# 第一章 绪论

#### 1.1研究背景

人工智能概念提出已经60余年随着物联网大数据的发展，越来越多的人工智能应用逐步实现，自动驾驶技术是目前最引人注目的人工智能技术集成应用之一。

自动驾驶中，很重要的一个问题就是自动识别道路边的交通标志。这个部分需要很高的实时性才能够保证行车安全。由于自动驾驶对于交通标志检测实时性的高要求, 单纯的CPU硬件架构已经很难满足系统高实时性的要求。随着半导体技术的不断发展，近年来有一些新的硬件开始广泛的应用在图像处理和人工智能领域，为计算密集的系统加速，这其中就包括FPGA和GPU。

FPGA(Field－Programmable Gate Array) 即现场可编程门阵列。随着FPGA芯片设计集成度不断提高，越来越复杂的功能可以在FPGA上得以实现。由于FPGA的可并行性，许多在传统CPU上只能串行执行的任务和算法可以充分利用FPGA的并行性来重新设计和实现。大大的提高了算法的执行速度。而且FPGA 10W~30W的功耗相比传统的CPU和GPU 100W~200W的功耗而言有着更低的功耗，更适合于对于功耗要求较高的嵌入式系统中。

OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉库，可以运行在Linux、Windows、Android和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

在此背景下，本研究将设计并实现一套基于CPU+FPGA的异构计算的硬件的嵌入式交通标志识别系统。该系统配合卷积神经网络及OpenCV的图像处理识别算法，将这些算法的运算密集部分使用FPGA进行加速。实时的从摄像头采集到的车前方的图像中提取出相应的交通标志牌上的信息，并将这些信息反馈给汽车的控制系统，使汽车自动做出相应的反应动作。同时该系统能够满足嵌入式场景下对于系统低功耗的要求。

#### **1.2 国内外研究现状**

当前已经有的关于交通标识识别算法的研究。一般的路标检测是基于颜色阈值或者形状分割定位路标的具体位置。使用形状检测来分割交通标志时，摄像头拍摄到的交通标志会因为拍摄角度的问题导致图片中交通标志的形状发生改变导致识别率偏低且运算量大。由于交通标志的颜色较统一，故有很多论文使用RGB颜色空间作为识别标准，但由于RGB颜色空间受光照影响明显，会降低识别准确性。后来又有人提出了基于HSV颜色空间的交通标志识别方法，很好的克服了光照强度对交通标志识别的影响。

对于交通标志内容的识别方法常用的方法有模板匹配法。即计算待识别的图片和一个或几个特定模板图片的相似度完成分类。但是这种方法对于待识别图片的角度很敏感，现实环境中路标的角度是多变的，所以该方法并不合适。机器学习和深度学习的发展为这一问题带来了新的机会，很多文献中使用卷积神经网络或者SVM算法来完成图片内容的识别。卷积神经网络是近年发展起来的，并引起广泛重视的一种高效识别方法。20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称CNN）。现在，CNN已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。

进行交通标志牌的识别是计算密集型任务，如果是从高帧率的视频中去识别，具有很高的实时性要求，只在传统的单纯CPU硬件架构上运行速度很慢，难以达到实际使用的目的。有少部分学者使用FPGA来对交通标志牌识别系统的部分关键节点进行硬件加速，但只限于在前期的图像预处理上，后期也只是使用简单的模板匹配作为识别算法，未使用神经网络作为交通标志的识别算法，准确性和鲁棒性较差。如果将神经网络以及图像前期预处理都放入FPGA中，可以使得原来需要数个CPU周期完成的计算任务能够通过FPGA在很少的时钟周期内就可以完成。同时在FPGA上进行合理的硬件结构设计,提升FPGA的资源使用率和吞吐量。

#### **1.3 研究内容**

研究内容是基于异构硬件加速的交通标志牌检测系统的设计与实现。实现一套能够将汽车行驶过程中前方的视野中的路标检测识别出来并通知汽车控制系统和司机的系统。致力于在整个的系统设计中，体现软硬协同的设计思路，即在FPGA部分实现图像预处理中关于色域转换，降噪，膨胀，腐蚀，神经网络中的卷积池化等适合于并行和流水化的操作以来提高整体系统关键点的速度和吞吐量。在CPU部分，通过软件来完成硬件数据流管理管理，通知用户，状态切换，结果输出等一些流程控制上的工作。将FPGA的高速性和软件在流程控制上的灵活性有机的结合起来。

**1.3.1 卷积神经网络的FPGA加速**

CNN是著名的深度学习架构，从人工神经网络扩展而来，它已经大量用于不同应用，包括视频监控，移动机器人视觉，数据中心的图像搜索[引擎](https://www.baidu.com/s?wd=%E5%BC%95%E6%93%8E&tn=24004469_oem_dg&rsv_dl=gh_pc_csdn)等。受生物视觉神经行为的启发，CNN用多层[神经元](https://www.baidu.com/s?wd=%E7%A5%9E%E7%BB%8F%E5%85%83&tn=24004469_oem_dg&rsv_dl=gh_pc_csdn)相连处理数据，在图像识别中可获得很高准确率。

一个典型CNN由两部分组成：特征提取器 + 分类器。特征提取器用于过滤输入图像，产生表示图像不同特征的特征图。这些特征可能包括拐角，线，圆弧等，对位置和形变不敏感。特征提取器的输出是包含这些特征的低维向量。该向量送入分类器（通常基于传统的人工神经网络）分类器的目的是决定输入属于某个类别的可能性。一个典型CNN包括多个计算层，例如，特征提取器可能包括几个卷积层和可选的下采样层。卷积层收到N个特征图作为输入，每个输入特征图被一个K \* K的核卷积，产生一个输出特征图的一个像素。滑动窗的间隔为S，一般小于K。总共产生M个输出特征图用于下一卷积层。计算神经网络中的一个卷积层其实质是一个多层嵌套循环的乘法和加法运算。由于CNN的这种特殊计算模式，通用处理器实现CNN并不高效，所以很难满足性能需求。而FPGA由于其可并行计算的特性，基于FPGA的加速器由于其更好的性能，高能效，快速开发周期以及可重配置能力吸引了越来越多研究者的注意。

FPGA对卷积神经网络的加速主要来自于循环展开和设计流水线两方面。循环展开是对于卷积神经网络中存在的大量嵌套循环，在某些层级进行展开，使用FPGA的资源并行计算。比如一个5X5的二维卷积 可以在1~3个时钟周期内同时使用FPGA的25个乘法器进行运算。这样就实现了原来在CPU中需要25个乘法周期内完成的工作在一个时钟周期内完成。虽然FPGA的主频只有50MHz~200MHz 只有通用CPU的 1/10。但只要进行合理的并行结构设计，就可以将大量的计算并行化。

**1.3.2 图像预处理在FPGA加速**

在本系统中，首先需要使用颜色阈值分割的方法从摄像头采集到的图片中选定大致的交通标志所在的位置，其中涉及到大量的色域变换，比较，过滤以及膨胀腐蚀运算操作。虽然对于一个像素的运算简单，但是对于视频流中的大量需要实时处理的图片，CPU无法达实时的处理要求。  而用FPGA做图像处理最关键的一点优势就是：FPGA能进行实时流水线运算，能达到最高的实时性。因此在一些对实时性要求非常高的应用领域，做图像处理基本就只能用FPGA。例如在一些分选设备中图像处理基本上用的都是FPGA，因为在其中相机从看到物料图像到给出执行指令之间的延时大概只有几毫秒，这就要求图像处理必须很快且延时固定，只有FPGA进行的实时流水线运算才能满足这一要求。

FPGA进行图像处理的优势在于进行的实时流水线运算和DSP，GPU等进行的图像处理运算不同。DSP，GPU，CPU对图像的处理基本是以帧为单位的，从相机采集的图像数据会先存在内存中，然后GPU会读取内存中的图像数据进行处理。假如采集图像的帧率是30帧，那么DSP，GPU要是能在1/30秒内完成一帧图像的处理，那基本上就能算是实时处理。

FPGA对图像进行实时流水线运算是以行为单位的。FPGA可以直接和图像传感器芯片连接获得图像数据流，如果是RAW格式的则还可以进行差值以获得RGB图像数据。FPGA能进行实时流水线处理的关键是它可以用其内部的Block Ram缓存若干行的图像数据。Block Ram类似于CPU里面的Cache，但Cache不可控制的，但Block Ram是完全可控的，可以用它实现各种灵活的运算处理。这样FPGA通过缓存若干行图像数据就可以对图像进行实时处理，少量数据数据就这样一边流过就一边处理，不需要送入DDR缓存了之后再读出来处理,大大减小了开销。

本系统预处理的过程中滤波、取边缘、膨胀腐蚀等算法一大类用3x3到NxN的算子进行的和卷积神经网络中最前面的卷积层运算是类似的，所以也可以借鉴卷积神经网络在FPGA上应用的例子。FPGA进行的这种算子法处理是并行流水线算法，其延时是固定,可以根据时钟周期来直接计算FPGA对图像进行预处理的速度。在密集运算中，消耗大量时间的操作往往不是运算本身，而是把数据在内存之间的搬移。GPU，CPU在进行运算时需要不停地在内存中读取和写入数据，这样内存带宽往往成了运算速度的瓶颈，数据读写过程中的功耗占的比重也不会小。FPGA则可以通过并行很多计算硬件的方法把要做的运算都展开，然后数据从中流过，完成一个阶段的运算之后就直接流入第二个阶段，不需要把一个计算阶段完成后的数据再送回内存中，再读出来交给下一个阶段的运算。如此可以节省很多时间。

**1.3.3 软硬件协同设计**

在上一个研究点中预处理部分有算法在FPGA上难以实现，所以需要将这部分算法的执行放到CPU中使用软件计算，这其中涉及Zynq中PL（FPGA）部分和PS（CPU）的数据交互问题。为了加快数据交互处理的能力，可以借助Xilinx的AXI总线， 该总线频率高，相对较外部DDR Ram交互更快，不易使数据传输成为整个处理流水线中的瓶颈。

#### **1.4 论文主要内容及章节安排**

本文的章节安排如下：

第一章为绪论。 介绍了交通标志识别算法的发展现状，FPGA在现在图像处理和深度神经网络中加速的应用情况。并阐述了本论文的整体结构以及章节安排。

第二章为相关技术介绍。。。

# **第二章 相关技术介绍**

#### **2.1 相关硬件平台**

**2.1.1 Zedboard**

ZedBoard是基于Xilinx Zynq-7000扩展式处理平台（EPP）的低成本开发板。此板可以运行基于Linux，Android，Windows或其他OS/ RTOS的设计。此外，可扩展接口使得用户可以方便访问处理系统和可编程逻辑。 Zynq-7000 EPP将ARM®处理系统和与Xilinx 7系列可编程逻辑完美地结合在一起，可以创建独特而强大的设计。

**2.2.2 Nexys4DDR**

Nexys4DDR 搭载了Xilinx A7系列芯片 可以在教学中方便的设计 Artix-7 FPGA 专门对高速逻辑设计进行优化。并且搭载了丰富的USB以太网接口。适用于构建从工业组合电路到嵌入式处理器等各种设计。开发板上搭载了温度传感器，数字麦克风以及其他IO设备。Nexys4开发板可以配合Xilinx 公司的开发套件Vivado完成多种应用的开发。

#### **2.2 图像处理相关技术 (基础的opencv 以及深度学习处理)**

**2.2.1交通标志数据库** GTSRB

The German Traffic Sign Recognition Benchmark是IJCNN 2011举办的一个图像分类挑战赛，其提供了GTSRB数据集，共包含了43类交通标志，训练样本39209张，测试样本12630张。提供的样本图像中包含了标志区域及其周围10%的区域（这是为采用边缘方法者所考虑），样本图像尺寸范围为15x15到250x250，当然它不一定是正方形。 每张图由90%的主体交通标志和10%的周边构成，即不用考虑分割的问题。

**2.2.2 图像色彩空间 （RGB->HSV转换）**

**(1) RGB 色彩空间**

**一般我们设备所采集到的数字图像均为RGB色彩空间的图像。**

RGB颜色空间以R(Red:红)、G(Green:绿)、B(Blue:蓝)三种基本色为基础，进行不同程度的叠加，产生丰富而广泛的颜色，所以俗称三基色模式。

RGB空间是生活中最常用的一个模型，电视机、电脑的CRT显示器等大部分都是采用这种模型。自然界中的任何一种颜色都可以由红、绿、蓝三种色光混合而成，现实生活中人们见到的颜色大多是混合而成的色彩。

RGB颜色空间以R(Red:红)、G(Green:绿)、B(Blue:蓝)三种基本色为基础，进行不同程度的叠加，产生丰富而广泛的颜色，所以俗称三基色模式。在大自然中有无穷多种不同的颜色，而人眼只能分辨有限种不同的颜色，RGB模式可表示一千六百多万种不同的颜色，在人眼看来它非常接近大自然的颜色，故又称为自然色彩模式。红绿蓝代表可见光谱中的三种基本颜色或称为三原色，每一种颜色按其亮度的不同分为256个等级。当色光三原色重叠时，由于不同的混色比例能产生各种中间色，例如，三原色相加可产生白色。所以RGB模式是加色过程。屏幕显示的基础是RGB模式，彩色印刷品却无法用RGB模式来产生各种彩色，所以，RGB模式常用于视频、多媒体与网页设计。

对图像处理而言，RGB是最为重要和常见的颜色模型，它建立在笛卡尔坐标系中，以红、绿、蓝三种  
　　基本色为基础，进行不同程度的叠加，产生丰富而广泛的颜色，俗称三基色模式。

RGB颜色空间是用一个单位长度的立方体来表示颜色的，黑蓝绿青红紫黄白8种常见颜色分别位居立方体的8个顶点，通常将黑色置于三维直角坐标系的原点，红绿蓝分别置于3根坐标轴土，整个立方体放在第1卦限内。如下图所示。而其中的青色与红色、紫色(或称品红色)与绿色、黄色与蓝色是互补色。各参数的取值范围是：R：0-255；G：0-255；B：0-255。参数值也称为三色系数或基色系数或颜色值，除以255后归一到0-1之间，但不是无穷多个而是有限多个值。由于每个灰度级都定为256，所以，红绿蓝分量全部组合起来共可表示256=2=16777216种不同的颜色。它比人眼能分辨的颜色种数多得多。因此，虽然自然界中的颜色非常多，但用RGB颜色空间来近似表达自然界中的颜色是完全够用了

1. **HSV色彩空间**

色调H

用角度度量，取值范围为0°～360°，从红色开始按逆时针方向计算，红色为0°，绿色为120°,蓝色为240°。它们的补色是：黄色为60°，青色为180°,品红为300°；

饱和度S

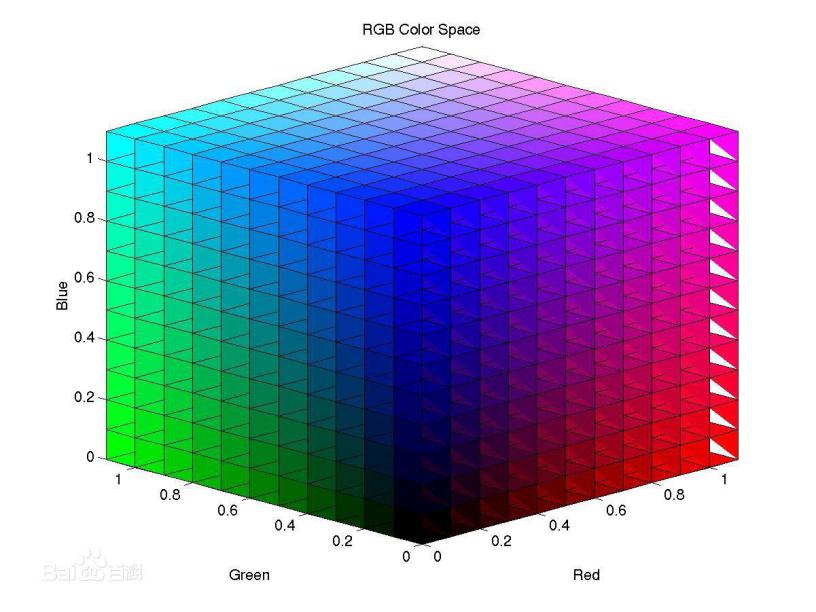
饱和度S表示颜色接近光谱色的程度。一种颜色，可以看成是某种光谱色与白色混合的结果。其中光谱色所占的比例愈大，颜色接近光谱色的程度就愈高，颜色的饱和度也就愈高。饱和度高，颜色则深而艳。光谱色的白光成分为0，饱和度达到最高。通常取值范围为0%～100%，值越大，颜色越饱和。

明度V

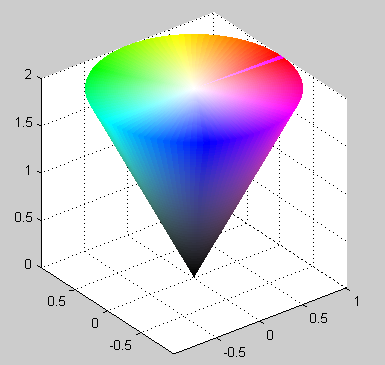
明度表示颜色明亮的程度，对于光源色，明度值与发光体的光亮度有关；对于物体色，此值和物体的透射比或反射比有关。通常取值范围为0%（黑）到100%（白）。

RGB颜色是面向硬件的，而HSV（Hue Saturation Value）颜色模型是面向用户的。

图：RGB色彩空间

HSV模型的三维表示从RGB立方体演化而来。设想从RGB沿立方体对角线的白色顶点

向黑色顶点观察，就可以看到立方体的六边形外形。六边形边界表示色彩，水平轴表示纯度，明度沿垂直轴测量。



**图：HSV色彩空间**

**2.2.3 图像处理技术 （滤波 膨胀 腐蚀）**

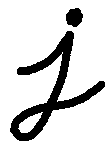
简单来讲，形态学操作就是基于形状的一系列图像处理操作。通过将 结构元素 作用于输入图像来产生输出图像最基本的形态学操作有二：腐蚀与膨胀(Erosion 与 Dilation)。 他们的运用广泛:

消除噪声

分割(isolate)独立的图像元素，以及连接(join)相邻的元素。

寻找图像中的明显的极大值区域或极小值区域。

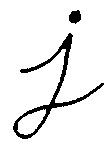
通过以下图像，我们简要来讨论一下膨胀与腐蚀操作(译者注：注意这张图像中的字母为黑色，背景为白色，而不是一般意义的背景为黑色，前景为白色）:



此操作将图像 IMG_257 与任意形状的内核 (IMG_258)，通常为正方形或圆形,进行卷积。

内核 IMG_259 有一个可定义的 锚点, 通常定义为内核中心点。

进行膨胀操作时，将内核 IMG_260 划过图像,将内核 IMG_261 覆盖区域的最大相素值提取，并代替锚点位置的相素。显然，这一最大化操作将会导致图像中的亮区开始”扩展” (因此有了术语膨胀 dilation )。对上图采用膨胀操作我们得到:



背景(白色)膨胀，而黑色字母缩小

腐蚀在形态学操作家族里是膨胀操作的孪生姐妹。它提取的是内核覆盖下的相素最小值。进行腐蚀操作时，将内核 IMG_263 划过图像,将内核 IMG_264 覆盖区域的最小相素值提取，以与膨胀相同的图像作为样本,我们使用腐蚀操作。从下面的结果图我们看到亮区(背景)变细，而黑色区域(字母)则变大了。

2.2.4 LeNet-5卷积神经网络

、卷积层

卷积层是卷积神经网络的核心基石。在图像识别里我们提到的卷积是二维卷积，即离散二维滤波器（也称作卷积核）与二维图像做卷积操作，简单的讲是二维滤波器滑动到二维图像上所有位置，并在每个位置上与该像素点及其领域像素点做内积。卷积操作被广泛应用与图像处理领域，不同卷积核可以提取不同的特征，例如边沿、线性、角等特征。在深层卷积神经网络中，通过卷积操作可以提取出图像低级到复杂的特征。

上图给出一个卷积计算过程的示例图，输入图像大小为H=5,W=5,D=3，即5×5大小的3通道（RGB，也称作深度）彩色图像。这个示例图中包含两（用K表示）组卷积核，即图中滤波器W0和W1。在卷积计算中，通常对不同的输入通道采用不同的卷积核，如图示例中每组卷积核包含（D=3）个3×3（用F×F表示）大小的卷积核。另外，这个示例中卷积核在图像的水平方向（W方向）和垂直方向（H方向）的滑动步长为2（用S表示）；对输入图像周围各填充1（用P表示）个0，即图中输入层原始数据为蓝色部分，灰色部分是进行了大小为1的扩展，用0来进行扩展。经过卷积操作得到输出为3×3×2（用Ho×Wo×K表示）大小的特征图，即3×3大小的2通道特征图，其中Ho计算公式为：Ho=(H−F+2×P)/S+1，Wo同理。 而输出特征图中的每个像素，是每组滤波器与输入图像每个特征图的内积再求和，再加上偏置bo，偏置通常对于每个输出特征图是共享的。输出特征图o[:,:,0]中的最后一个−2计算如上图右下角公式所示。

记住这几个符号：

H：图片高度；

W：图片宽度；

D：原始图片通道数，也是卷积核个数；

F：卷积核高宽大小；

P：图像边扩充大小；

S：滑动步长。

在卷积操作中卷积核是可学习的参数，经过上面示例介绍，每层卷积的参数大小为D×F×F×K。卷积层的参数较少，这也是由卷积层的主要特性即局部连接和共享权重所决定。

局部连接：每个神经元仅与输入神经元的一块区域连接，这块局部区域称作感受野（receptive field）。在图像卷积操作中，即神经元在空间维度（spatial dimension，即上图示例H和W所在的平面）是局部连接，但在深度上是全部连接。对于二维图像本身而言，也是局部像素关联较强。这种局部连接保证了学习后的过滤器能够对于局部的输入特征有最强的响应。局部连接的思想，也是受启发于生物学里面的视觉系统结构，视觉皮层的神经元就是局部接受信息的。

权重共享：计算同一个深度切片的神经元时采用的滤波器是共享的。例上图中计算o[:,:,0]的每个每个神经元的滤波器均相同，都为W0，这样可以很大程度上减少参数。共享权重在一定程度上讲是有意义的，例如图片的底层边缘特征与特征在图中的具体位置无关。但是在一些场景中是无意的，比如输入的图片是人脸，眼睛和头发位于不同的位置，希望在不同的位置学到不同的特征 。请注意权重只是对于同一深度切片的神经元是共享的，在卷积层，通常采用多组卷积核提取不同特征，即对应不同深度切片的特征，不同深度切片的神经元权重是不共享。另外，偏重对同一深度切片的所有神经元都是共享的。

通过介绍卷积计算过程及其特性，可以看出卷积是线性操作，并具有平移不变性（shift-invariant），平移不变性即在图像每个位置执行相同的操作。卷积层的局部连接和权重共享使得需要学习的参数大大减小，这样也有利于训练较大卷积神经网络。

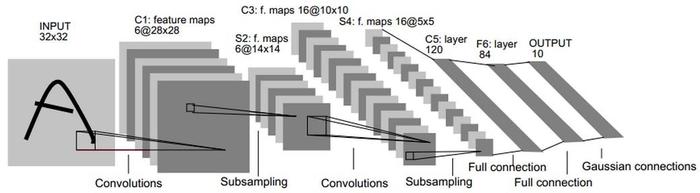
整体计算过程如下（与上图中的数据不同，但是计算过程相同）：

2、池化层

池化是非线性下采样的一种形式，主要作用是通过减少网络的参数来减小计算量，并且能够在一定程度上控制过拟合。通常在卷积层的后面会加上一个池化层。池化包括最大池化、平均池化等。其中最大池化是用不重叠的矩形框将输入层分成不同的区域，对于每个矩形框的数取最大值作为输出层，如上图所示。

三、Lenet-5

LeNet5 这个网络虽然很小，但是它包含了[深度学习](https://cuijiahua.com/blog/tag/%e6%b7%b1%e5%ba%a6%e5%ad%a6%e4%b9%a0/" \o "查看与 深度学习 相关的文章" \t "https://cuijiahua.com/blog/2018/01/_blank)的基本模块：卷积层，池化层，全链接层。是其他[深度学习](https://cuijiahua.com/blog/tag/%e6%b7%b1%e5%ba%a6%e5%ad%a6%e4%b9%a0/" \o "查看与 深度学习 相关的文章" \t "https://cuijiahua.com/blog/2018/01/_blank)模型的基础， 这里我们对LeNet5进行深入分析。同时，通过实例分析，加深对与卷积层和池化层的理解。

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_4.jpg)

LeNet-5共有7层，不包含输入，每层都包含可训练参数；每个层有多个Feature Map，每个FeatureMap通过一种卷积滤波器提取输入的一种特征，然后每个FeatureMap有多个神经元。

各层参数详解：

1、INPUT层-输入层

首先是数据 INPUT 层，输入图像的尺寸统一归一化为32\*32。

注意：本层不算LeNet-5的网络结构，传统上，不将输入层视为网络层次结构之一。

2、C1层-卷积层

输入图片：32\*32

卷积核大小：5\*5

卷积核种类：6

输出featuremap大小：28\*28 （32-5+1）=28

神经元数量：28\*28\*6

可训练参数：（5\*5+1) \* 6（每个滤波器5\*5=25个unit参数和一个bias参数，一共6个滤波器）

连接数：（5\*5+1）\*6\*28\*28=122304

详细说明：对输入图像进行第一次卷积运算（使用 6 个大小为 5\*5 的卷积核），得到6个C1特征图（6个大小为28\*28的 feature maps, 32-5+1=28）。我们再来看看需要多少个参数，卷积核的大小为5\*5，总共就有6\*（5\*5+1）=156个参数，其中+1是表示一个核有一个bias。对于卷积层C1，C1内的每个像素都与输入图像中的5\*5个像素和1个bias有连接，所以总共有156\*28\*28=122304个连接（connection）。有122304个连接，但是我们只需要学习156个参数，主要是通过权值共享实现的。

3、S2层-池化层（下采样层）

输入：28\*28

采样区域：2\*2

采样方式：4个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过sigmoid

采样种类：6

输出featureMap大小：14\*14（28/2）

神经元数量：14\*14\*6

连接数：（2\*2+1）\*6\*14\*14

S2中每个特征图的大小是C1中特征图大小的1/4。

详细说明：第一次卷积之后紧接着就是池化运算，使用 2\*2核 进行池化，于是得到了S2，6个14\*14的 特征图（28/2=14）。S2这个pooling层是对C1中的2\*2区域内的像素求和乘以一个权值系数再加上一个偏置，然后将这个结果再做一次映射。同时有5x14x14x6=5880个连接。

4、C3层-卷积层

输入：S2中所有6个或者几个特征map组合

卷积核大小：5\*5

卷积核种类：16

输出featureMap大小：10\*10 (14-5+1)=10

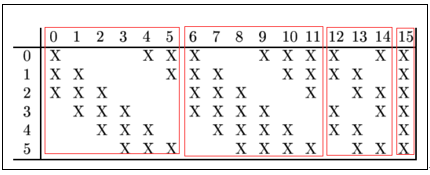
C3中的每个特征map是连接到S2中的所有6个或者几个特征map的，表示本层的特征map是上一层提取到的特征map的不同组合

存在的一个方式是：C3的前6个特征图以S2中3个相邻的特征图子集为输入。接下来6个特征图以S2中4个相邻特征图子集为输入。然后的3个以不相邻的4个特征图子集为输入。最后一个将S2中所有特征图为输入。

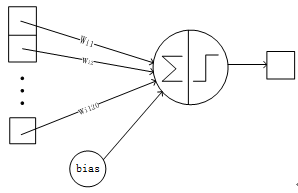
则：可训练参数：6\*(3\*5\*5+1)+6\*(4\*5\*5+1)+3\*(4\*5\*5+1)+1\*(6\*5\*5+1)=1516

连接数：10\*10\*1516=151600

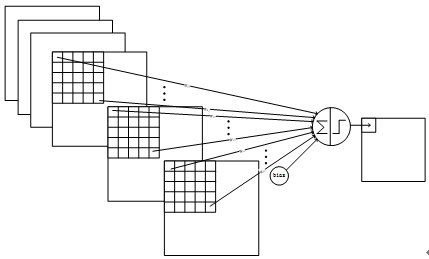
详细说明：第一次池化之后是第二次卷积，第二次卷积的输出是C3，16个10x10的特征图，卷积核大小是 5\*5. 我们知道S2 有6个 14\*14 的特征图，怎么从6 个特征图得到 16个特征图了？ 这里是通过对S2 的特征图特殊组合计算得到的16个特征图。具体如下：

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_5.png)

C3的前6个feature map（对应上图第一个红框的6列）与S2层相连的3个feature map相连接（上图第一个红框），后面6个feature map与S2层相连的4个feature map相连接（上图第二个红框），后面3个feature map与S2层部分不相连的4个feature map相连接，最后一个与S2层的所有feature map相连。卷积核大小依然为5\*5，所以总共有6\*(3\*5\*5+1)+6\*(4\*5\*5+1)+3\*(4\*5\*5+1)+1\*(6\*5\*5+1)=1516个参数。而图像大小为10\*10，所以共有151600个连接。

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_9.png)

C3与S2中前3个图相连的卷积结构如下图所示：

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_6.png)

上图对应的参数为 3\*5\*5+1，一共进行6次卷积得到6个特征图，所以有6\*（3\*5\*5+1）参数。 为什么采用上述这样的组合了？论文中说有两个原因：1）减少参数，2）这种不对称的组合连接的方式有利于提取多种组合特征。

5、S4层-池化层（下采样层）

输入：10\*10

采样区域：2\*2

采样方式：4个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过sigmoid

采样种类：16

输出featureMap大小：5\*5（10/2）

神经元数量：5\*5\*16=400

连接数：16\*（2\*2+1）\*5\*5=2000

S4中每个特征图的大小是C3中特征图大小的1/4

详细说明：S4是pooling层，窗口大小仍然是2\*2，共计16个feature map，C3层的16个10x10的图分别进行以2x2为单位的池化得到16个5x5的特征图。有5x5x5x16=2000个连接。连接的方式与S2层类似。

6、C5层-卷积层

输入：S4层的全部16个单元特征map（与s4全相连）

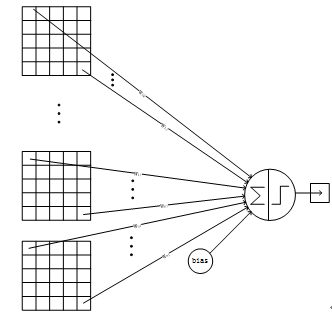
卷积核大小：5\*5

卷积核种类：120

输出featureMap大小：1\*1（5-5+1）

可训练参数/连接：120\*（16\*5\*5+1）=48120

详细说明：C5层是一个卷积层。由于S4层的16个图的大小为5x5，与卷积核的大小相同，所以卷积后形成的图的大小为1x1。这里形成120个卷积结果。每个都与上一层的16个图相连。所以共有(5x5x16+1)x120 = 48120个参数，同样有48120个连接。C5层的网络结构如下：

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_7.png)

7、F6层-全连接层

输入：c5 120维向量

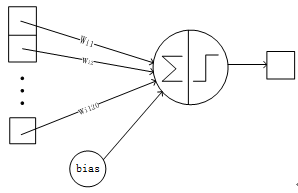
计算方式：计算输入向量和权重向量之间的点积，再加上一个偏置，结果通过sigmoid函数输出。

可训练参数:84\*(120+1)=10164

详细说明：6层是全连接层。F6层有84个节点，对应于一个7x12的比特图，-1表示白色，1表示黑色，这样每个符号的比特图的黑白色就对应于一个编码。该层的训练参数和连接数是(120 + 1)x84=10164。ASCII编码图如下：

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_8.png)

F6层的连接方式如下：

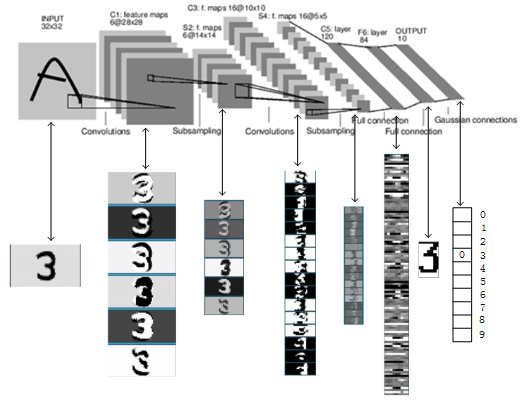
[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_9.png)

8、Output层-全连接层

Output层也是全连接层，共有10个节点，分别代表数字0到9，且如果节点i的值为0，则网络识别的结果是数字i。采用的是径向基函数（RBF）的网络连接方式。假设x是上一层的输入，y是RBF的输出，则RBF输出的计算方式是：

[IMG_266](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_10.png)

上式w\_ij 的值由i的比特图编码确定，i从0到9，j取值从0到7\*12-1。RBF输出的值越接近于0，则越接近于i，即越接近于i的ASCII编码图，表示当前网络输入的识别结果是字符i。该层有84x10=840个参数和连接。

[](https://cuijiahua.com/wp-content/uploads/2018/01/dl_3_11.png)

上图是LeNet-5识别数字3的过程。

四、总结

LeNet-5是一种用于手写体字符识别的非常高效的卷积神经网络。

卷积神经网络能够很好的利用图像的结构信息。

卷积层的参数较少，这也是由卷积层的主要特性即局部连接和共享权重所决定。

#### **2.3 FPGA 硬件加速技术 (大致举个例子分析下FPGA对于的硬件加速性能)**

#### **2.4 Vivado相关开发工具 (介绍下vivado的功能)**

Vivado设计套件，是FPGA厂商赛灵思公司2012年发布的集成设计环境。包括高度集成的设计环境和新一代从系统到IC级的工具，这些均建立在共享的可扩展数据模型和通用调试环境基础上。这也是一个基于AMBA AXI4 互联规范、IP-XACT IP封装元数据、工具命令语言(TCL)、Synopsys 系统约束(SDC) 以及其它有助于根据客户需求量身定制设计流程并符合业界标准的开放式环境。赛灵思构建的的Vivado 工具把各类可编程技术结合在一起，能够扩展多达1 亿个等效[ASIC](https://baike.baidu.com/item/ASIC/2014676" \t "https://baike.baidu.com/item/Vivado/_blank) 门的设计。

#### **2.5 本章小结**

**第三章**

3.1 交通标志识别系统设计简介

3.2 图像预处理设计的思路

3.3 深度神经网络的设计

3.4 FPGA 循环展开层次的分析

**第四章 系统的具体实现 (主要写硬件 各种仿真贴图 硬件设计图等) 30%**

4.1 整体系统的硬件设计与实现

4.2 预处理系统的硬件系统的设计与实现

4.3 神经网络硬件系统的设计与实现

4.4 理论系统性能分析

4.4 本章小结

**第五章 系统测试与验证 (各种测试性能 波形图 准确率 对比 P图) 10%**

5.1 系统测试环境搭建

5.2 预处理系统子系统测试

5.3 深度神经网络子系统测试

5.4 整体系统测试 与性能测试 和能耗对比

**第六章 总结与展望 (5%)**

6.1 工作总结

6.2 工作展望