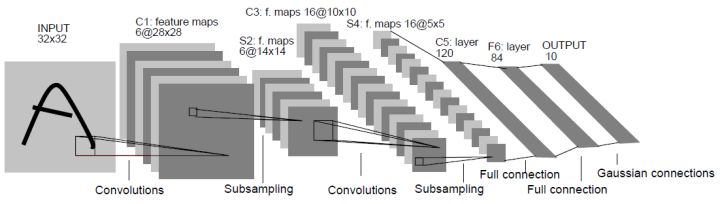
Lenet-5是卷积神经网络中很适合入门的网络结构。它在1998年由Yann LeCuu等人在论文”Gradient-Based Learning Applied to Document Recognition“中提出，用于解决mnist数据集的字符识别问题。由于网络结构比较简单，所以可以在自己的笔记本电脑中快速运行。

本文参考论文以及[Deep Learning | Coursera](http://link.zhihu.com/?target=https%3A//www.coursera.org/specializations/deep-learning)相关讲解，主要介绍Lenet的基本结构和其keras实现。

**1.网络结构**



Lenet-5除输入层外由7层网络构成：

**(1).卷积层Conv1：**

接受32\*32\*1的图像输入，本层包含6个大小为5\*5，步长为1\*1的卷积核，padding类型为'valid'(即没有padding)。输出神经元：28\*28\*6

**(2).池化（pooling）层:**

对上一层的输出做2\*2的max pooling，输出神经元形状为14\*14\*6。

**(3).卷积层Conv2：**

接受14\*14\*6的输入，本层有16个大小为5\*5，步长为1\*1的卷积核，同第一个卷积层，本层的padding类型为'valid。输出神经元为10\*10\*16。

**(4).池化层：**

对上层输出做2\*2的max pooling，输出神经元形状为5\*5\*16。

**(5).全连接层fc1：**

本层将上层的5\*5\*16神经元展开作为输入，本层包含120个神经元。

**(6).全连接层fc2：**

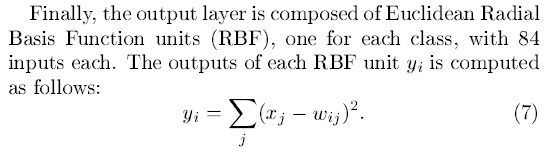
本层包含84个神经元。

**(7).输出层：**

本层包含10个神经元，分别代表数字0到9。

**(8).激活函数**

前7层采用论文中使用的tanh激活函数，输出层论文采用的是Guassian Connection，



在实践的代码中我用的是现在采用的是softmax 函数。

**2.代码实现**

数据集来源于kaggle（关于kaggle的入门介绍 [Kaggle入门](https://zhuanlan.zhihu.com/p/24883666)）中[Digit Recognizer | Kaggle](http://link.zhihu.com/?target=https%3A//www.kaggle.com/c/digit-recognizer) 数字识别练习中的数据集。

首先导入必要的python库：

import numpy as np

from keras.layers import Input,Add,Conv2D,Dense,ZeroPadding2D,Activation,MaxPooling2D,Flatten

from keras.models import Model

from keras.utils import to\_categorical

加载数据集：

我将数据集保存在data目录下。

train.csv文件中共有42000个样本，将其分为40000个训练集和2000个验证集。 test.csv文件中有42000个样本。

train=np.loadtxt('data/train.csv',delimiter=',',skiprows=1)

data\_train=train[:40000]

data\_val=train[40000:]

data\_test=np.loadtxt('data/test.csv',delimiter=',',skiprows=1)

查看各数据集的样本数量：

print(train.shape)

print(data\_train.shape)

print(data\_val.shape)

print(data\_test.shape)

输出为：



定义Lenet-5模型：

def lenet\_5(input\_shape=(32,32,1),classes=10):

X\_input=Input(input\_shape)

X=ZeroPadding2D((1,1))(X\_input)

X=Conv2D(6,(5,5),strides=(1,1),padding='valid',name='conv1')(X)

X=Activation('tanh')(X)

X=MaxPooling2D((2,2),strides=(2,2))(X)

X=Conv2D(6,(5,5),strides=(1,1),padding='valid',name='conv2')(X)

X=Activation('tanh')(X)

X=MaxPooling2D((2,2),strides=(2,2))(X)

X=Flatten()(X)

X=Dense(120,activation='tanh',name='fc1')(X)

X=Dense(84,activation='tanh',name='fc2')(X)

X=Dense(10,activation='softmax')(X)

model=Model(inputs=X\_input,outputs=X,name='lenet\_5')

return model

使用模型：

model=lenet\_5(input\_shape=(28,28,1),classes=10)

model.compile(optimizer='sgd',loss='categorical\_crossentropy',metrics=['accuracy'])

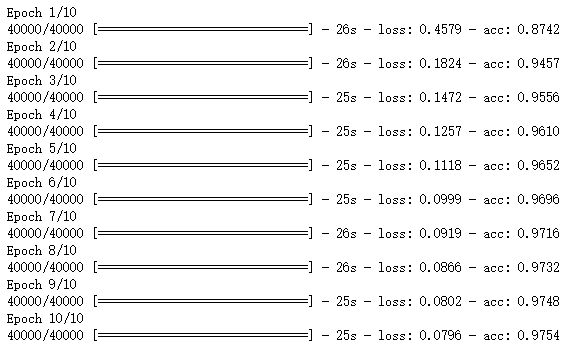
开始训练：

X\_train=data\_train[:,1:].reshape((data\_train.shape[0],28,28,1))

Y\_train=to\_categorical(data\_train[:,0])

model.fit(X\_train,Y\_train,epochs=10,batch\_size=16)

输出每个epoch后的准确率：



在验证集检验模型的准确率：

X\_val=data\_val[:,1:].reshape((data\_val.shape[0],28,28,1))

Y\_val=to\_categorical(data\_val[:,0])

preds=model.evaluate(X\_val,Y\_val)

print("Validation loss="+str(preds[0]))

print("Validation accuracy="+str(preds[1]))

输出验证集的loss和准确率：

https://pic4.zhimg.com/80/v2-33ec1d6518b64ff4e48401733710d3cb_hd.jpg

得到的准确率为97%

对测试集的样本预测：

X\_test=data\_test.reshape((data\_test.shape[0],28,28,1))

predicted=np.argmax(model.predict(X\_test),axis=1)

with open('data/submission.csv','w') as f:

f.write('ImageId,Label\n')

for i in range(len(predicted)):

f.write(str(i+1)+','+str(predicted[i])+'\n')

运行代码后，将在data的文件夹下获得一个submission.csv的文件，将其提交至kaggle，会得到大概为97%的准确率。

97%对于mnist数据集来说并不算很高的准确率。tensorflow官网给出的模型[https://www.tensorflow.org/get\_started/mnist/pros](http://link.zhihu.com/?target=https%3A//www.tensorflow.org/get_started/mnist/pros) 可以达到99.2%，如果对模型参数做更细致的微调，可以达到更高的准确率。对模型的优化将是以后要讨论的内容。

lenet-5是个比较基础的模型，相当于卷积神经网络模型中的”hello world“，想要达到更好的效果，可以关注更复杂的模型如vgg16，resnet50，resnet101等模型，以及参数优化，这将是本专栏计划以后要写的内容。