博客<https://blog.csdn.net/fengbingchun/article/details/50529500>

二、卷积神经网络结构

卷积神经网络整体架构：卷积神经网络是一种多层监督学习网络。

卷积神经网络结构包括：卷积层、降采样层、全连接层。每一层有多个特征图，每个特征图通过一种卷积滤波器提取输入的一种特制，每个特征有多个神经元。

卷积层：通过卷积运算，可以使原始信号特征增强，并且降低噪音。

降采样层：使用降采样的原因是，根据图像局部相关性的原理，对图像进行了子采样，可以减少计算量，同事保持图像旋转不变性。

全连接层：采用softmax全连接，得到的激活值即卷积神经网络提取到的图片特征，

基于matlab的卷积神经网络（CNN）讲解及代码

<https://blog.csdn.net/xiongchao99/article/details/78843154>

1. 经典反向传播公式详细推导

<https://blog.csdn.net/walegahaha/article/details/51867904>

1. 卷积神经网络（CNN）反向传播算法公式详细推导

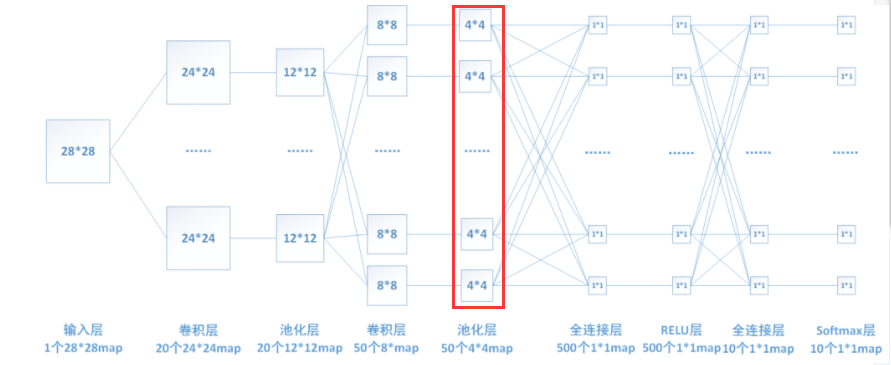
<https://blog.csdn.net/walegahaha/article/details/51945421>

网友matlab实现bp神经网络

<https://blog.csdn.net/weiwei9363/article/details/72472126>

对于全连接层的理解

<https://blog.csdn.net/gyh_420/article/details/78569225>

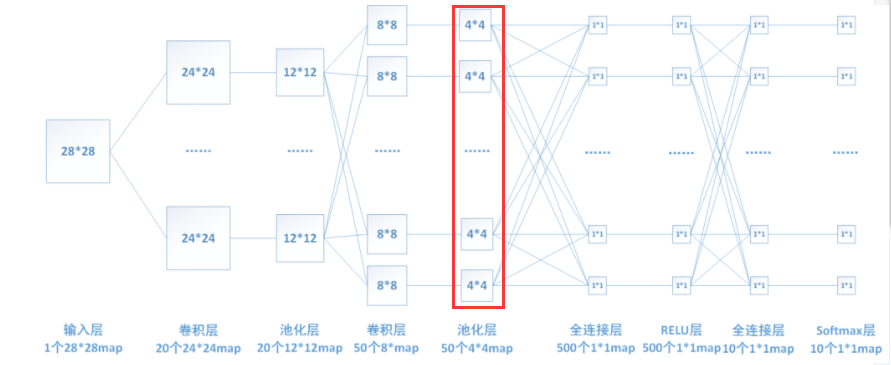


ResNet练习<https://blog.csdn.net/liangyihuai/article/details/79140481>

包含数据集合源代码

**深度学习的几个疑问？** 大熊 2018/6/19

1. CNN的全连接层，输入层是一个特征图，这个特征图是一个二维的，是通过将所有列抽出合并成一列作为全连接层的输入的吗？



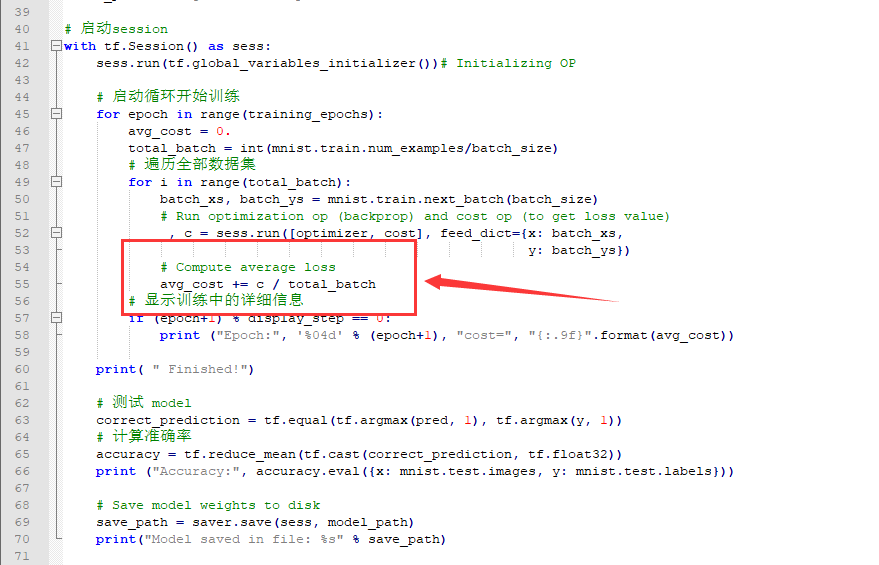
就是这里的特征图是怎么输入到全连接层的？

1. CNN在训练的时候，每次训练都是选一个batch进行训练，然后进行权重更新？这个batch的loss是如何计算的？

怎么计算这个batch的loss？

以下是本书代码5-2mnist的截取部分，这里的avg\_cost对应的是一个batch的loss，这

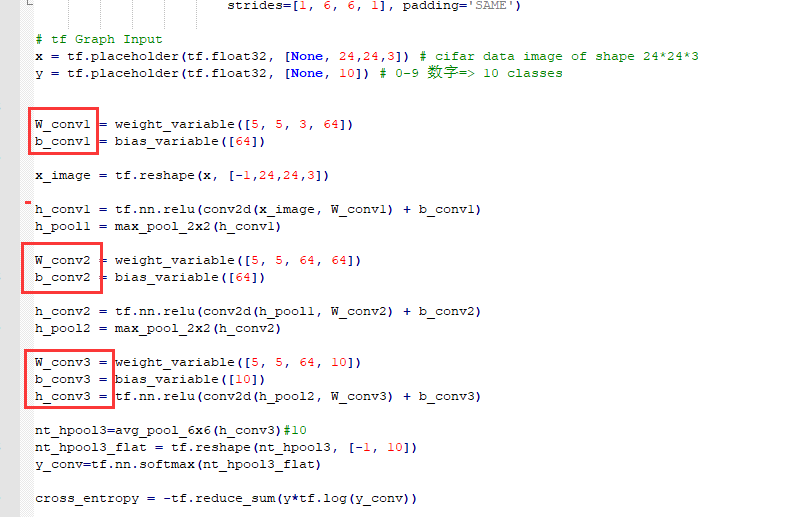
个loss是如何作用去更新W和b的？



1. 在tensorflow中，权重（W、b）的更新是怎么实现的？

以下是本书代码8-9的部分，这里的网络是作者自定义的，那么这里的W\_conv和b\_conv

是怎么进行更新的呢？



1. 深度学习keras框架中，模型的保存和加载

参考网友 <https://blog.csdn.net/cymy001/article/details/78647640>

以MNIST为例

**初次训练模型并保存**

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

#载入数据

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

#(60000, 28, 28)

print('x\_shape:', x\_train.shape )

#(60000, 28, 28)

print('y\_shape:', y\_train.shape)

#(60000, 28, 28)->(60000, 784)

x\_train = x\_train.reshape( x\_train.shape[0], -1 )/255.0

x\_test = x\_test.reshape( x\_test.shape[0], -1)/255.0

#换成 one hot格式

y\_train = np\_utils.to\_categorical( y\_train, num\_classes = 10 )

y\_test = np\_utils.to\_categorical( y\_test, num\_classes = 10 )

#创建模型，输出784个神经元，输出10个神经元

model = Sequential( [ Dense(units = 10, input\_dim = 784, bias\_initializer = 'one', activate = 'softmax')] )

#定义优化器

sgd = SGD(lr=0.2)

#定义优化器，loss function,训练过程中计算准确率

model.compile( optimizers = sgd, loss = 'mse', metrics =['accuracy'] )

#训练模型

model.fit( x\_train, y\_train, batch\_size = 64, epoch = 5 )

#评估模型

loss,accuracy = model.evaluate(x\_test, y\_test)

print("\ntest loss", loss)

print("accuracy", accuracy)

#保存模型

model.save( 'model.h5' ) #HDF5文件，pip install h5py

**载入模型，训练模型，保存模型**

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

from keras.models import load\_model

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 载入模型

model = load\_model('model.h5')

#评估模型

loss,accuracy = model.evaluate(x\_test, y\_test)

print("\ntest loss",loss)

print('accuracy',accuracy)

#训练模型

model.fit(x\_train, y\_train, batch\_size = 64, epoch = 2 )

#评估模型

loss,accuracy = model.evaluate( x\_test, y\_test )

print("\ntest loss", loss)

print("accuracy", accuracy)

#保存参数,载入参数

model.save\_weight('my\_model\_weight.h5')

model.load\_weight('my\_model\_weight.h5')

#保存网络结构，载入网络结构

from keras.models import model\_from\_json

json\_string = model.to\_json()

model = model\_from\_json( json\_string )

print( json\_string )

**POD使用已经训练好的模型进行识别**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 载入模型，使用模型 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

xTest = X\_data

yTest = Y\_data

modelName = 'model\_epoch\_7.h5'

trainedModel = load\_model( osj(model\_path,modelName) )

for i in range(len(xTest)):

img = xTest[23]

img\_new = cv2.resize(img, (224, 224) )

x = np.expand\_dims(img\_new, axis=0)

y\_pre = trainedModel.predict( x )

print(yTest[i])

show\_pic((xTest[i]).astype(np.uint8))