## 利用TensorFlow构建CNN网络进行手写数字识别

目录

[利用TensorFlow构建CNN网络进行手写数字识别 1](#_Toc124175573)

[摘要 2](#_Toc124175574)

[1.基础知识 2](#_Toc124175575)

[一、卷积神经网络 2](#_Toc124175576)

[二、CNN基本原理 2](#_Toc124175577)

[三、 Tensorflow 3](#_Toc124175578)

[2. 代码实现 4](#_Toc124175579)

[MNIST 数据集读取： 4](#_Toc124175580)

[数据可视化： 4](#_Toc124175581)

[数据预处理： 5](#_Toc124175582)

[构建CNN网络 6](#_Toc124175583)

[CNN网络编译与训练 7](#_Toc124175584)

[CNN模型验证： 8](#_Toc124175585)

[CNN模型保存 8](#_Toc124175586)

[加载CNN保存模型： 8](#_Toc124175587)

[预测结果可视化： 9](#_Toc124175588)

## 摘要

手写数字识别是常见的图像识别任务，计算机通过手写体图片来识别图片中的字。与印刷字体不同的是，不同人的手写体风格迥异，大小不一，造成了计算机对手写识别任务的困难，我们小组决定通过应用深度学习和TensorFlow工具对MNIST手写数据集进行训练并建模，CNN网络对于图像的识别和分类效果很好，所以讨论后决定采用卷积神经网络实现。

## 1.基础知识

### 一、卷积神经网络

   卷积神经网络CNN是多层神经网络的一个变种，传统的多层神经网络，当隐层数变多时，节点数目过多时就会造成参数个数过多，训练难度极大。它充分利用了图片中相邻区域的信息，通过稀疏连接和共享权值的方式大大减少参数矩阵的规模，也提高了训练速度。

为了减少参数的个数，提出了以下三个概念：

   （1）稀疏连接（sparse connectivity ）：通过对数据中的局部区域进行建模，发现局部的一些特性。第m层结点的输入为第m-1层神经元的一个子集，被选择的子集的大小称为感受野。每组感受野中，其参数是相互共享的，也就是权值共享。

   （2）共享权值（Shared weights)：简化计算过程，使得需要优化的参数减少。如感受野大小为3时，此时包含三个参数w1,w2,w3，这样极大缩减需要学习的参数数量。

   （3）池化（Pooling）：也就是子采样，目的是解决图像中的平移不变性，达到要识别的内容和其位置无关。采用最大池化max-pooling，将输入图像划分为一系列不重叠的正方形区域，对每个子区域输出其中的最大值，通过消除非最大值，进一步降低计算量。

### 二、CNN基本原理

1. 卷积滤波

   卷积其实是图像处理中一种常用的线性滤波方法，使用卷积可是降噪，蜕化等。是对图片中每一个像素点，计算它的邻域像素和卷积核滤波器矩阵的对应位置元素的乘积，然后将乘积累加。

   卷积核：是一个权值矩阵，表示如何处理单个像素与其周围像素之间的关系。卷积核每个元素越接近，则越模糊降噪；相差大，则锐化。一般比较小，因为和计算量有关。每个卷积核就是一个特征提取器，n个就提取n个特征。

1. 卷积层

   将卷积操作和反向传播结合，诞生卷积神经网络，通过大量图片让程序自己训练学习卷积核参数就可以。

   卷积层节点和全连接层节点有三个不同：

   1、局部感知域（稀疏连接）：对于每个计算单元，只考虑其像素位置附近的输入，也就是只取上一层输入的一个子集。这些局部信息组合形成图像。10X10的卷积核进行卷积操作的话，每个节点100个输入，对应100个参数。而且同层的卷积核共享权值，大大减少了参数。

   2、权值共享：对图片进行卷积时，会让卷积核逐一划过每个像素，这样处理每个像素的参数都是相同的。

   3、多核卷积：以多核的方式来充分提取特征。每个卷积核生成一幅新图像，这些图像称为特征图，这是比像素更高级的特征。可以理解为经过滤波后的不同通道。

1. 池化

   得到了更高级别的特征后，可以交给分类器进行训练分类了。但是在这之前，需要池化操作进行降维。

   池化：将图像按窗口大小划分为不重叠的区域，然后对每一个区域内的元素进行聚合。一般采用2x2窗口大小，聚合方法：取各个元素最大值；取平均值。采用2x2的池化操作，处理完的图像长和宽都是原图的一般，变为1/4。既达到降维的目的，也使得特征提取有了“平移不变性（有几个像素的位移的情况下，依然可以获得稳定的特征集合）”。我们只关心那些特征可以代表物体。

### Tensorflow

1. 计算图（也叫数据流图）

   Tensorflow是基于计算图的数值计算系统。保证了各种算法实现的灵活性。

   计算图的执行：看做数据tensor按照图的拓扑顺序，从输入节点逐步流过所有中间节点，最后流出输出节点的过程。

   图节点：operations，图中的节点代表数字计算操作的算子。它是参与计算的基本单位，每个算子对应一种运算，tensorflow内置了很多算子，各种数值，数组，矩阵，状态操作，神经网络的各个操作，存档啊，队列，同步啊，控制流等。

   连接边：tensor，节点之间的连接边代表参与计算的高维数组数据。它代表多维数组，对应的是神经网络的高维参数矩阵。tensor可以是任意维度，每维可以是任意长度，其中的元素可以是任意内置类型。4维tensor可以表示一个mini-batch的图片，四个维度分别是批大小，像素行数，像素列数，通道数。

   变量：Variable，来存储参数值，表示了图的状态。可以与tensor一样参与各种运算，区别是tensor的值每次计算后丢弃，Variable的值会不断更新。

1. Session会话

   Session：是驱动Tensorflow系统执行计算交互的入口。负责完成多计算设备或集群分布式的节点布置和数据传输节点的添加，将子图分配给相应的执行器单元。

1. Tensorflow系统架构

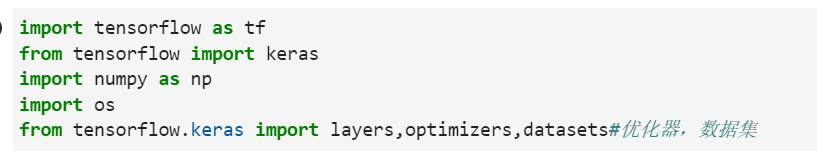
   Tensorflow是拥有“client—>master—>worker”架构的分布式系统。

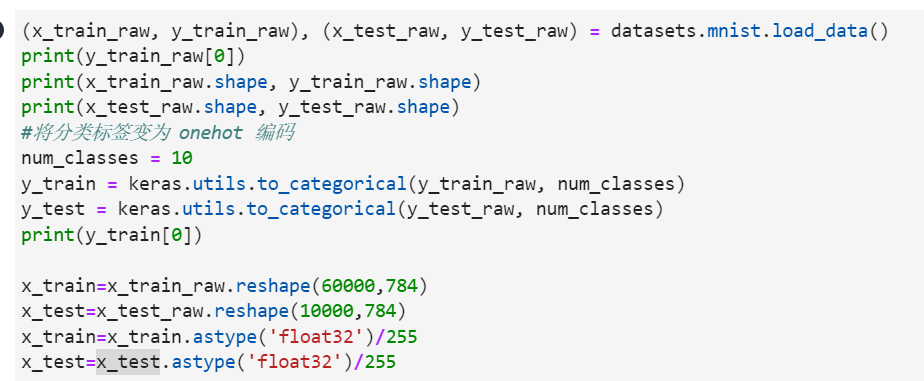
   一般的执行流程：客户端通过会话tf.Session接口与master进行通信，向master提交触发执行请求，master将执行任务分配到一个或多个worker进程上，执行的结果通过master返回客户端。可见worker是最终负责计算的角色，每一个worker进程都会管理和使用计算机上的 硬件设备资源来处理计算子图的运算过程。

## 代码实现

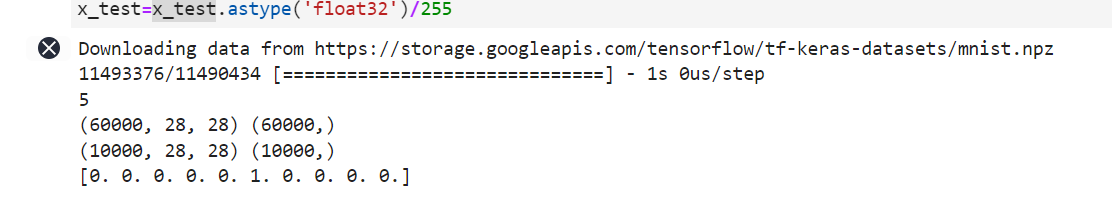
### MNIST 数据集读取：

从tensorflow直接读取数据集，联网下载解压





输出结果：



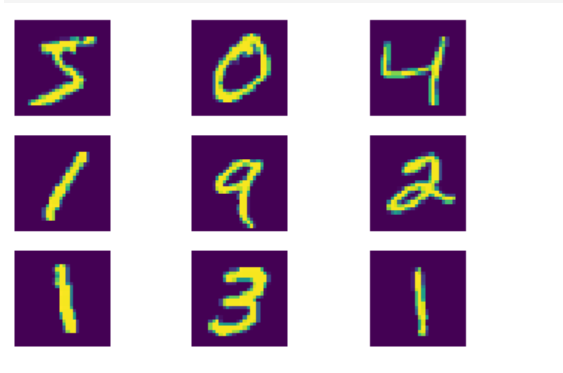
在mnist数据集中，images是一个形状为[60000,28,28]的张量，第一个维度数字用来索引图片，第二，三维度数字用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，介于0，255之间

标签数据是“one-hot vectors”一个one-hot向量除了某一位数字是1之外，其余各维度数字都是0，如标签1可以表示为([0,1,0,0,0,0,0,0,0,0,0])，因此labels是一个[60000,10]的数字矩阵。

### 数据可视化：



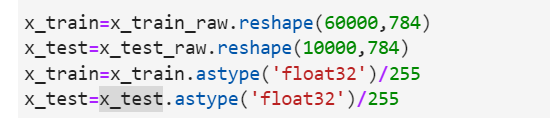
**输出结果：**

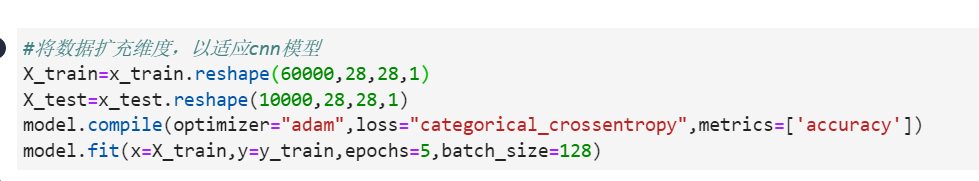


### 数据预处理：

1.将图像整理成向量，将28\*28的图像展开成784\*1的向量

2现在像素点的动态范围为0-255。处理图形像素值时，将图像像素点归一化到0-1的范围内





### 构建CNN网络

传统方法构建CNN网络，代码量较多，所以我们尝试使用高级API构建网络，以简化构建网络的过程。

**2.1 卷积函数tf.nn.conv2d()**

tf.nn.conv2d(input,filter,strides,padding,use\_cudnn\_on\_gpu=None,name=None)

* input: 指定需要做卷积的输入图像，它要求是一个Tensor，具 [batch,in\_height,in\_width,in\_channels]这样的形状(shape)，具体含义是"训练时一个batch的图片数量，图片高度，图片宽度，图片通道数"，注意这是一个四维的Tensor，要求类型为float32或者float64.
* filter：相当于CNN中的卷积核，它要求是一个Tensor，具有[filter\_height,filter\_width,in\_channels,out\_channels] 这样的shape，具体含义是"卷积核的高度，卷积核的宽度，图像通道数，滤波器个数"，要求类型与参数input相同。有一个地方需要注意，第三维in\_channels，就是参数input中的第四维。
* strides：卷积时在图像每一维的步长，这是一个一维的向量，长度为4，与输入input对应，一般值为[1,x,x,1],x取步长。
* padding：定义元素边框与元素内容之间的空间。string类型的量，只能是"SAME"和“VALID”其中之一，这个值决定了不同的卷积方式。SAME表示填充，VALID则表示不填充。use\_cudnn\_on\_gpu: bool类型，是否使用cudnn加速，默认是True.
* name：指定名字

**2.2 池化函数**

tf.nn.max\_pool()和tf.nn.avg\_pool()

tf.nn.max\_pool(input,ksize,strides,padding,name=None)

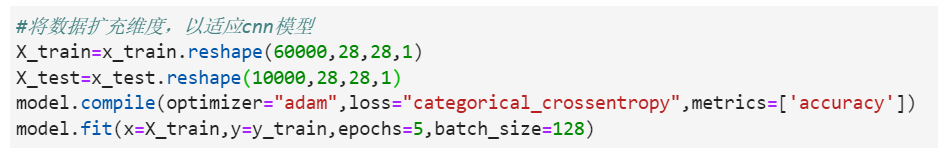
tf.nn.avg\_pooll(input,ksize,strides,padding,name=None)

* input：需要池化的输入，一般池化层接在卷积层后面，所以输入通常是feature map,依然是[batch,height,width,channels]这样的shape。
* ksize：池化窗口的大小，取一个思维向量，一般是[1,height,width,1]，因为我们不想在batch和channels上做池化，所以这两个维度设为1.
* strides：和卷积参数含义类似，窗口在每一个维度上滑动的步长，一般也是[1,stride,stride,1]。
* padding：和卷积参数含义一样，也是"VALID"或者"SAME"。

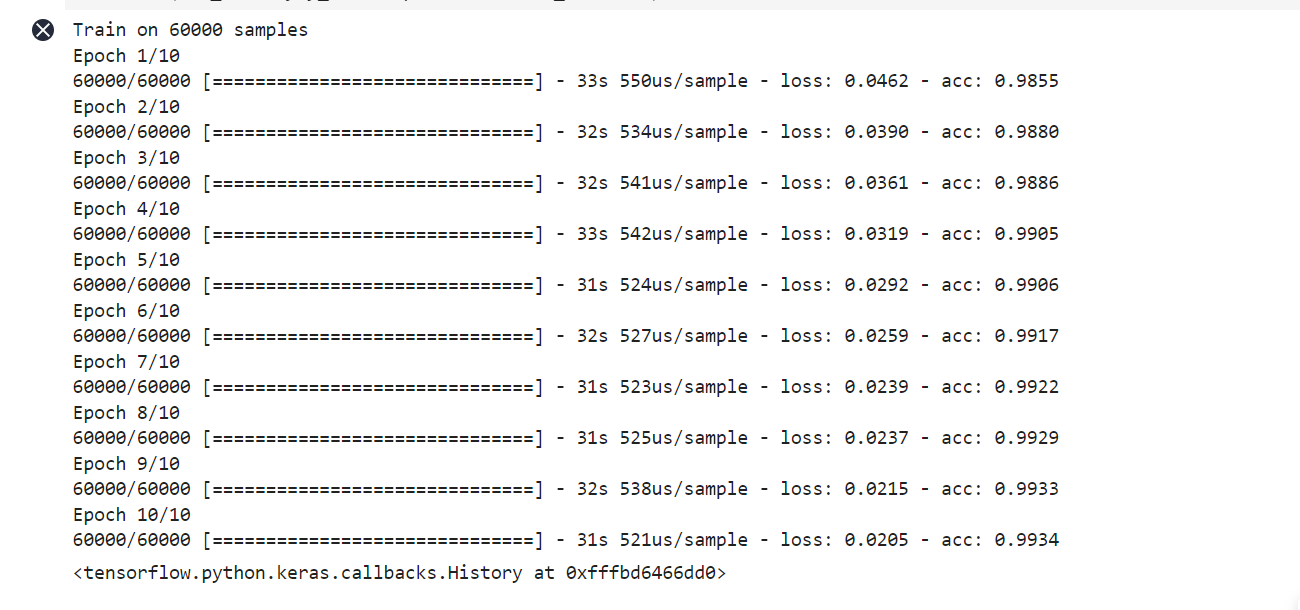


以上网络中，我们用keras.layers添加了两个卷积池化层，之后又添加了dropout层，防止过拟合，最后添加了两层全连接层

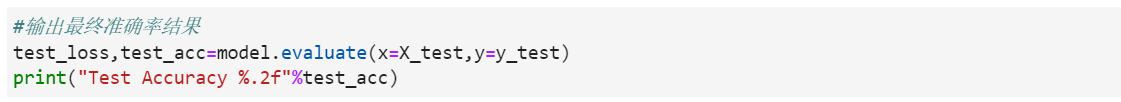
### CNN网络编译与训练



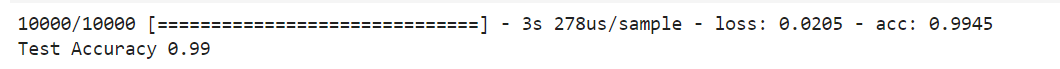
输出：



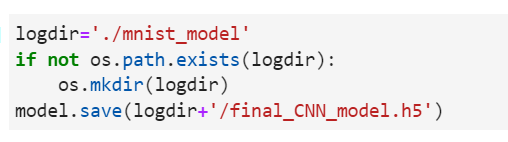
### CNN模型验证：



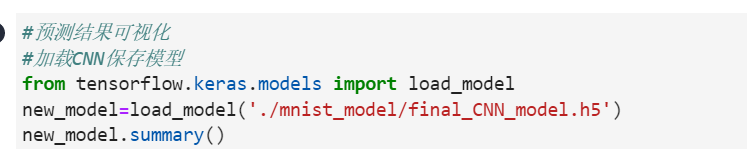
输出：



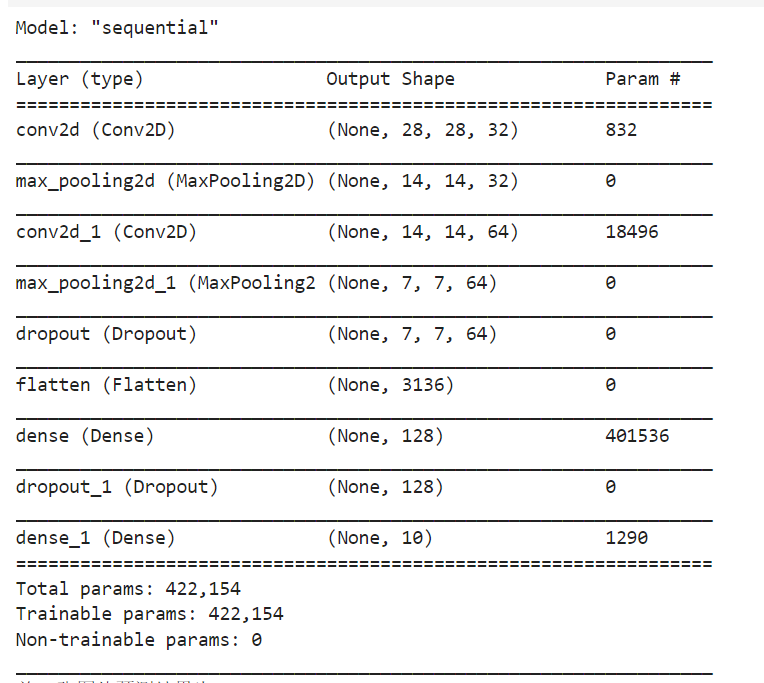
### CNN模型保存



### 加载CNN保存模型：



输出：



### 预测结果可视化：

输出：

