**一、前言**

本文代码是我在学习TensorFlow时写的第一个代码，主要是根据LeNet模型编写，但是自距此模型问世以来，深度学习领域发生了很大变化，所以其中某些方法和参数已经有所改变，目前此模型通过在GPU上训练，精确度可以达到98%左右，下面我将详细介绍Tensorflow中的实现，本教程适合TensorFlow的初学者，主要有以下特点：

* 1. 使用Mnist数据集
  2. 测试精度可达98%
  3. 可视化输出
  4. 兼具卷积层、pooling（池化）层、dropout层、全连接层、softmax layer
  5. 提供两个实现此模型原始代码

**代码实现：**

**代码地址：<https://github.com/WzsGo/LeNet_mnist_TensorFlow.git>**

这个仓库中有两个文件，基础版本（v1.0） 和 提高版本（v2.0）

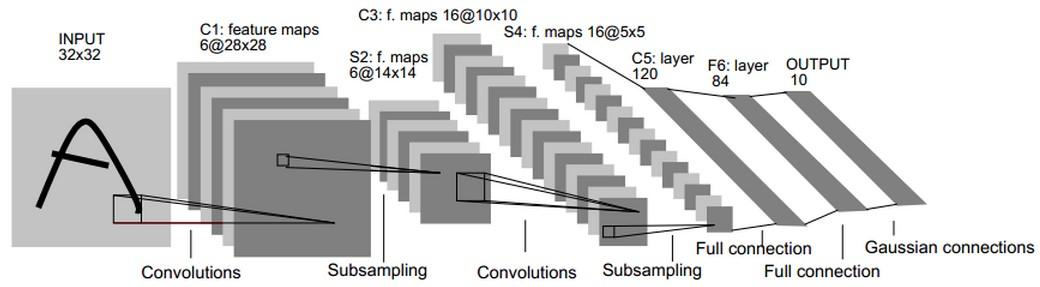
1. **python LeNet\_mnist\_v1.0.py**

基础版本，使用tf.nn.\*\*\*函数实现，注重构造网络中的细节实现，容易理解CNN的运行原理，从细节理解CNN结构。

1. **python LeNet\_mnist\_v2.0.py**

提高版本，使用tf.layer.\*\*\*函数实现，使用tf框架提供的集成函数，特点是构造方便，宏观上更好理解，但是没有展现细节。

**二、模型结构**



**三、代码实现简介**

**(1)定义的超参数：dropout层的保留概率/每次提取图片数目/迭代次数**

train\_keep\_prop = 0.5/batch\_size = 100/epcoh = 1000

可以更改这些参数，以改善训练结果。

**(2)模型结构**

**input - 输入数据：Mnist**

数据集被分成两部分：60000 行的训练数据集（mnist.train）和10000行的测试数据集（mnist.test），每张图片是 28\*28\*1，经过reshape后，维度格式为：[28,28,1]

**conv1 - 卷积层：**

卷积层：kenel：5\*5\*32，strides = 1,padding = SAME -->> [28,28,32]

池化层：ksize: 2\*2 ,strides = 2 -->> [14,14,32]

drop层：keep\_prop = train\_keep\_prop -->> [14,14,32]

**conv2 - 卷积层：**

卷积层：kenel：5\*5\*64，strides = 1,padding = SAME -->> [14,14,64]

池化层：ksize: 2\*2 ,strides = 2 -->> [7,7,64]

drop层：keep\_prop = train\_keep\_prop -->> [7,7,64]

**Flaten层：**

将[7,7,64]矩阵形式 -->> [7\*7\*64]向量形式

**Fucn1 - 全连接层：**

权重：[7\*7\*64,1024]

**Fucn2 - 全连接层：**

权重：[1024,10]

**Softmax层：**

输出one-hot向量，对应每一类的概率。

**四、代码实现**

代码地址：<https://github.com/WzsGo/LeNet_mnist_TensorFlow.git>

这个仓库中有两个文件，基础版本（v1.0） 和 提高版本（v2.0）

1. **python LeNet\_mnist\_v1.0.py**

基础版本，使用tf.nn.\*\*\*函数实现，注重构造网络中的细节实现，容易理解CNN的运行原理，从细节理解CNN结构。

1. **python LeNet\_mnist\_v2.0.py**

提高版本，使用tf.layer.\*\*\*函数实现，使用tf框架提供的集成函数，特点是构造方便，宏观上更好理解，但是没有展现细节。

**五、执行程序**

执行命令： python LeNet\_mnist\_v1.0.py

执行命令：python LeNet\_mnist\_v2.0.py

**六、训练结果**

