**机器学习实验**

**­­——图像分类系统**

**2120161081 赵冬迪（架构设计、部分代码实现、文档编写）**

**2120161064 夏光敏（部分代码实现、测试工作、文档编写）**

**2017年5月4日**

**目录**

[设计文档 1](#_Toc481748686)

[1. 问题描述 2](#_Toc481748687)

[2. 整体架构 2](#_Toc481748688)

[3. 算法描述 3](#_Toc481748689)

[3.1 基于特征工程的方法 3](#_Toc481748690)

[3.1.1 HOG特征 3](#_Toc481748691)

[3.1.2 SVM分类器 4](#_Toc481748692)

[3.1.3 Softmax分类器 4](#_Toc481748693)

[3.2 神经网络 5](#_Toc481748694)

[3.2.1 卷积神经网络 5](#_Toc481748695)

[3.2.2 深度残差网络 6](#_Toc481748696)

[参考文献 8](#_Toc481748697)

[测试文档 9](#_Toc481748698)

[1. 测试过程 10](#_Toc481748699)

[2. 测试数据集 10](#_Toc481748700)

[3. 测试结果 11](#_Toc481748701)

[3.1 基于特征工程的方法 11](#_Toc481748702)

[3.1.1 提取HOG特征 11](#_Toc481748703)

[3.1.2 线性SVM 12](#_Toc481748704)

[3.1.3 非线性SVM 12](#_Toc481748705)

[3.1.4 Softmax分类器 13](#_Toc481748706)

[3.2 神经网络 13](#_Toc481748707)

[3.2.1 卷积神经网络 13](#_Toc481748708)

[3.2.2 深度残差网络 15](#_Toc481748709)

[环境配置及相关说明 18](#_Toc481748710)

[1. 环境配置 19](#_Toc481748711)

[1.1 HOG+SVM/Softmax 19](#_Toc481748712)

[1.2 神经网络 19](#_Toc481748713)

[源代码说明 21](#_Toc481748714)

[1． HOG+SVM/Softmax 22](#_Toc481748715)

[1.1 HOG特征提取 22](#_Toc481748716)

[1.2 线性SVM 23](#_Toc481748717)

[1.3 非线性SVM 23](#_Toc481748718)

[1.4 Softmax分类器 24](#_Toc481748719)

[2． 神经网络 25](#_Toc481748720)

[2.1 卷积神经网络 25](#_Toc481748721)

[2.2 深度残差网络 26](#_Toc481748722)

# 设计文档

## 问题描述

图像分类/目标识别已经在计算机视觉领域成为最重要的研究问题之一。研究者们已经提出了一系列不同的局部描述符、特征编码模式和分类方法。

在本项目中，我们实现了我们自己的目标识别系统。

## 整体架构

在本项目中，主要采用了线性SVM分类器、非线性SVM分类器、Softmax分类器、卷积神经网络和深度残差网络五个模型对目标数据集进行目标识别与分类。其中，线性SVM、非线性SVM和Softmax分类器是传统的基于特征工程的分类方法。而卷积神经网络和深度残差网络是端到端的学习方法。因此，在采用传统的基于特征工程的方法时需要从原始的图片中提取特征。在本项目中，采用的是图片的方向梯度直方图（HOG）特征。系统整体架构如下图：

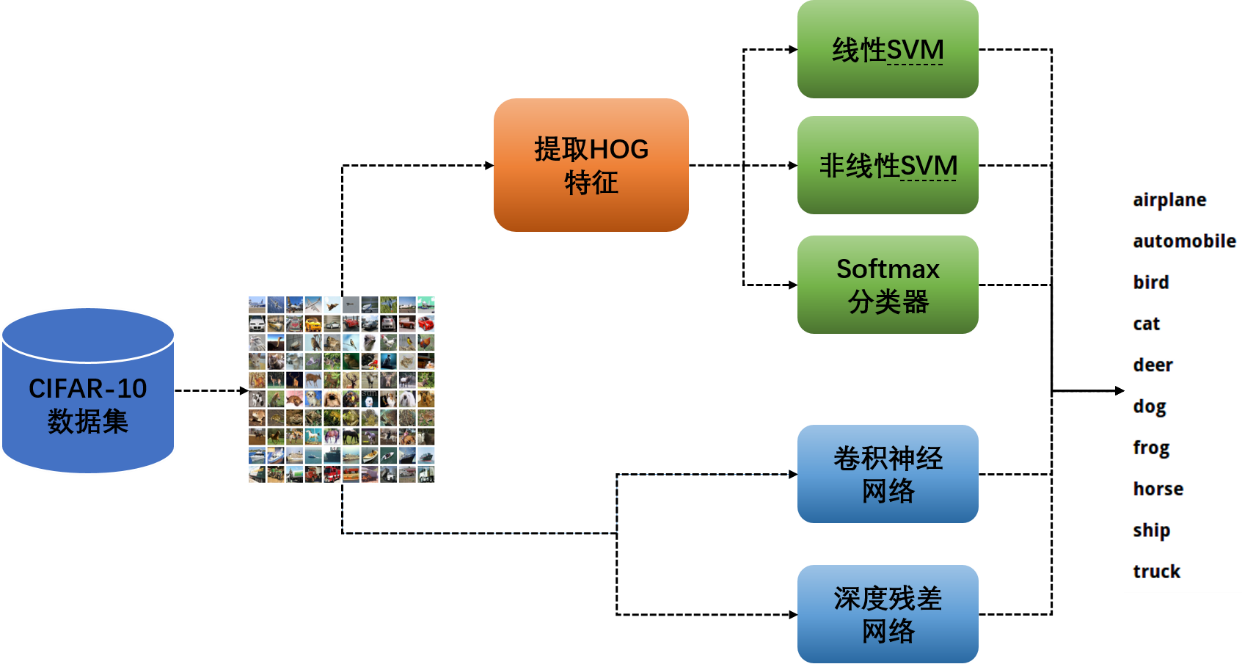


图2-1 系统整体架构图

## 算法描述

### 基于特征工程的方法

#### HOG特征

方向梯度直方图（HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。其主要思想是在一副图像中，局部目标的表象和形状能够被梯度或边缘的方向密度分布很好地描述。

提取图片的HOG特征，首相将图像分成小的连通区域，成为细胞单元。然后采集细胞单元中各像素点的梯度或边缘的方向直方图。最后把这些直方图组合起来就可以构成特征描述器。

与其他的特征描述方法相比，HOG有很多优点。首先，由于HOG是在图像的局部方格单元上操作，所以它对图像几何的和光学的形变都能保持很好的不 变性，这两种形变只会出现在更大的空间领域上。

算法流程如下：

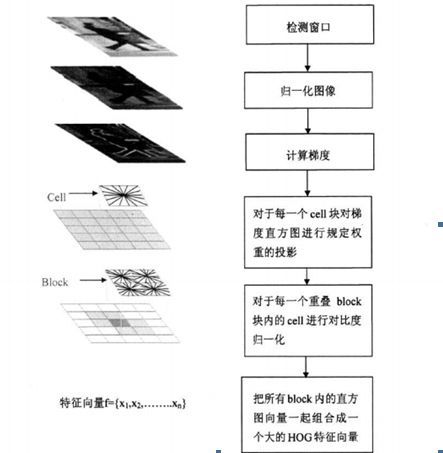


图3-1 HOG特征提取流程

#### SVM分类器

支持向量机（SVM）是一种而类分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机。支持向量机还包括核技巧，这使它成为是实质上的非线性分类器。支持向量机的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题。

在本项目中，分别采用线性支持向量机和非线性支持向量机对数据集进行分类，并比对准确率。

在多数计算机视觉运用中，我们需要的不仅仅是一个简单的SVM线性分类器，我们需要更加强大的工具来解决训练数据无法用一个超平面分割的情形。以目标识别为例，训练数据包含一组含有目标物体的图像和不包含目标物体的图像。这些训练数据超级复杂，以至于为每个样本找到一个合适的表达以让它们能够线性可分是非常困难的。线性SVM分类器的思想是在简单的线性可分SVM的基础上加入惩罚因子C，对于错误分类的样本，会对目标函数增加相应的惩罚，并优化改变后的目标函数。而非线性SVM的思想则是在线性SVM的基础上加入了核方法。核方法是比SVM更一般的机器学习方法，其主要思想是将数据通过核函数从低维空间映射到高维空间，是低维空间不可分的数据在高维空间中变得可分。

基于HOG特征的SVM分类方法已经在目标检测、行人检测等方面得到了广泛的应用。因此，将HOG+SVM的方法应用在图片分类中，作为本项目的对比实验。

#### Softmax分类器

Softmax分类器是Logistic分类器的扩展，适用于类别标签多余两类的情况。相比较于SVM，Softmax分类器采用了不同的损失函数——交叉熵损失函数。Softmax分类器要做的事情，就是要最小化预测类别的概率分布中两个概率分布的交叉熵。

在本项目中，设计了Softmax分类器对图片数据集进行分类，作为对比实验。

### 神经网络

#### 卷积神经网络

卷积神经网络（Constitutional Neural Networks, CNN）是在多层神经网络的基础上发展起来的针对图像分类和识别而特别设计的一种深度学习方法。多层神经网络包括一个输入层和一个输出层，中间有多个隐藏层。每一层有若干个神经元，相邻的两层之间的后一层的每一个神经元都分别与前一层的每一个神经元连接。在一般的识别问题中，输入层代表特征向量，输入层的每一个神经元代表一个特征值。

在图像识别问题中，输入层的每一个神经元可能代表一个像素的灰度值。但这种神经网络用于图像识别有几个问题，一是没有考虑图像的空间结构，识别性能会受到限制；二是每相邻两层的神经元都是全相连，参数太多，训练速度受到限制。

而卷积神经网络就可以解决这些问题。卷积神经网络使用了针对图像识别的特殊结构，可以快速训练。因为速度快，使得采用多层神经网络变得容易，而多层结构在识别准确率上又很大优势。

本实验参考了Alex Krizhevsky等人在2012年的文章­[3]中卷积神经网络的结构，共使用了7个隐藏层，前5个是卷积层（有些使用了max-pooling）。后两个是全连接层，输出层是有10个单元的softmax层，分别对应10个图像类别。

为了防止过拟合，采用了dropout的技术，将隐藏层中随机选取的一半神经元的输出设置为0，在加快训练速度的同时，可以防止模型过拟合的现象。

模型结构通过Tensorboard工具可视化如下：

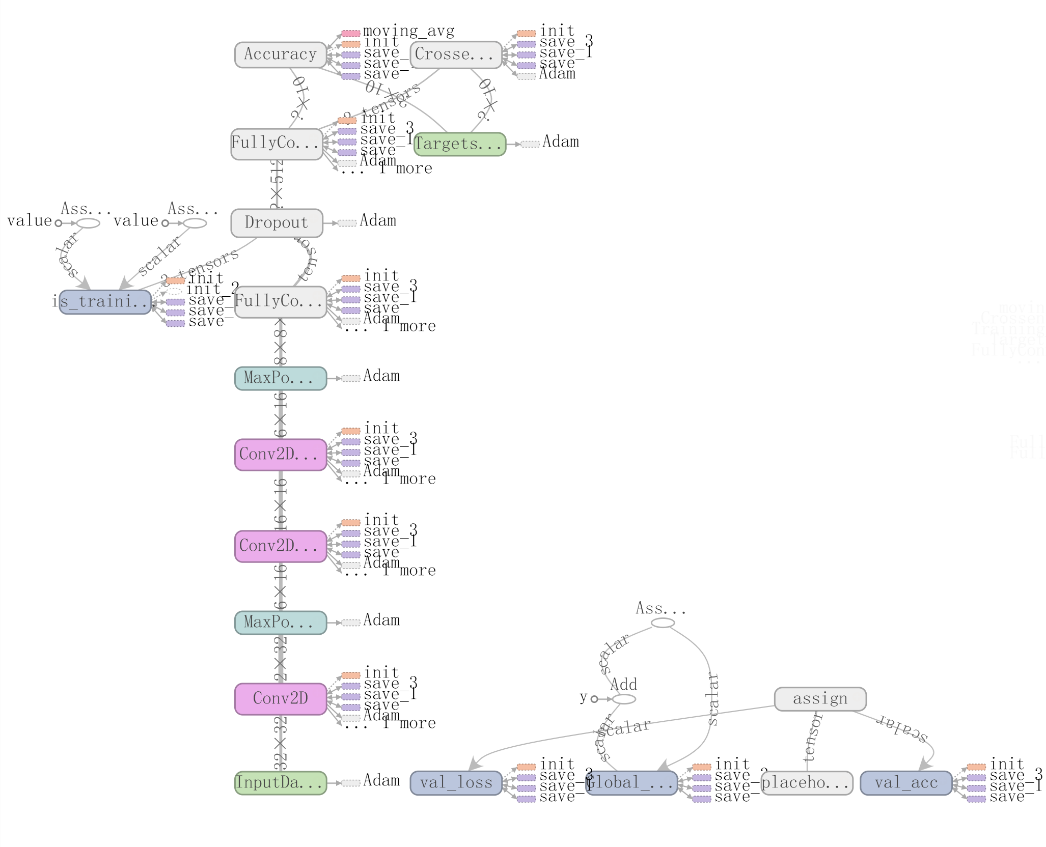


图3-2 卷积神经网络模型结构

#### 深度残差网络

深度残差网络自2015年提出以来，在众多比赛中表现优越。如论文中描述，深度残差网络的缘起在于一个“反常”的现象。在训练深度神经网络的时候，训练误差和测试误差都随着网络深度的增加而增加。这个问题显然不是由于过拟合问题导致的，因为训练误差同样在增加。

对于一个恒等变换的网络，随着网络深度的增加，网络的效果应该和浅层网络一样好。既然非线性网络层不能很好地学习恒等变换，作者的办法是直接拷贝低层，设计如下的模块：

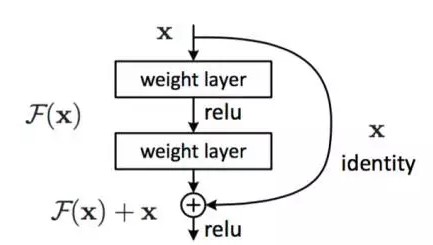


图3-3 残差块

实际上，恒等变换往往不是最优的，深度残差网络新增的非线性层的拟合目标不再是H(x)，而是F(x)=H(x)-x，这也是为什么网络被称为深度残差网络。新增的非线性层拟合目标是低层网络拟合的残差。

深度残差网络在多个计算机视觉任务中都取得了优异的成绩。

在本实验中设计了一个含有34个隐藏层的深度残差网络，包含5个残差块，最后一层是一个包含10个节点的全连接层，对应10个类别，激活函数为Softmax函数。网络的结构通过Tensorboard可视化如下：

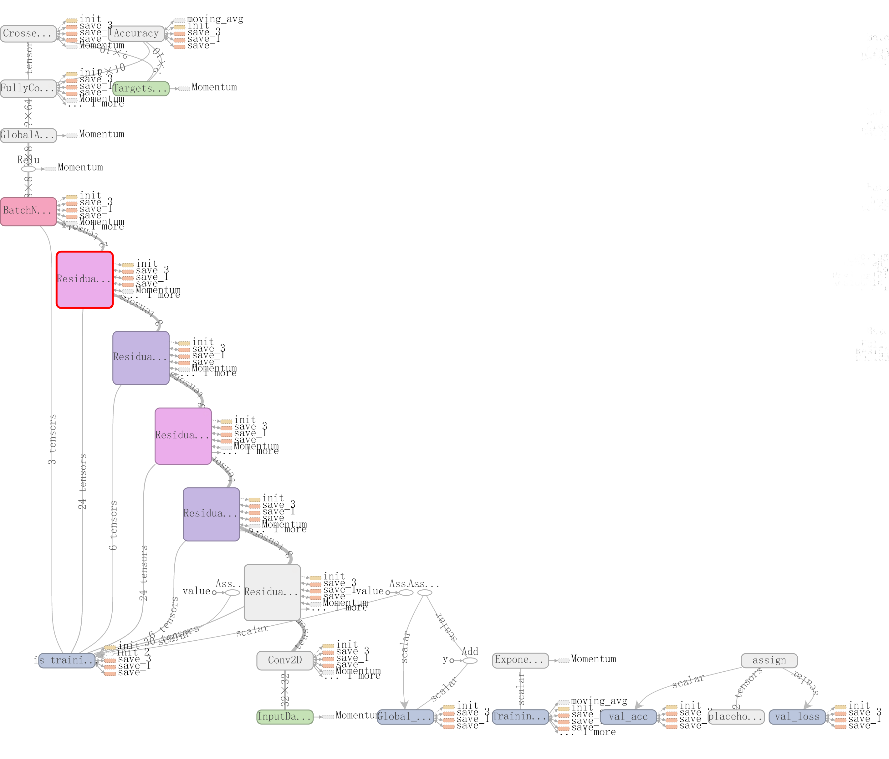


图3-4 深度残差网络结构

## 小组分工

小组共包括两名成员。其中，赵冬迪负责架构设计、卷积神经网络和深度残差网络部分的代码实现以及部分文档的编写；夏光敏负责线性/非线性SVM、Softmax分类器的代码实现，以及测试的工作，并负责了部分文档的编写。两人一同研讨完成了PPT的制作。

在实验完成过程中，两名组员共同努力，互帮互助，遇到问题一同讨论解决。在完成自己的部分工作的基础上还积极地帮助对方完成工作，使项目得以顺利地完成。

## 参考文献

[1] 李航. 统计学习方法[M]. 清华大学出版社, 2012.

[2] Han F, Shan Y, Cekander R, et al. A Two-Stage Approach to People and Vehicle Detection With HOG-Based SVM[J]. 2006.

[3] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[4] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2016:770-778.

# 测试文档

## 测试过程

本实验在CIFAR-10数据集上对我们设计的模型进行训练。CIFAR-10已划分好训练数据和测试数据，在训练数据上分别对模型进行训练，并在测试数据上测试分类结果。并比对模型在训练数据集和测试数据集上的正确率，观察模型的效果。

对于基于传统的特征工程的分类器，包括线性SVM、非线性SVM和Softmax分类器，在测试集上对比准确率。对于卷积神经网络和深度残差网络，对训练过程中准确率等参数的变化进行可视化的比对，并选取样本进行分类，查看分类效果。

## 测试数据集

CIFAR-10数据集包由Alex Krizhevsky、Vinod Nair和Geoffrey Hinton收集而来。数据集下载地址：<http://www.cs.toronto.edu/~kriz/cifar.html>。

CIFAR-10数据集由60000张32\*32的彩色图片组成，共包含10类，每一类6000张图片。数据集中包含50000万张训练图片和10000张测试图片。数据集被分为五个训练批次和一个测试批次。测试批次每一类都准确地包含了1000张随机挑选的图片。其余的图片按随机顺序包含在训练批次中，有的训练批次中可能会出现某一类的图片比其他类别图片多的情况。但是在所有的训练批次中，每一类图片都准确地包含5000张图片。

下图给出数据集中类别标签，并为每一类给出10张随机的图片：

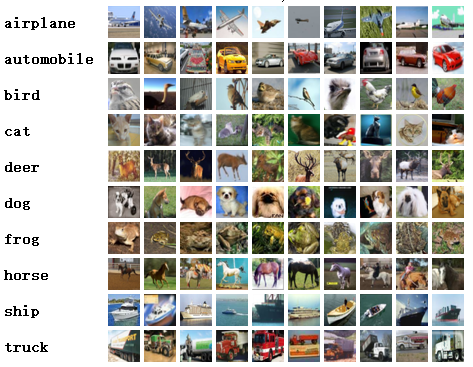


图2-1 CIFAR-10数据集样例

## 测试结果

### 基于特征工程的方法

#### 提取HOG特征

数据集中共包含5个文件，如下：

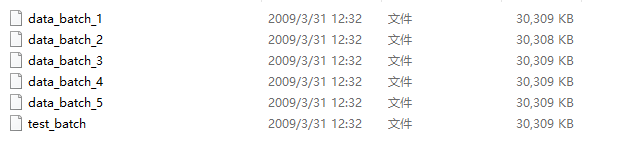


图3-1 CIFAR-10数据集

其中，前五个文件是训练数据，最后一个测试数据。对于训练数据和测试数据提取图片的HOG特征并保存成特征文件，结果如下图：

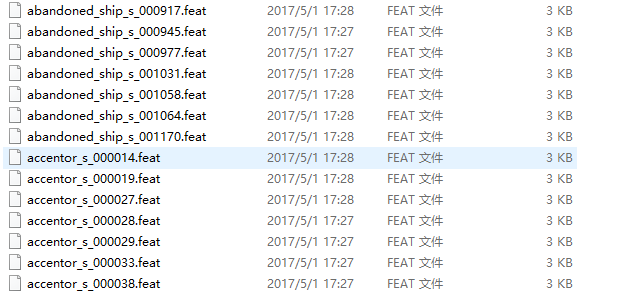


图3-2 HOG特征文件

#### 线性SVM

根据提取后的HOG特征和标签，在训练数据集上训练线性SVM分类器。设置分类器类型为“LIN\_SVM”，生成线性SVM模型，在训练数据集上进行训练，并在测试数据集上测试分类的准确率。

训练过程和结果如下：

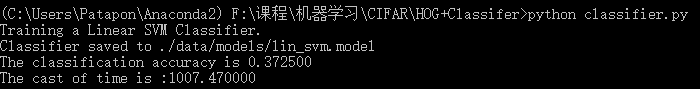


图3-3 线性SVM训练过程

#### 非线性SVM

据提取后的HOG特征和标签，在训练数据集上训练非线性SVM分类器。设置分类器类型为“SVM”，生成非线性SVM模型，在训练数据集上进行训练，并在测试数据集上测试分类的准确率。非线性SVM分类器采用RBF核函数，将特征数据从低维特征映射到高维空间，再利用SVM分类器进行分类。

训练过程和结果如下：

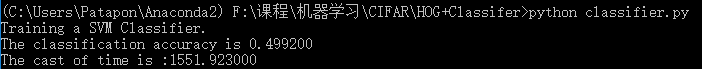


图3-4 非线性SVM训练结果

#### Softmax分类器

根据提取后的HOG特征和标签，在训练数据集上训练Softmax分类器。设置分类器类型为“SOFTMAX”，生成Softmax模型，在训练数据集上进行训练，并在测试数据集上测试分类的准确率。

训练过程和结果如下：

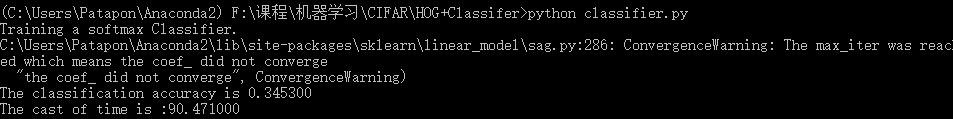


图3-5 Softmax分类器训练结果

比对线性SVM、非线性SVM和Softmax在测试数据集上的分类效果，准确率比对如下：

表3-1 分类器效果比对

|  |  |
| --- | --- |
| 分类器 | 准确率 |
| 线性SVM | 37.25% |
| 非线性SVM | **49.92%** |
| Softmax分类器 | 34.53% |

由表可以看出，非线性SVM在CIFAR-10数据集上的分类准确率远远高于线性SVM和Softmax的分类效果。这是由于，图像的维度较高，在低维的线性空间中很难做到可分，而将特征映射到高维空间中，可能分类效果较好。

### 神经网络

#### 卷积神经网络

搭建含有7个隐含层的卷积神经网络，在训练数据集上对网络进行训练，通过100次迭代，优化网络的参数。训练过程和结果如下：

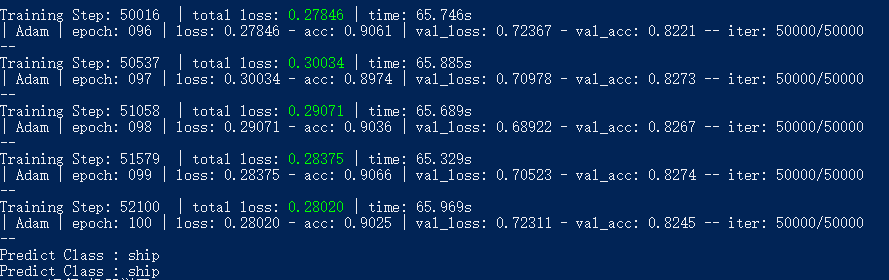


图3-6 卷积神经网络训练过程

由图可以看出，经过100次迭代，卷积神经网络在训练数据上的准确率达到了90%~91%，而在测试数据上的准确率也达到了82%左右。模型训练结束后对两个样本进行了分类，并输出了结果，均为ship，这两个样本如下：



图3-7 卷积神经网络分类样本

真实样本均为ship。

在训练过程中，在训练数据集上的准确率随着每个批次中训练数据的增加而变化的过程通过可视化工具Tensorboard展示如下图：

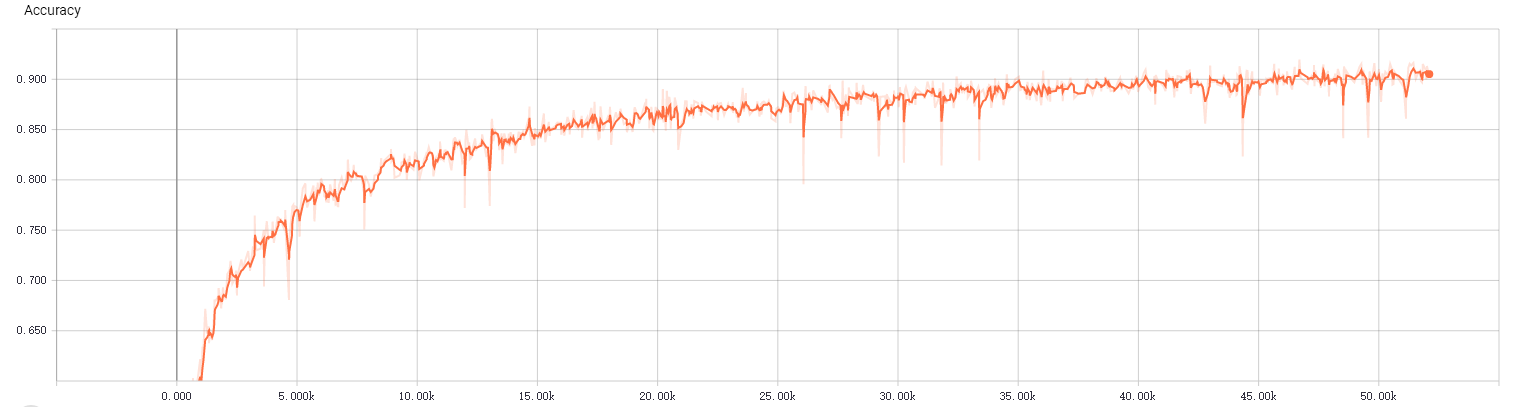


图3-8 卷积神经网络训练过程准确率变化图

在测试集上的准确率随训练过程而变化的趋势通过Tensorboard展示如下：

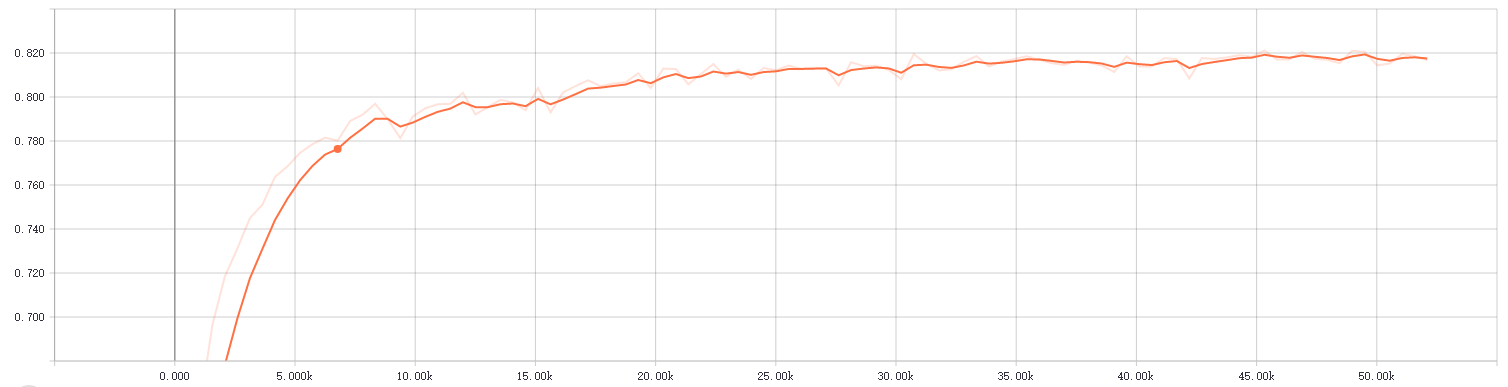


图3-9 卷积神经网络训练过程测试准确率变化图

训练过程中测试集上的误差函数变化趋势如下：

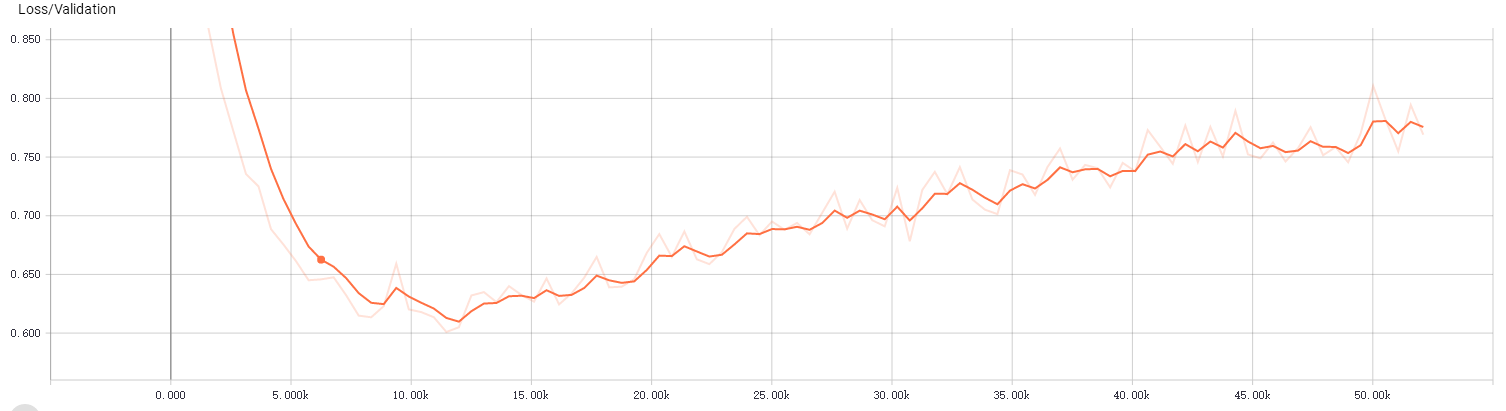


图3-10误差函数变化趋势图

#### 深度残差网络

搭建含有34个隐含层的卷积神经网络，在训练数据集上对网络进行训练，通过150次迭代，优化网络的参数。训练过程和结果如下：

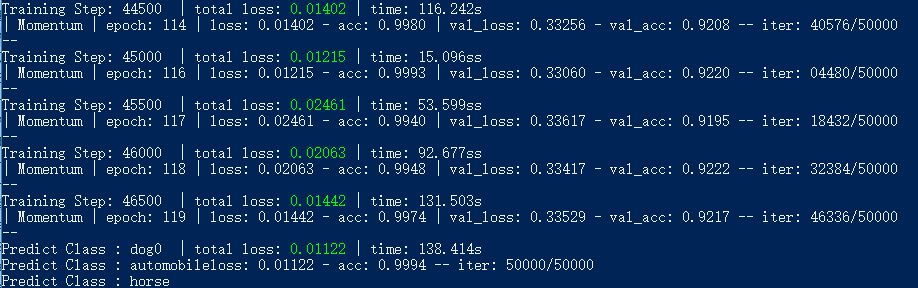


图3-11 深层残差网络训练过程

由图可以看出，经过150次迭代，卷积神经网络在训练数据上的准确率达到了99.94%，而在测试数据上的准确率也稳定在了92%左右。模型训练结束后对三个样本进行了分类，并输出了结果，这三个样本如下：

D:\datasets\cifar-10-batches-py\26036\5_1411.jpgD:\datasets\cifar-10-batches-py\26036\1_4.jpgD:\datasets\cifar-10-batches-py\26036\7_388.jpg

图3-12 深度残差网络分类样本

真实样本标签为：

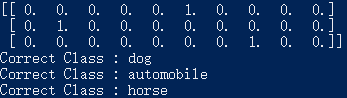


图3-13 样本真实标签

可见样本分类标签与真实标签相同。

在训练过程中，在训练数据集上的准确率随着每个批次中训练数据的增加而变化的过程通过可视化工具Tensorboard展示如下图：

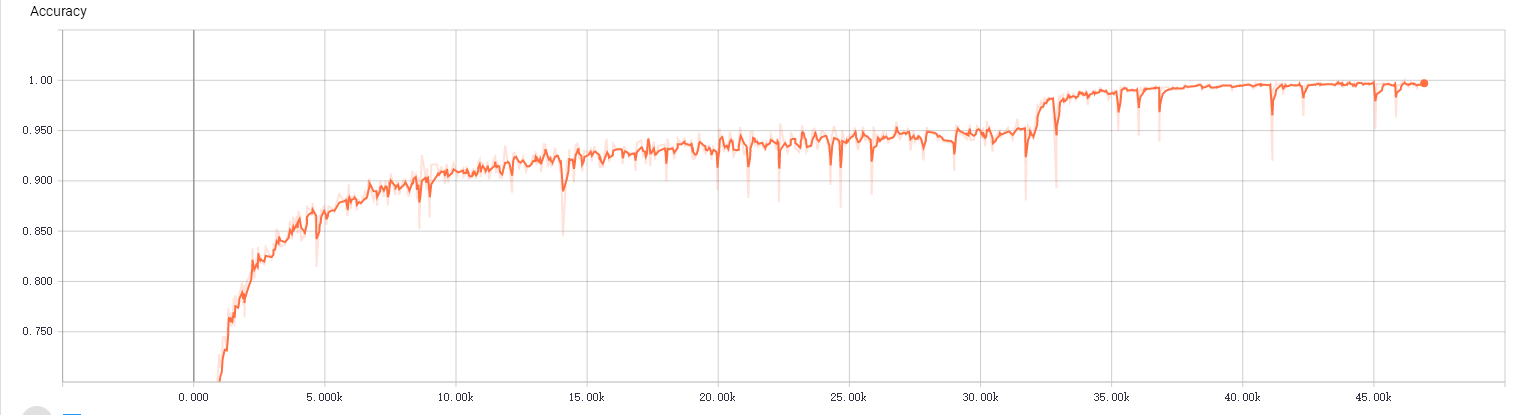


图3-14 深度残差网络训练过程准确率变化图

在测试集上的准确率随训练过程而变化的趋势通过Tensorboard展示如下：

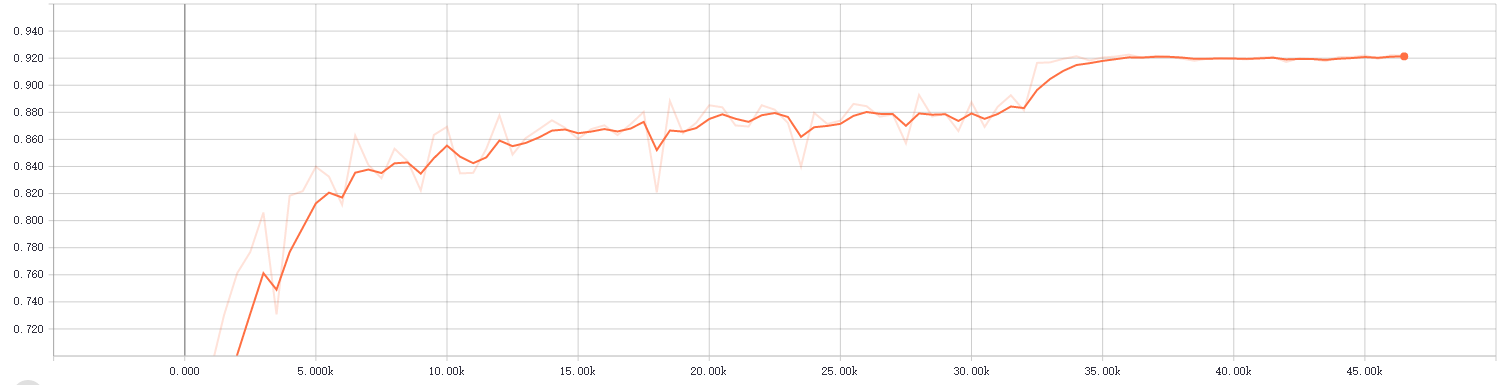


图3-15 深度残差网络训练过程测试准确率变化图

训练过程中在测试集上误差函数变化趋势如下：

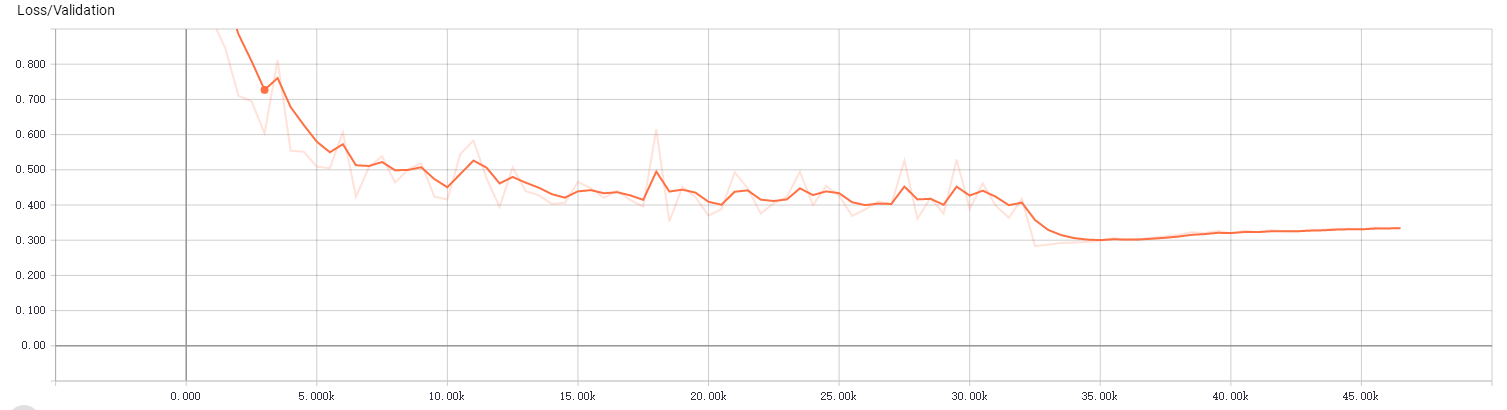


图3-16误差函数变化趋势图

# 环境配置及相关说明

## 环境配置

由于本项目中包含多个独立的实验，分别为基于传统的特征工程的方法和端到端的神经网络。两部分的实验在不同的电脑上完成，所以环境配置存在一些不同，下面分别进行介绍。

### HOG+SVM/Softmax

HOG+SVM/Softmax的程序是基于Python2.7编写的，在Python科学计算环境Anaconda下进行实验。Anaconda是一个基于Python的科学计算环境，自带包管理器conda，包含了常用的如Numpy、Scipy等常用的扩展库。

在HOG+SVM/Softmax的程序中，依赖的包如下表：

表1-1 HOG+SVM/Softmax实验扩展包

|  |  |
| --- | --- |
| **扩展包** | **功能** |
| Numpy | 数值计算扩展库，用来存储和处理大型矩阵 |
| Scipy | 在Numpy库的基础上增加了常用的库函数 |
| Matplotlib | Python的2D绘图库 |
| Scikit-learn | 集成了多种分类和聚类算法的Python工具包 |
| Skimage | Python下的图像处理库 |

### 神经网络

由于神经网络参数较多，优化起来计算量很大，计算时间较久，因此采用了GPU加速的方法。神经网络部分的实验是在一台装有NVIDIA GTX 1060，6G显存显卡的机器上进行的，环境配置如下：

这部分实验程序是基于Python3.5编写的，由于使用了GPU加速，需要安装NVIDIA运算平台CUDA，本实验使用了CUDA8.0。

其他依赖的扩展库包括：

表1-2 神经网络实验扩展包

|  |  |
| --- | --- |
| **扩展包** | **功能** |
| Numpy | 数值计算扩展库，用来存储和处理大型矩阵 |
| Scipy | 在Numpy库的基础上增加了常用的库函数 |
| Matplotlib | Python的2D绘图库 |
| Scikit-learn | 集成了多种分类和聚类算法的Python工具包 |
| Skimage | Python下的图像处理库 |
| Tensorflow | Google开发的深度学习框架 |
| Tflearn | 对Tensorflow接口进行了更高层次封装的工具包 |
| PIL | Python图形处理库 |

# 源代码说明

## HOG+SVM/Softmax

### HOG特征提取

HOG特征的提取参考网上的资料，利用扩展包Skimage集成的hog方法对图片数据进行特征提取，下面对核心代码进行分析：

def getFeat(TrainData, TestData):

for data in TestData:

image = np.reshape(data[0].T, (32, 32, 3))

gray = rgb2gray(image)/255.0

fd = hog(gray, 9, [8, 8], [2, 2], 'L2-Hys', False, True)

fd = np.concatenate((fd, data[1]))

filename = list(data[2])

fd\_name = filename[0].split('.')[0]+'.feat'

fd\_path = os.path.join('./data/features/test/', fd\_name)

joblib.dump(fd, fd\_path)

print "Test features are extracted and saved."

for data in TrainData:

image = np.reshape(data[0].T, (32, 32, 3))

gray = rgb2gray(image)/255.0

fd = hog(gray, 9, [8, 8], [2, 2], 'L2-Hys', False, True)

fd = np.concatenate((fd, data[1]))

filename = list(data[2])

fd\_name = filename[0].split('.')[0]+'.feat'

fd\_path = os.path.join('./data/features/train/', fd\_name)

joblib.dump(fd, fd\_path)

print "Train features are extracted and saved."

def rgb2gray(im):

gray = im[:, :, 0]\*0.2989+im[:, :, 1]\*0.5870+im[:, :, 2]\*0.1140

return gray

分别对训练批次的数据和测试批次的数据进行特征提取，主要通过hog方法，hog函数的使用方法如下：

hog(image, orientations, pixels\_per\_cell, cells\_per\_block, transform\_sqrt, visualise)

参数解释如下：

image:图像矩阵；

orientations：取向桶的数量；

pixels\_per\_cell：每个细胞单元的像素点个数；

cells\_per\_block：每个块内细胞单元的个数；

transform\_sqrt：是否进行gamma校正；

visualise：是否生成可视化图片。

### 线性SVM

采用Scikit-learn工具中集成的线性SVM方法对提取HOG特征后的图片数据进行分类，并统计准确率和模型训练的时间。

if clf\_type is 'LIN\_SVM':

clf = LinearSVC()

print "Training a Linear SVM Classifier."

clf.fit(fds, labels)

print "Classifier saved to {}".format('./data/models/lin\_svm.model')

for feat\_path in glob.glob(os.path.join('./data/features/test', '\*.feat')):

total += 1

data\_test = joblib.load(feat\_path)

data\_test\_feat = data\_test[:-1].reshape((1, -1))

result = clf.predict(data\_test\_feat)

if int(result) == int(data\_test[-1]):

num += 1

rate = float(num)/total

t1 = time.time()

print 'The classification accuracy is %f'%rate

print 'The cast of time is :%f'%(t1 - t0)

### 非线性SVM

非线性SVM是在线性SVM分类器的基础上，加入了核方法，将低维空间中线性不可分的数据映射到高维空间，使其在高维空间中变得可分。在本实验中，选用RBF核函数对数据进行映射，惩罚因子设置为0.5。

elif clf\_type is 'SVM':

clf = SVC(C = 0.5, decision\_function\_shape = 'ovr', degree = 4)

print "Training a SVM Classifier."

clf.fit(fds, labels)

print "Classifier saved to {}".format('./data/models/svm.model')

for feat\_path in glob.glob(os.path.join('./data/features/test', '\*.feat')):

total += 1

data\_test = joblib.load(feat\_path)

data\_test\_feat = data\_test[:-1].reshape((1, -1))

result = clf.predict(data\_test\_feat)

if int(result) == int(data\_test[-1]):

num += 1

rate = float(num)/total

t1 = time.time()

print 'The classification accuracy is %f'%rate

print 'The cast of time is :%f'%(t1 - t0)

### Softmax分类器

elif clf\_type is 'SOFTMAX':

clf = LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, solver='sag', multi\_class='multinomial', verbose=0, warm\_start=False, n\_jobs=1)

print "Training a softmax Classifier."

clf.fit(fds, labels)

print "Classifier saved to {}".format('./data/models/softmax.model')

for feat\_path in glob.glob(os.path.join('./data/features/test', '\*.feat')):

total += 1

data\_test = joblib.load(feat\_path)

data\_test\_feat = data\_test[:-1].reshape((1, -1))

result = clf.predict(data\_test\_feat)

if int(result) == int(data\_test[-1]):

num += 1

rate = float(num)/total

t1 = time.time()

print 'The classification accuracy is %f'%rate

print 'The cast of time is :%f'%(t1 - t0)

使用Scikit-learn工具包中集成的线性回归方法。设置multi\_class参数设置为multinomial，则会使用Softmax函数对数据进行分类，预测分类的结果是样本属于每一类的概率。

## 神经网络

### 2.1 卷积神经网络

本实验中，卷积神经网络的实现是通过一个封装了Tensorflow接口的高层次的深度学习框架——Tflearn来实现的。核心代码如下：

img\_prep = ImagePreprocessing()

img\_prep.add\_featurewise\_zero\_center()

img\_prep.add\_featurewise\_stdnorm()

img\_aug = ImageAugmentation()

img\_aug.add\_random\_flip\_leftright()

img\_aug.add\_random\_rotation(max\_angle=25.)

network = input\_data(shape=[None, 32, 32, 3],

data\_preprocessing=img\_prep,

data\_augmentation=img\_aug)

network = conv\_2d(network, 32, 3, activation='relu')

network = max\_pool\_2d(network, 2)

network = conv\_2d(network, 64, 3, activation='relu')

network = conv\_2d(network, 64, 3, activation='relu')

network = max\_pool\_2d(network, 2)

network = fully\_connected(network, 512, activation='relu')

network = dropout(network, 0.5)

network = fully\_connected(network, 10, activation='softmax')

network = regression(network, optimizer='adam',

loss='categorical\_crossentropy',

learning\_rate=0.001)

model = tflearn.DNN(network, tensorboard\_verbose=1)

model.fit(X, Y, n\_epoch=100, shuffle=True, validation\_set=(X\_test, Y\_test),

show\_metric=True, batch\_size=96, run\_id='cifar10\_cnn')

首先对数据进行预处理，包括对每个通道的数据标准化和图片的增强。然后建立了一个含有7个隐藏层的卷积神经网络，其中含有5个卷积层（含有两个最大池化层）和2个全连接层。迭代次数为100次，batch\_size为96。

卷积层使用的激活函数是修正线性单元（RELU）函数，第二个全连接层采用softmax激活函数。网络采用交叉熵作为损失函数。

### 2.2 深度残差网络

深度残差网络同样采用Tflearn工具实现，核心代码如下：

img\_prep = tflearn.ImagePreprocessing()

img\_prep.add\_featurewise\_zero\_center(per\_channel=True)

img\_aug = tflearn.ImageAugmentation()

img\_aug.add\_random\_flip\_leftright()

img\_aug.add\_random\_crop([32, 32], padding=4)

net = tflearn.input\_data(shape=[None, 32, 32, 3],

data\_preprocessing=img\_prep,

data\_augmentation=img\_aug)

net = tflearn.conv\_2d(net, 16, 3, regularizer='L2', weight\_decay=0.0001)

net = tflearn.residual\_block(net, n, 16)

net = tflearn.residual\_block(net, 1, 32, downsample=True)

net = tflearn.residual\_block(net, n-1, 32)

net = tflearn.residual\_block(net, 1, 64, downsample=True)

net = tflearn.residual\_block(net, n-1, 64)

net = tflearn.batch\_normalization(net)

net = tflearn.activation(net, 'relu')

net = tflearn.global\_avg\_pool(net)

net = tflearn.fully\_connected(net, 10, activation = 'softmax')

mom = tflearn.Momentum(0.1, lr\_decay = 0.1, decay\_step = 32000, staircase = True)

net = tflearn.regression(net, optimizer=mom,

loss='categorical\_crossentropy')

model = tflearn.DNN(net, tensorboard\_verbose = 1, clip\_gradients = 0.)

model.fit(X, Y, n\_epoch = 200, validation\_set = (testX, testY), snapshot\_epoch = False, snapshot\_step = 500,show\_metric = True, batch\_size = 128, shuffle = True, run\_id = 'resnet\_cifar10')

首先同样对数据进行预处理，包括对每个通道的数据标准化和图片的增强。深度残差网络的建立主要通过残差块（residual\_block），加入残差块的代码语句解释如下：

**tflearn.layers.conv.residual\_block** (incoming, nb\_blocks, out\_channels, downsample=False, downsample\_strides=2, activation='relu', batch\_norm=True, bias=True, weights\_init='variance\_scaling', bias\_init='zeros', regularizer='L2', weight\_decay=0.0001, trainable=True, restore=True, reuse=False, scope=None, name='ResidualBlock')

本实验中使用的参数为nb\_blocks，即残差块的个数；以及out\_channels，即卷积层含有卷积过滤器的个数。

模型训练过程，迭代次数为200次，batch\_size为128。