【此系列笔记，最好和其他框架和markdown版本的笔记，一一对应】

记录从简单到复杂的代码

# 使用

# 【深度学习pipline】

# 【1数据】

跑各种超参数的时候，也要注意保留模型参数！

自己多准备utils

剩余时间，已跑时间

### 【代码排版】

要调节的参数 单独放排列在前面！

接着训练的epoch信息要保留！

跑一段时间参数信息就要保留一下

### 【读取数据-scratch】

##### 1-解压数据

*# 如果训练下载的Kaggle的完整数据集，把下面改False*

demo **=** True

**if** demo:

**import** zipfile

**for** fin **in** ['train\_tiny.zip', 'test\_tiny.zip', 'trainLabels.csv.zip']:

**with** zipfile**.**ZipFile('../data/kaggle\_cifar10/' **+** fin, 'r') **as** zin:

zin**.**extractall('../data/kaggle\_cifar10/')

##### 2-整理数据



原来

/train

/test 各个图片

整理后

/input\_dir/train\_valid/label 文件夹

/input\_dir/train/各label 夹

/input\_dir/valid/各label 夹

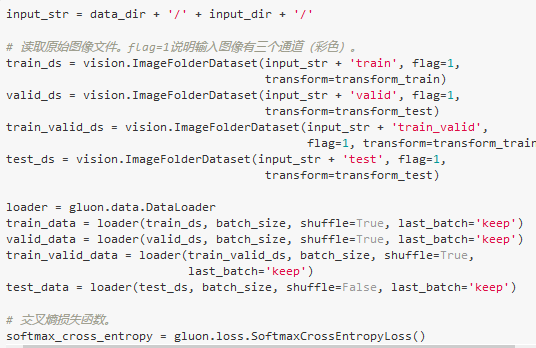
/input\_dir/test/unkown



##### 3-读取数据

###### 【gluon】

我们可以使用Gluon中的ImageFolderDataset类来读取整理后的数据集。

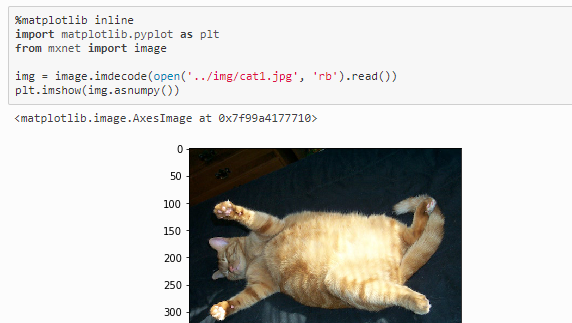


#### -csv

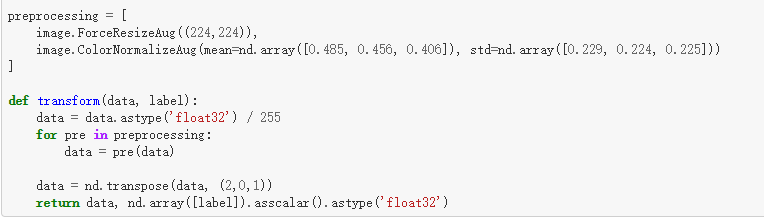


### 【图片数据】

#### 【读】

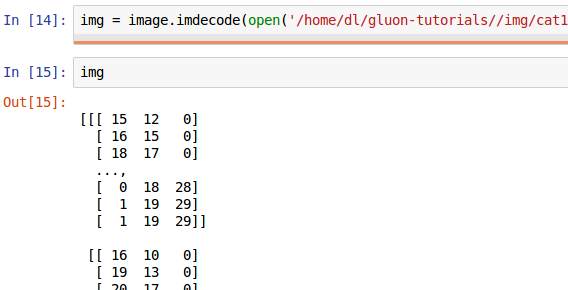


#### 增强

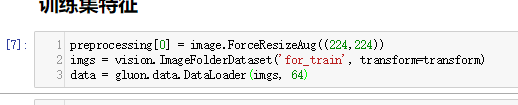








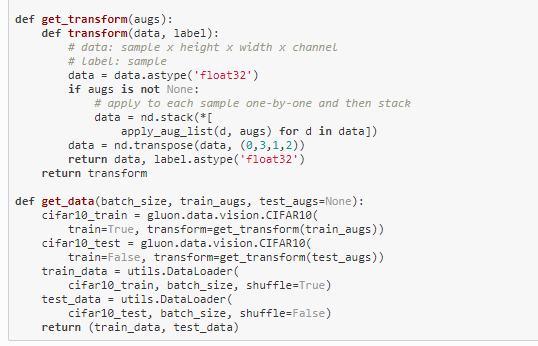
#### 【变形】



### 【文本数据】

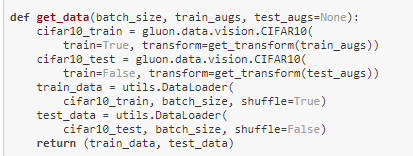
### 【读取框架自带数据】

Gluon带增广参数



【】

#### 【gluon】

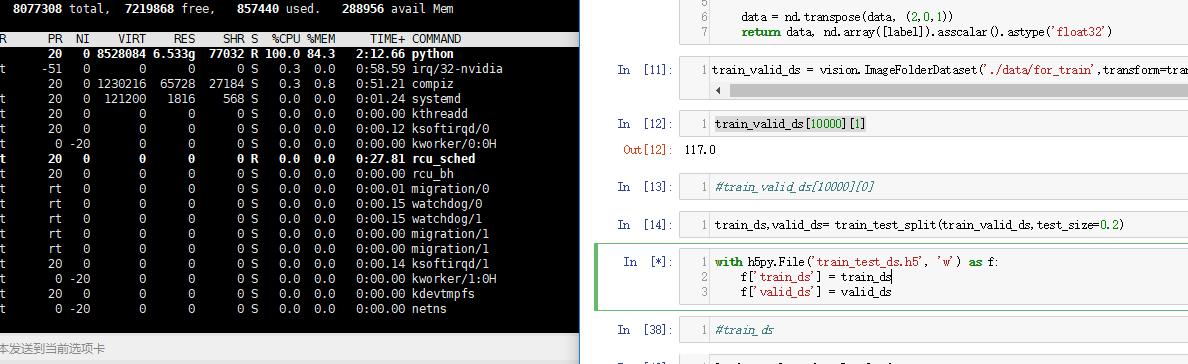




##### -nd.stack(\*data) 先拆

### 【数据划分】

内存占用比较大啊,也很费时间！



### 【读数据的步骤】

##### 形成dataset

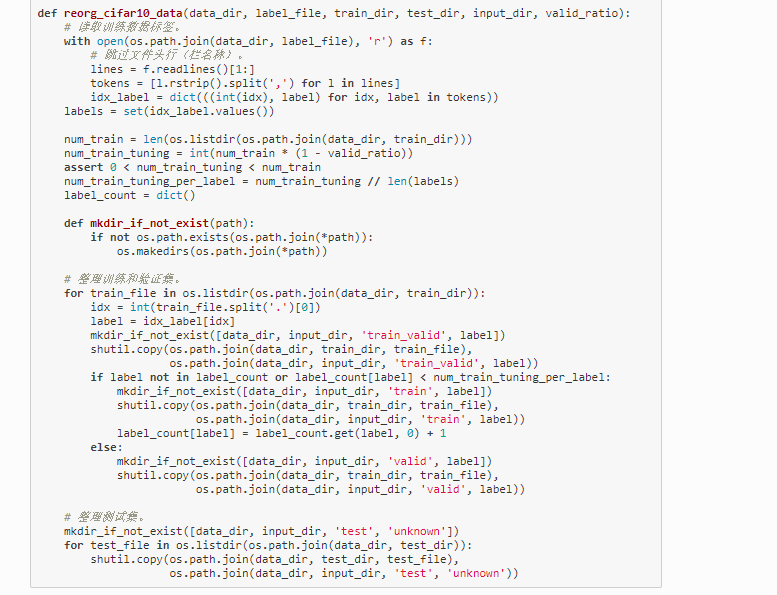
##### 形成dataloader

1. 形成dataloader (iterable,可for循环)，

valid的batch\_size无所谓

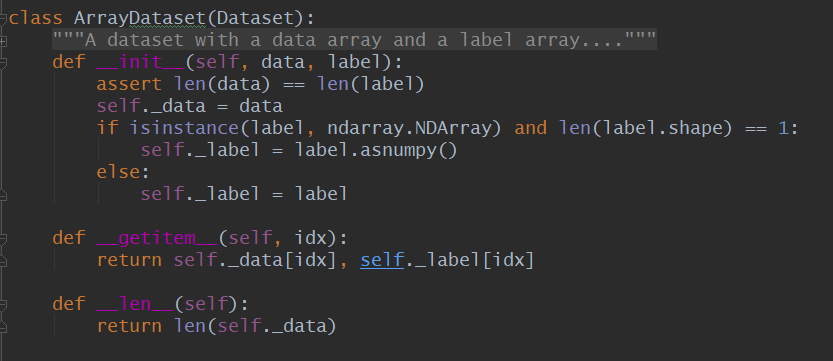
#### 【读取数据流程】

#### 【1】分类放进文件夹

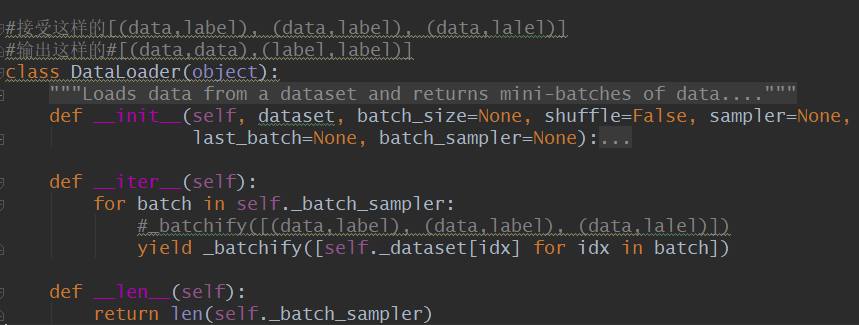
 

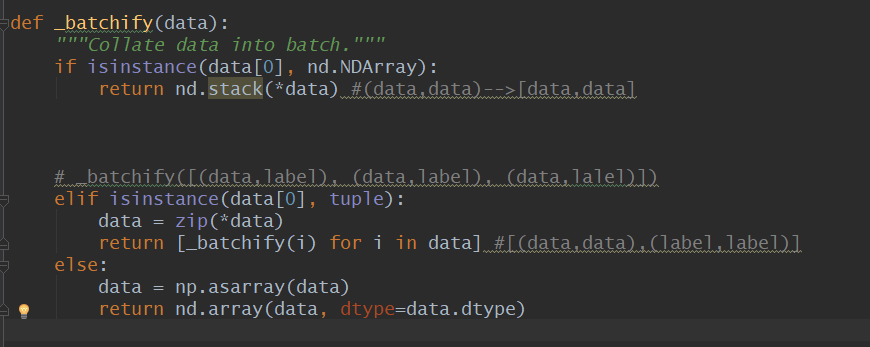
【2 】dataset(Transform+数据增强)+DataLoader（注意接受和输出格式）分立->tuple->分立

分立->tuple->分立 pytorch也是类似



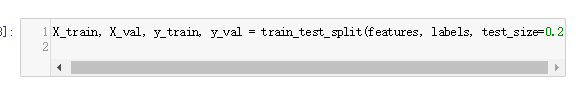
mxnet 的loader 都给变成了一批纯data，和一批纯标签





!!!! 一个data和label是组合方式

所有的【data..】 和 所有的label



这种符合现有的train写法

【(一个data,一个label)】，

##### 【多个版本transform角度（带数据增强）】

这里带label的原因是，下文Folder Dataset的时候有transform要带label。

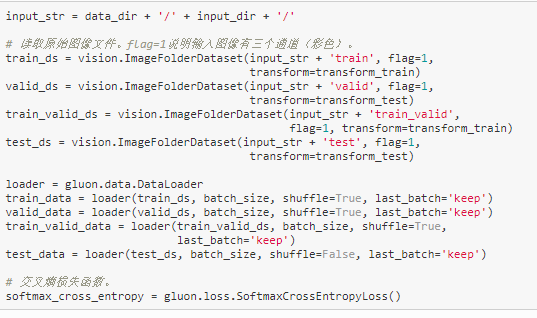
通过看源码知道label是数字化了，但是在源码外面 看不透label是啥。

Sherlock的，就没有带label。我也觉得自己写DataSet函数比较好。能看清楚

###### 【data,label版—适用于ImageFolderDataset--limu -120 dog】

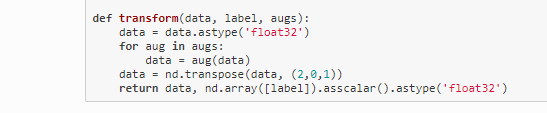
图片大小 96



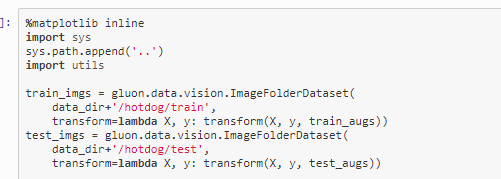


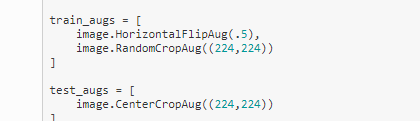
###### 【（data,label）外套私货版-李沐热狗迁移】

参数有了仨，实际上不能直接使用，只能有两个参数



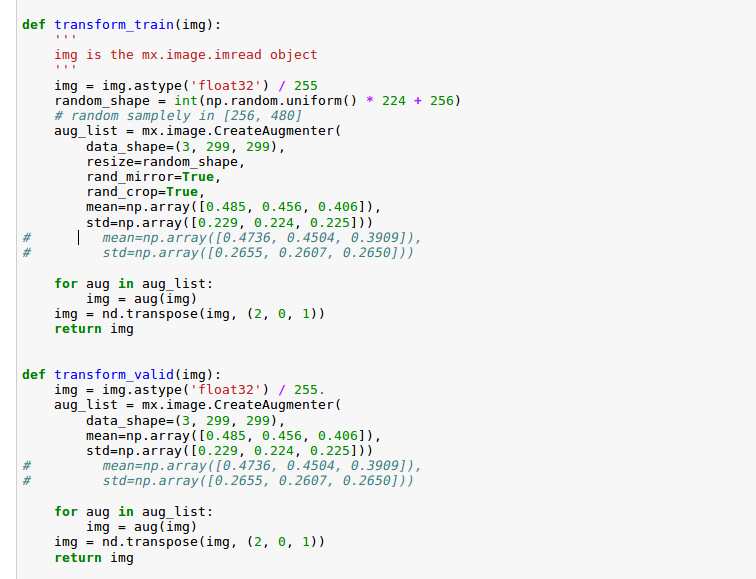
这里才是真正transform，



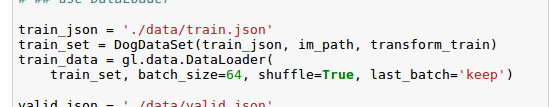


###### 【单data(img)版-自设计--sherlock-120dog】

图片大小 224-480



如上所说，建议写自己的DataSet类。下面Sherlock就是这么干的！



##### 【多个版本数据增强】---其实和上面是一体的

代码

###### 【1】

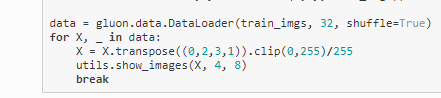


###### 【2】

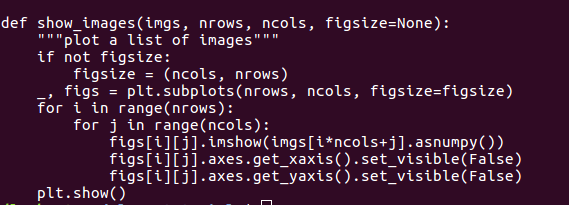


### 【显示数据】

取一批数据看看，break就好了



在utils中



# 【2模型】

### 【网络模型的选取与设计】

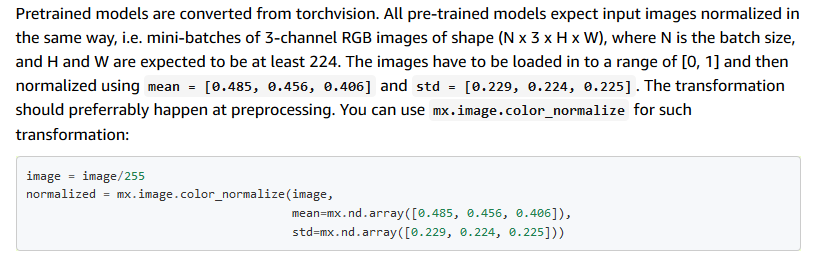
#### model zoo-gluon,tf,

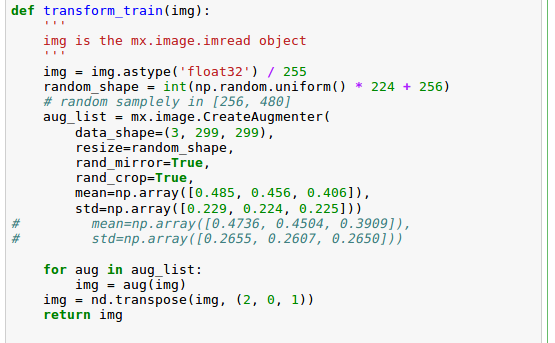
Sherlock 给的狗代码就是model\_zoo

<http://mxnet.incubator.apache.org/api/python/gluon/model_zoo.html>

好像只有vision类的?

##### 使用预训练参数的前提





### 【网络模型代码tip】

#### 【tflearn 可以改造成 打印网络结构，但是我忘了怎么弄了】

#### 【gluon最终网络为class还是函数呢？】

-class可以方便的在forward中打印网络形状，但是只能打印最外面一层的网络结构

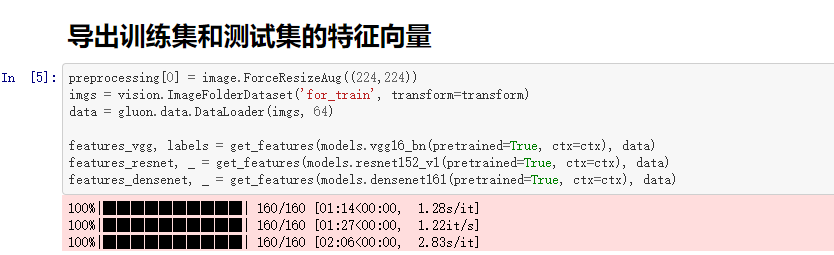
# 【3损失函数】

### 【损失函数的选择】

### 【迁移学习】注意模型也要搞到gpu上

#### get\_feature大法

<https://github.com/ypwhs/DogBreed_gluon/blob/master/get_features.ipynb>

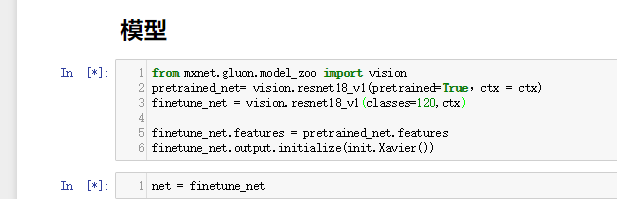


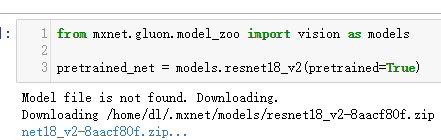
否则报错,list out of range。

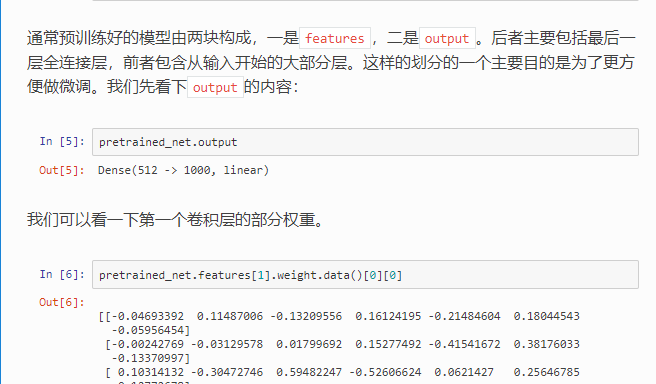
下载时间有点长！！！

改环境变量， 具体在后面源码部分。

#### 下载权重数据

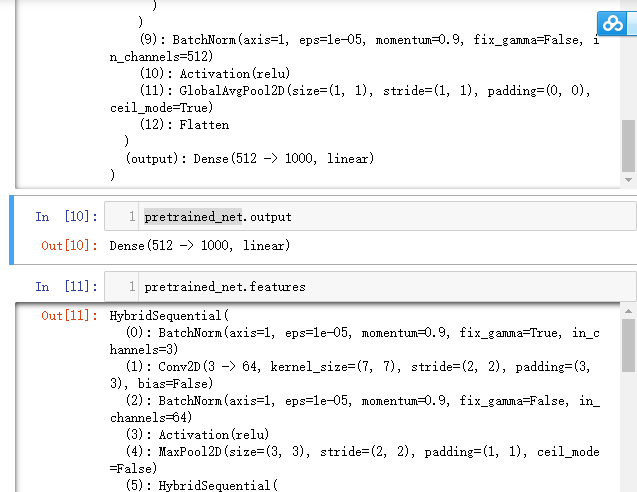


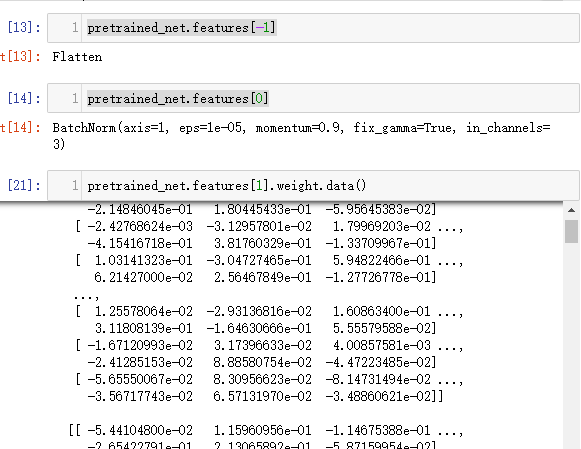




更详细的：

#### 显示权重数据

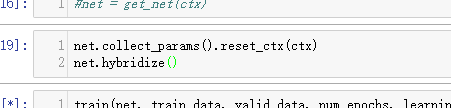




#### 实际使用：下载一个空网络然后，然后塞进参数

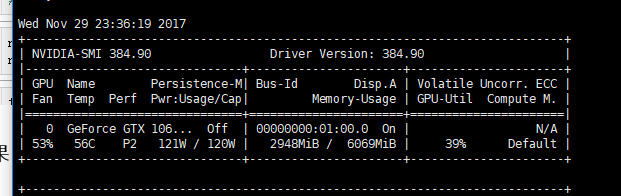


#### 实际使用：Model\_zoo的参数要放在gpu上！



#### 使用效果：

##### Resnet 18 显存占用只有一半



### 【训练-找到合适参数】

# 【4优化器】

Pytorch的注意！

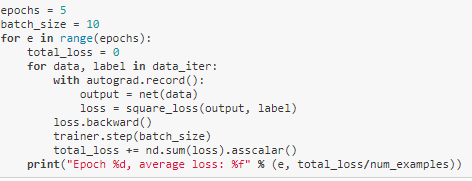


# 【5训练】

## 【.step()】

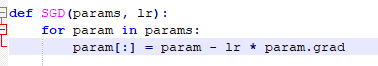
### 【mxnet-gluon-trainer.step】



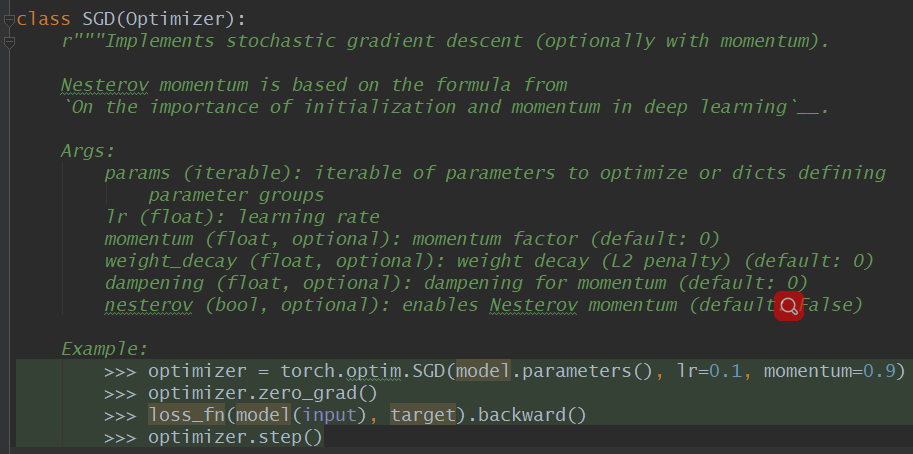


### 【mxnet-scratch-SGD】





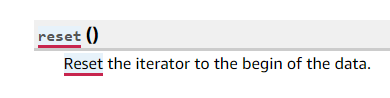
### 【pytorch-optimizer.step】需要zero\_grad一下



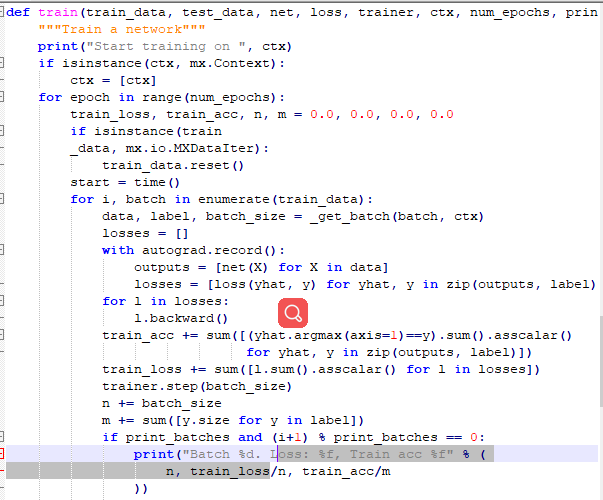
### 【各种的train】 写法相当多变！最终要有一套自己喜欢的！

#### 李沐-utils.py中的，功能初级，不带weight\_decay,lr\_decay,lr\_period

-reset()



1-数据整理用Sherlock的，再跑李沐教程



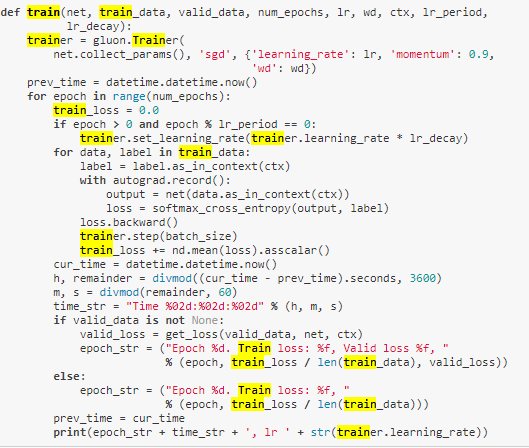
#### 李沐-cifar10 比utils高级，带各种decay!!!! 和acc



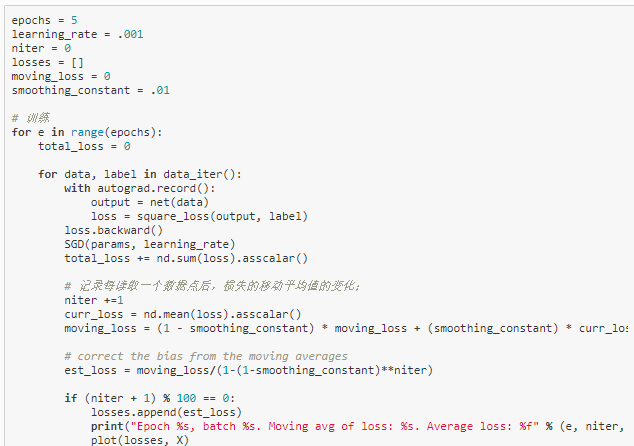
#### 李沐-cifar10[改]-trainer要写在外面

#### 李沐-120 dog 和上面一样，没有acc.

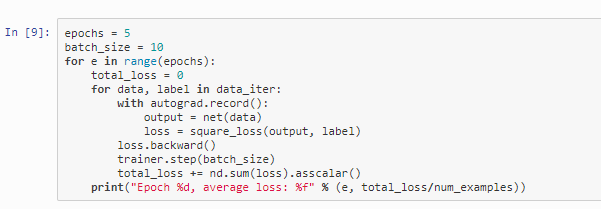
做这个项目的时候，要把train换成utils的



#### 李沐线性回归从0中的=两种loss。。没有acc



#### 李沐线性gluon中的=只有average loss



#### 李沐正则化Gluon中的=带画出loss图的



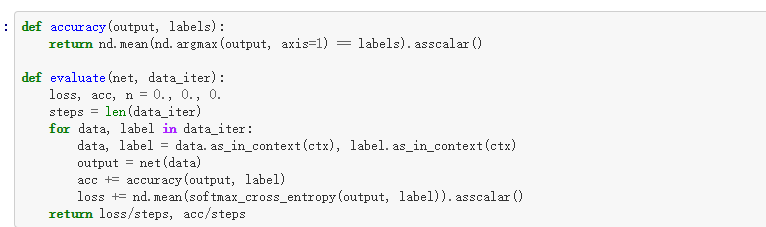
用不用画出时间图呢？

### 【train中的acc】

回归问题可没有哦！

ypw 120dog版本的

# Licensed to the Apache Software Foundation (ASF) under one  
# or more contributor license agreements. See the NOTICE file  
# distributed with this work for additional information  
# regarding copyright ownership. The ASF licenses this file  
# to you under the Apache License, Version 2.0 (the  
# "License"); you may not use this file except in compliance  
# with the License. You may obtain a copy of the License at  
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing,  
# software distributed under the License is distributed on an  
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY  
# KIND, either express or implied. See the License for the  
# specific language governing permissions and limitations  
# under the License.  
  
# coding: utf-8  
# pylint: disable=  
*"""Dataset generator."""*\_\_all\_\_ = ['DataLoader']  
  
import numpy as np  
  
from . import sampler as \_sampler  
from ... import nd  
  
  
def \_batchify(data):  
 *"""Collate data into batch."""* if isinstance(data[0], nd.NDArray):  
 return nd.stack(\*data) #(data,data)-->[data,data]  
  
  
  
 # \_batchify([(data,label), (data,label), (data,lalel)])  
 elif isinstance(data[0], tuple):  
 data = zip(\*data)  
 return [\_batchify(i) for i in data] #[(data,data),(label,label)]  
 else:  
 data = np.asarray(data)  
 return nd.array(data, dtype=data.dtype)  
  
#接受这样的[(data,label), (data,label), (data,lalel)]  
#输出这样的#[(data,data),(label,label)]  
class DataLoader(object):  
 *"""Loads data from a dataset and returns mini-batches of data.  
  
 Parameters  
 ----------  
 dataset : Dataset  
 Source dataset. Note that numpy and mxnet arrays can be directly used  
 as a Dataset.  
 batch\_size : int  
 Size of mini-batch.  
 shuffle : bool  
 Whether to shuffle the samples.  
 sampler : Sampler  
 The sampler to use. Either specify sampler or shuffle, not both.  
 last\_batch : {'keep', 'discard', 'rollover'}  
 How to handle the last batch if batch\_size does not evenly divide  
 `len(dataset)`.  
  
 keep - A batch with less samples than previous batches is returned.  
 discard - The last batch is discarded if its incomplete.  
 rollover - The remaining samples are rolled over to the next epoch.  
 batch\_sampler : Sampler  
 A sampler that returns mini-batches. Do not specify batch\_size,  
 shuffle, sampler, and last\_batch if batch\_sampler is specified.  
 """* def \_\_init\_\_(self, dataset, batch\_size=None, shuffle=False, sampler=None,  
 last\_batch=None, batch\_sampler=None):  
 self.\_dataset = dataset  
  
 if batch\_sampler is None:  
 if batch\_size is None:  
 raise ValueError("batch\_size must be specified unless " \  
 "batch\_sampler is specified")  
 if sampler is None:  
 if shuffle:  
 sampler = \_sampler.RandomSampler(len(dataset))  
 else:  
 sampler = \_sampler.SequentialSampler(len(dataset))  
 elif shuffle:  
 raise ValueError("shuffle must not be specified if sampler is specified")  
  
 batch\_sampler = \_sampler.BatchSampler(  
 sampler, batch\_size, last\_batch if last\_batch else 'keep')  
 elif batch\_size is not None or shuffle or sampler is not None or \  
 last\_batch is not None:  
 raise ValueError("batch\_size, shuffle, sampler and last\_batch must " \  
 "not be specified if batch\_sampler is specified.")  
  
 self.\_batch\_sampler = batch\_sampler  
  
 def \_\_iter\_\_(self):  
 for batch in self.\_batch\_sampler:  
 #\_batchify([(data,label), (data,label), (data,lalel)])  
 yield \_batchify([self.\_dataset[idx] for idx in batch])  
  
 def \_\_len\_\_(self):  
 return len(self.\_batch\_sampler)



#ypw中的

def evaluate(data\_iter,net):

ctx =

loss, acc, n = 0., 0., 0.

steps = len(data\_iter)

for data, label in data\_iter:

data, label = data.as\_in\_context(ctx), label.as\_in\_context(ctx)

output = net(data)

acc += accuracy(output, label)

loss += nd.mean(softmax\_cross\_entropy(output, label)).asscalar()

return loss/steps, acc/steps

#### 多类lr从0中的

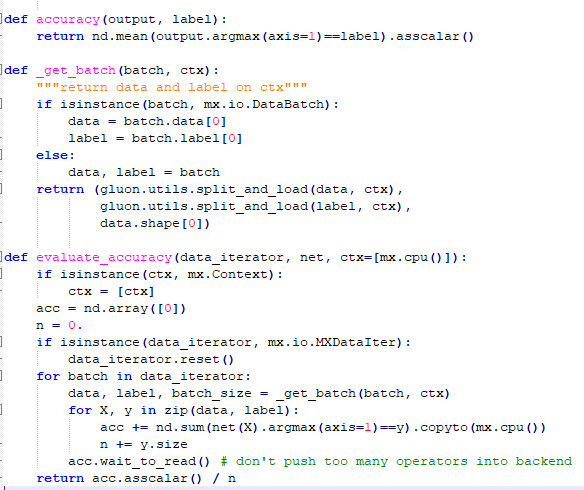
【一个batch的acc和数据集的acc】

数据集可能放不进显存中，所以也是要分batch的！



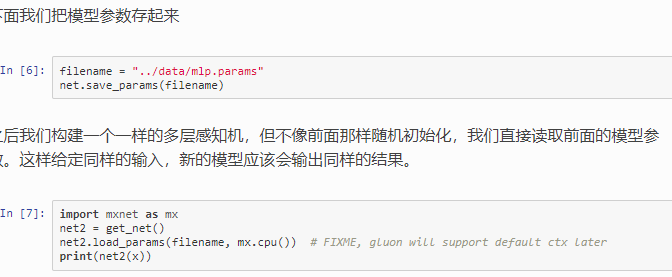
#### -李沐utils中的

Acc 除以的n是什么？



### 【用序列化之后的模型，测试】

### 【序列化模型】

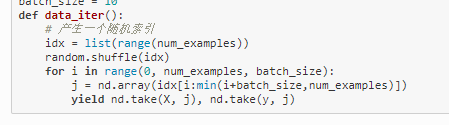


### 【高级】

【】

# Utils 有时实现的是和源码一样的功能

## -data\_iter()

、**import** random

batch\_size **=** 10

**def** **data\_iter**():

*# 产生一个随机索引*

idx **=** list(range(num\_examples))

random**.**shuffle(idx)

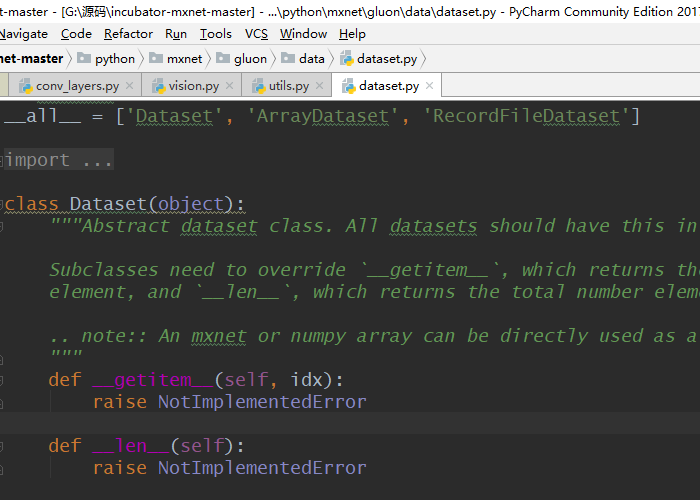
**for** i **in** range(0, num\_examples, batch\_size):

j **=** nd**.**array(idx[i:min(i**+**batch\_size,num\_examples)])

**yield** nd**.**take(X, j), nd**.**take(y, j)

# 源码

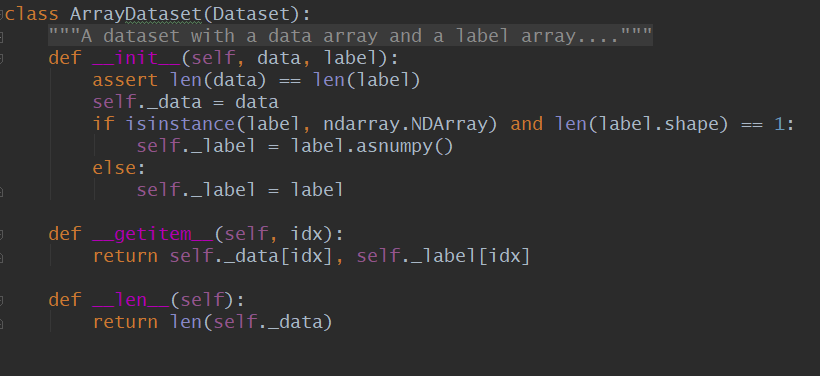
## Dataset

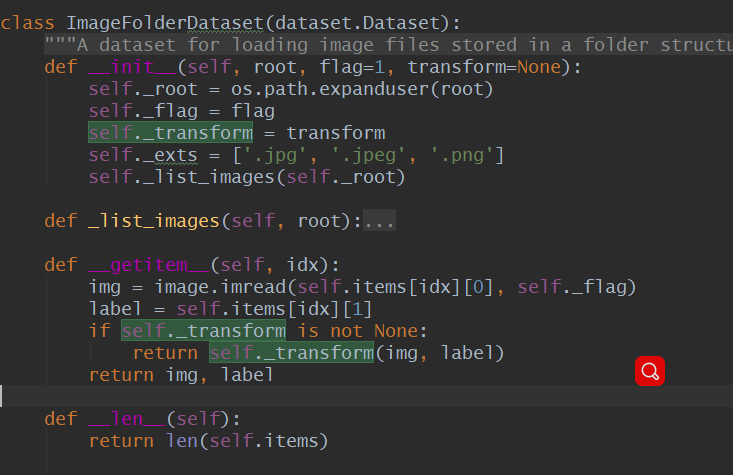


img,label一起

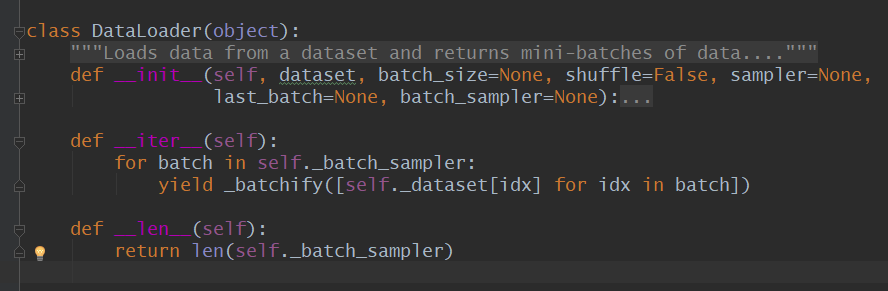
将分立的data,label 能一起检索







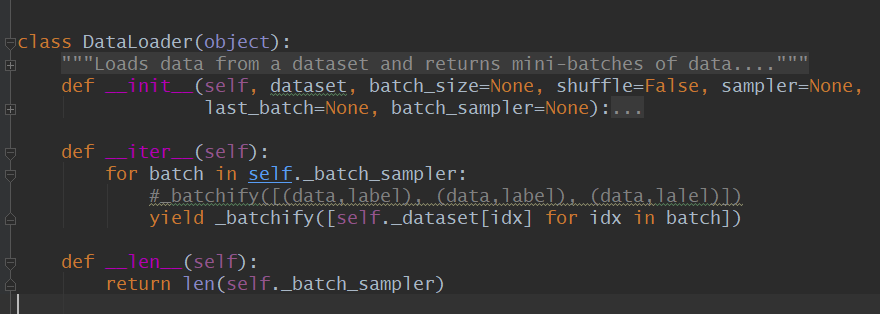
可迭代对象-for循环。



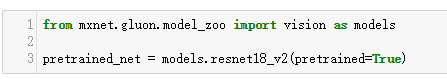
## DataLoader :

[(data,label), (data,label), (data,lalel)])

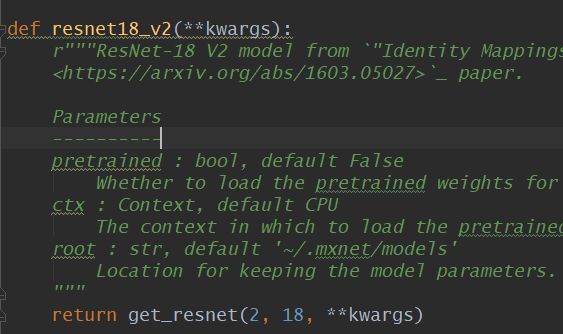
转化为单data,和lable



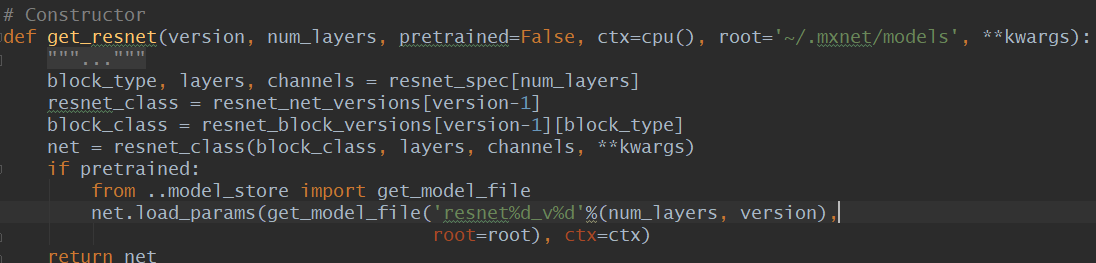
## Model Zoo（一定要改MXNET\_GLUON\_REPO环境变量）



### --ctx是可以reset的







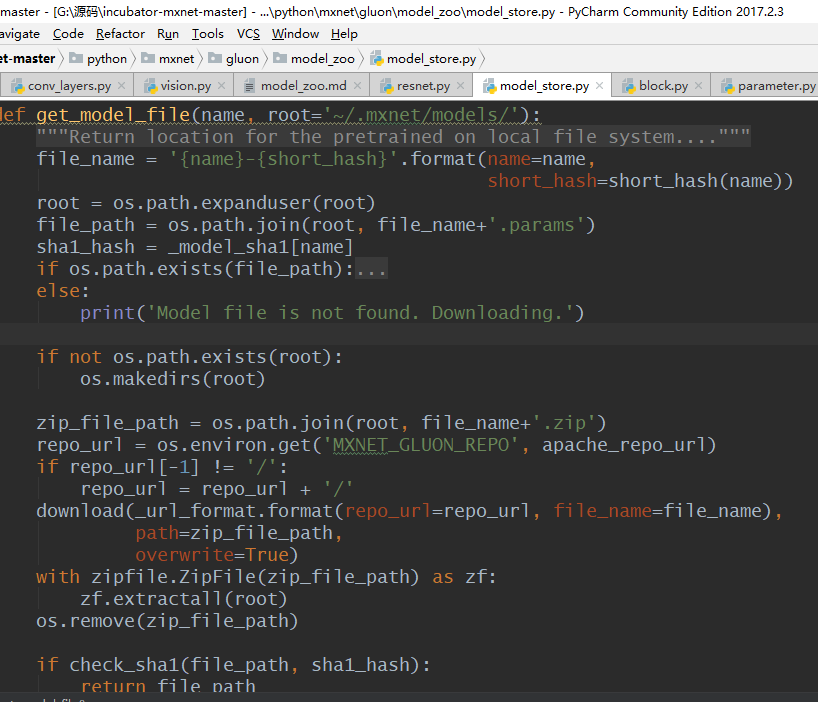
改下载环境变量！

repo\_url = os.environ.get('MXNET\_GLUON\_REPO', apache\_repo\_url)

在国内可以设置环境变量MXNET\_GLUON\_REPO为https://apache-mxnet.s3.cn-north-1.amazonaws.com.cn/来加速从Gluon下载数据集和预训练模型参数。

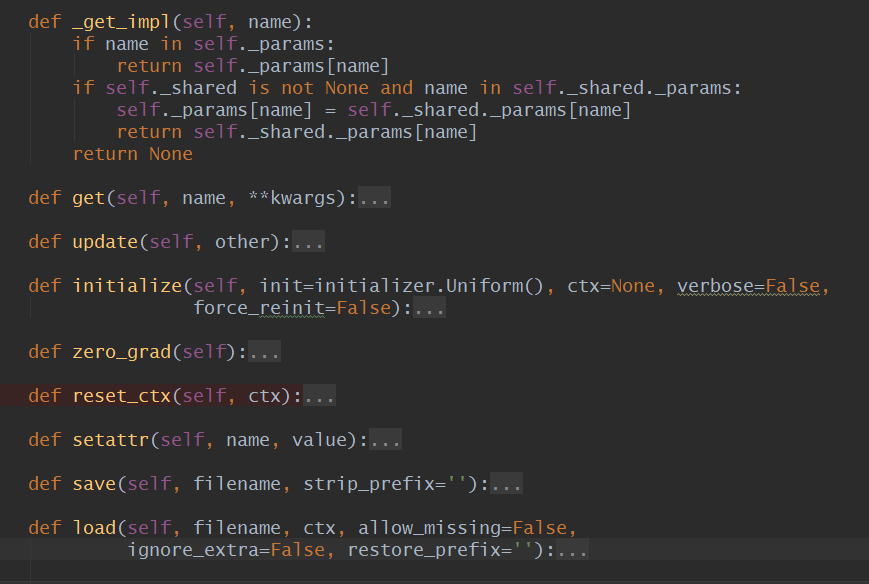
命令行下 当场就好使了





往下持续追到parameter.py





## 【】

## 【】

## 【】

## 【】

## 【】