得到的特征图中相邻的4 元素相加求和，而后乘以一个可训练参数以及加上一个可训练阈值，最后通过tansig 函数计算得到本层的输出结果。通过下采样操作后，S2 中每个特征图的大小是C1 中特征图大小的1/4，得到6 个124 × 1 的特征图层。该层的可训练参数和为6 + 6 = 12。

4.C3层，C3 层也是一个卷积层，它同样通过25 × 1 的卷积核去卷积S2 层，然后得到的每个特征图的大小为(124 − 25 + 1) = 100 ，它有16 种不同的卷积核，所以存在16个特征图。C3 中的每个卷积核与S2 中特征图的连接方式如图5.6所示(非全连接)，该层共有(25 + 1) × 16 = 416 个可训练参数？60\*25+16？包括16 个卷积核参数以及16 个可训练阈值。

5. S4 层是一个下采样层，下采样的方法同S2，下采样的倍数也是4，该层将C3 中的特征图中相邻的4 个元素相加求和，乘以可训练参数和增加可训练阈值后，通过tansig 函数处理，得到16 个大小为25 × 1 的特征图层。该层的可训练参数数目为16 + 16 = 32。

6. C5 层是卷积层，有120 个25 × 1 的卷积核。每个卷积核与S4 层的全部16 个特征图相连接。由于S4 层特征图的大小为25 × 1，所以经过卷积以及加上可训练阈值后C5 特征图的大小为1 × 1。C5 层有(25 + 1) × 120 = 3120 个可训练参数？

7. F6 层是全连接层，有60 个单元，与C5 层全相连。输入向量与该层的权值矩阵相乘，再加上阈值向量后，经过tansig 函数得到60×1 的输出向量。F6 层有60×120 = 7200个连接权值和60 个可训练阈值，总共7260 可训练参数。

8.输出层。输出层由两个神经元构成，输入向量与权值矩阵相乘，加上阈值向量，通过tansig函数得到本层2 × 1 的输出向量。

算法代码：

//C1-S2

**void** **ex\_c1**(**float** in\_layer[520\*1], **float** out\_layer[6\*124\*1],**const** **float** bias[6],

**const** **float** weight[6\*25\*1],**const** **float** sample[6],**const** **float** samplebias[6]) {

**int** r,m,k;

**static** **float** y[6\*496\*1];

**static** **float** y2[6\*124\*1];

**for**(r=0; r<6; r++){

**for**(m=0; m<496; m++){

y[r\*496+m]=bias[r];

}

}

**for**(r=0; r<6; r++){

**for**(m=0; m<496; m++){

**for**(k=0; k<25; k++){

y[r\*496+m]+= in\_layer[m+k] \* weight[r\*25+k];

}

}

}

**for**(r=0; r<6\*496; r++){

y[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* y[r]);

}

**for**(r=0; r<6; r++){

**for**(m=0; m<124; m++){

y2[r\*124+m]=samplebias[r];

y2[r\*124+m]+= (y[r\*496+m\*4]+ y[r\*496+m\*4+1]+y[r\*496+m\*4+2]+y[r\*496+m\*4+3])\* sample[r];

}

}

**for**(r=0; r<6\*124; r++){

y[r]=y2[r];

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* y[r]);

}

}

//C3-S4

**void** **ex\_c3**(**float** in\_layer[6\*124], **float** out\_layer[16\*100],**const** **float** bias[16],

**const** **float** weight[6\*16\*25],**const** **float** sample[16],**const** **float** samplebias[16]) {

**int** r,m,k,q;

**static** **float** y[16\*100];

**static** **float** y2[16\*25];

**for**(r=0; r<16; r++){

**for**(m=0; m<100; m++){

y[r\*100+m]=bias[r];

}

}

**for**(r=0; r<16; r++){

**for**(q=0;q<6;q++){

**for**(m=0; m<100; m++){

**for**(k=0; k<25; k++){

y[r\*100+m]+= in\_layer[q\*124+m+k] \* weight[(r\*6+q)\*25+k];

}

}

}

}

**for**(r=0; r<16\*100; r++){

y[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* y[r]);

}

**for**(r=0; r<16; r++){

**for**(m=0; m<25; m++){

y2[r\*25+m]=samplebias[r];

y2[r\*25+m]+= (y[r\*100+m\*4]+ y[r\*100+m\*4+1]+y[r\*100+m\*4+2]+y[r\*100+m\*4+3])\* sample[r];

}

}

**for**(r=0; r<16\*25; r++){

y[r]=y2[r];

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* y[r]);

}

}

//C5

**void** **ex\_c5**(**float** in\_layer[16\*25], **float** out\_layer[120],**const** **float** bias[120],

**const** **float** weight[16\*120\*25]) {

**int** r,k,q;

**for**(r=0; r<120; r++){

out\_layer[r]=bias[r];

}

**for**(r=0; r<120; r++){

**for**(q=0;q<16;q++){

**for**(k=0; k<25; k++){

out\_layer[r]+= in\_layer[q\*25+k] \* weight[(r\*16+q)\*25+k];

}

}

}

**for**(r=0; r<120; r++){

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* out\_layer[r]);

}

}

//F6

**void** **ex\_F6**(**float** in\_layer[120], **float** out\_layer[60],**const** **float** bias[60],

**const** **float** weight[120\*60]) {

**int** r,q;//q--in,r--out

**for**(r=0; r<60; r++){

out\_layer[r]=bias[r];

}

**for**(r=0; r<60; r++){

**for**(q=0;q<120;q++){

out\_layer[r]+= in\_layer[q] \* weight[r\*120+q];

}

}

**for**(r=0; r<60; r++){

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \*out\_layer[r]);

}

}

//F7

**void** **ex\_F7**(**float** in\_layer[60], **float** out\_layer[30],**const** **float** bias[30],

**const** **float** weight[60\*30]) {

**int** r,q;

**for**(r=0; r<30; r++){

out\_layer[r]=bias[r];

}

**for**(r=0; r<30; r++){

**for**(q=0;q<60;q++){

out\_layer[r]+= in\_layer[q] \* weight[r\*60+q];

}

}

**for**(r=0; r<30; r++){

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* out\_layer[r]);

}

}

//F8

**void** **ex\_F8**(**float** in\_layer[30], **float** out\_layer[2],**const** **float** bias[2],

**const** **float** weight[60]) {

**int** r,q;

**for**(r=0; r<2; r++){

out\_layer[r]=bias[r];

}

**for**(r=0; r<2; r++){

**for**(q=0;q<30;q++){

out\_layer[r]+= in\_layer[q] \* weight[r\*30+q];

}

}

**for**(r=0; r<2; r++){

out\_layer[r]=1.71593428 \* **tanh**(0.6666666666666666667 \* out\_layer[r]);

}

}