# Caffe自定义数据层

## Caffe数据层分析

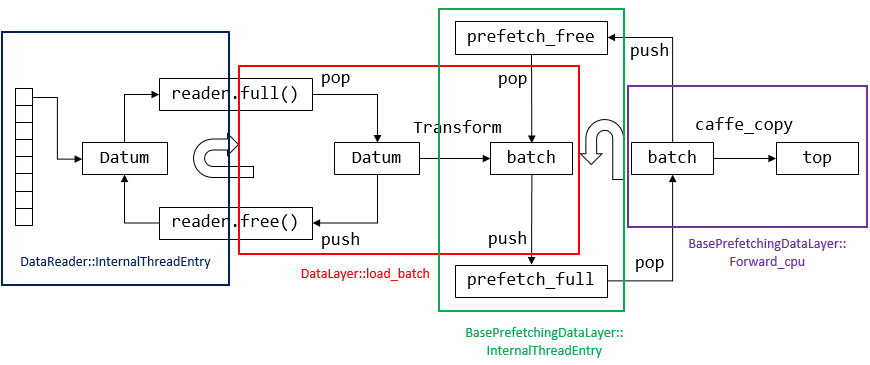


图 1 Caffe数据装载过程

以DataLayer为例，说明Caffe数据的装载过程。

数据的装载涉及到3个类，DataReader,DataLayer,BasePrefetchingDataLayer类，其中DataLayer是BasePrefetchingDataLayer的子类，Caffe中大量使用了类模板和虚函数，这样的好处是，如果父类和子类都共有同样名称的函数，或者父类的函数是纯虚函数，子类将其实现，使用父类指针调用该函数的时候，实际上是调用子类的函数，这就使得Caffe在框架上实现时，常常是用较为底层的父类搭建的，而实际函数执行过程在子类上。

在DataReader中，InternalThreadEntry是一个独立的线程函数，该函数内部将LMDB类型的数据读入到内存中，然后在一个循环中不断使用一个指针依次指向内存数据，将数据从装载到full的Datum队列中, Datum是一个以字符串形式保存的图像数据类。

BasePrefetchingDataLayer类中也维护了两个队列：prefetch\_free和prefetch\_full，其中保存的是Batch类数据，包括一个data和label,在该类的InternalThreadEntry函数中，将free中的batch指针pop出来，传给DataLayer的load\_batch装载batch数据，装载后push到full队列中。在这一过程中DataLayer的load\_batch函数是连接DataReader类和BasePrefetchingDataLayer类的桥梁，它将来自DataLayer的Datum对象经过Transform（其中包括白化、剪裁、镜向等操作）把数据保存到batch中，其实也就是prefetch\_free队列pop出来的batch地址上。

BasePrefetchingDataLayer类的Forward\_cpu中，prefetch\_full队列将batch数据pop出来，使用caffe\_copy函数将batch数据输入到网络中。

以上三个类的4个装载过程在训练过程中是一直执行着，直到训练结束。

## 自定义数据层

### 2.1 原理

我们自定义自己的数据层时，需要解决几个问题，数据如何载入到内存，如何输入到网络。下面使用一个1000\*2000大小，包括50\*100个20\*20的手写数字图像作为数据源，对lenet网络进行训练。

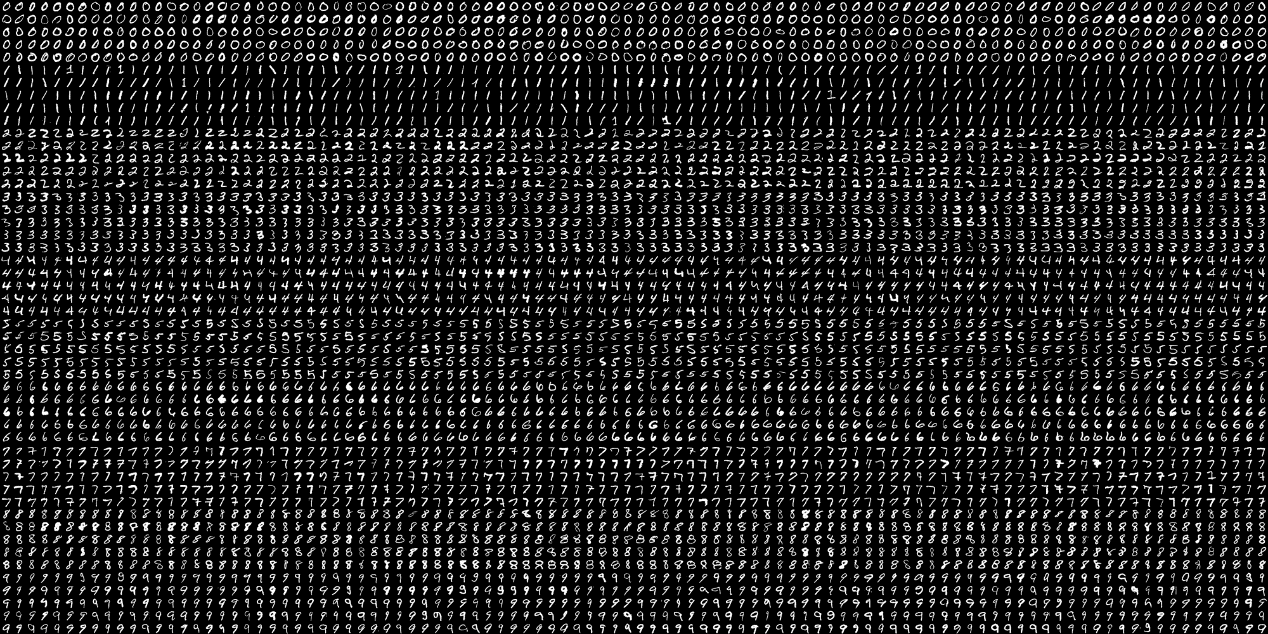


图 2 数据源图像

由于我们使用的是一幅图像，毫无疑问，我们需要将其裁剪成20\*20大小的子图像作为训练数据，我们在自己定义的数据层中实现这一过程。我们可以类比Caffe自带的ImageLayer来实现我们的数据层。

因为我们使用的是图像数据，而不是Datum类型数据，我们可以将图 1进行改造，结构如图 3所示：在将小图像及标签存放在一个vector向量中，在load\_batch函数中，将Mat类型的图像数据及标签经过Transform转成batch类型，当图像数据使用完后，游标再从向量第一个元素开始，这样完成数据的载入与传输。

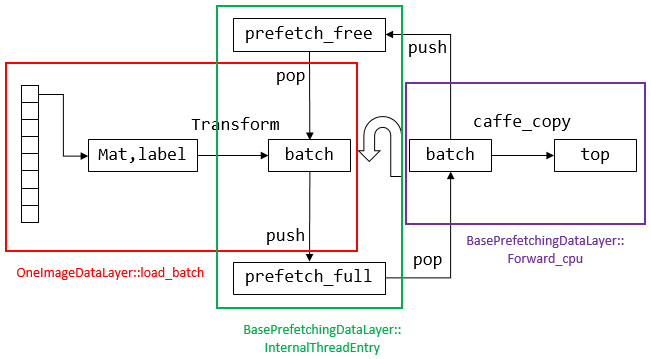


图 3 自定义数据层数据装载过程

### 实现

1. 在caffe.proto中添加层参数定义

我们的数据层应该包括以下信息，图像地址，行方向子图的数量，列方向子图的数量，子图的宽高大小，训练批的大小。我们定义的新层名字是OneImageDataLayer，层参数名为OneImageDataLayerParameter，为此我们在caffe.proto文件中添加以下内容：

|  |
| --- |
| //新的数据层参数  message OneImageDataLayerParameter{  optional uint32 iCols = 1;  optional uint32 iRows = 2;  optional uint32 iWidth = 4 [default = 0];  optional uint32 iHeight = 5 [default = 0];  optional string strLoadAddress = 6;  optional uint32 iBatchSize = 9;  }; |

并在LayerParameter中添加新层参数：

|  |
| --- |
| optional OneImageDataLayerParameter oneImage\_param = 201; |

1. 实现OneImageDataLayer类

添加one\_image\_data\_layer.hpp和one\_image\_data\_layer.cpp，实现OneImageDataLayer类，可以从caffe自带的image\_data\_hpp和image\_data.cpp修改而来。其中主要修改DataLayerSetUp和load\_batch函数。

DataLayerSetUp函数的实现，其中主要是读取层参数，载入图像，将图像切分保存到列表中，初始化prefetch\_的形状，prefetch\_是Batch类型，即是free和full队列中的成员。

|  |
| --- |
| template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**DataLayerSetUp**(**const vector**<**Blob**<**Dtype**>\*>&** bottom**,**  const vector**<**Blob**<**Dtype**>\*>&** top**)** **{**  iWidth **=** **this->**layer\_param\_**.**oneimage\_param**().**iwidth**();**  iHeight **=** **this->**layer\_param\_**.**oneimage\_param**().**iheight**();**  iCols **=** **this->**layer\_param\_**.**oneimage\_param**().**icols**();**  iRows **=** **this->**layer\_param\_**.**oneimage\_param**().**irows**();**  strLoadAddress **=** **this->**layer\_param\_**.**oneimage\_param**().**strloadaddress**();**  iBatchSize **=** **this->**layer\_param\_**.**oneimage\_param**().**ibatchsize**();**  dataSet**.**clear**();**  //载入图像  cv**::**Mat fullImage **=** cv**::**imread**(**strLoadAddress**,** 0**);**  CHECK**(**fullImage**.**data**)** **<<** "Load Image Failed!"**;**  **for** **(**int idxY **=** 0**;** idxY **<** iRows**;** idxY**++)**  **{**  **for** **(**int idxX **=** 0**;** idxX **<** iCols**;** idxX**++)**  **{**  cv**::**Rect rcPatch **=** cv**::**Rect**(**idxX **\*** iWidth**,** idxY **\*** iHeight**,** iWidth**,** iHeight**);**  cv**::**Mat matPatch **=** cv**::**Mat**(**fullImage**,** rcPatch**);**  int label **=** idxY **/** 5**;**  dataSet**.**push\_back**(**make\_pair**(**matPatch**.**clone**(),** label**));**  **}**  **}**  LOG**(**INFO**)** **<<** "Total Image Number: " **<<** dataSet**.**size**();**  ShuffleImages**();**  iImageIdx **=** 0**;**  // data  vector**<**int**>** top\_shape**(**4**);**  top\_shape**[**0**]** **=** iBatchSize**;**  top\_shape**[**1**]** **=** 1**;**  top\_shape**[**2**]** **=** iWidth**;**  top\_shape**[**3**]** **=** iHeight**;**  **for** **(**int i **=** 0**;** i **<** **this->**PREFETCH\_COUNT**;** **++**i**)**  **{**  **this->**prefetch\_**[**i**].**data\_**.**Reshape**(**top\_shape**);**  **}**  top**[**0**]->**Reshape**(**top\_shape**);**  LOG**(**INFO**)** **<<** "output data size: " **<<** top**[**0**]->**num**()** **<<** ","  **<<** top**[**0**]->**channels**()** **<<** "," **<<** top**[**0**]->**height**()** **<<** ","  **<<** top**[**0**]->**width**();**  // label  vector**<**int**>** label\_shape**(**1**,** iBatchSize**);**  top**[**1**]->**Reshape**(**label\_shape**);**  **for** **(**int i **=** 0**;** i **<** **this->**PREFETCH\_COUNT**;** **++**i**)**  **{**  **this->**prefetch\_**[**i**].**label\_**.**Reshape**(**label\_shape**);**  **}**  **}** |

ShuffleImages函数的实现：

|  |
| --- |
| template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**ShuffleImages**()** **{**  shuffle**(**dataSet**.**begin**(),** dataSet**.**end**());**  **}** |

Load\_batch函数的实现：

|  |
| --- |
| // This function is called on prefetch thread  template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**load\_batch**(**Batch**<**Dtype**>\*** batch**)** **{**  CPUTimer batch\_timer**;**  batch\_timer**.**Start**();**  double read\_time **=** 0**;**  double trans\_time **=** 0**;**  CPUTimer timer**;**  vector**<**int**>** top\_shape**(**4**);**  top\_shape**[**0**]** **=** iBatchSize**;**  top\_shape**[**1**]** **=** 1**;**  top\_shape**[**2**]** **=** iWidth**;**  top\_shape**[**3**]** **=** iHeight**;**  **this->**transformed\_data\_**.**Reshape**(**top\_shape**);**  batch**->**data\_**.**Reshape**(**top\_shape**);**  Dtype**\*** prefetch\_data **=** batch**->**data\_**.**mutable\_cpu\_data**();**  Dtype**\*** prefetch\_label **=** batch**->**label\_**.**mutable\_cpu\_data**();**  **for** **(**int item\_id **=** 0**;** item\_id **<** iBatchSize**;** **++**item\_id**)**  **{**  // get a blob  timer**.**Start**();**  CHECK\_GT**(**dataSet**.**size**(),** iImageIdx**);**  //取出图像数据和标签数据  cv**::**Mat cv\_img **=** dataSet**[**iImageIdx**].**first**;**  int label **=** dataSet**[**iImageIdx**].**second**;**  iImageIdx**++;**  read\_time **+=** timer**.**MicroSeconds**();**  timer**.**Start**();**  //传输图像数据和标签  int offset **=** batch**->**data\_**.**offset**(**item\_id**);**  **this->**transformed\_data\_**.**set\_cpu\_data**(**prefetch\_data **+** offset**);**  **this->**data\_transformer\_**->**Transform**(**cv\_img**,** **&(this->**transformed\_data\_**));**  prefetch\_label**[**item\_id**]** **=** label**;**  trans\_time **+=** timer**.**MicroSeconds**();**  **if** **(**iImageIdx **>=** dataSet**.**size**())** **{**  DLOG**(**INFO**)** **<<** "Restarting data prefetching from start."**;**  iImageIdx **=** 0**;**  ShuffleImages**();**  **}**  **}**  batch\_timer**.**Stop**();**  DLOG**(**INFO**)** **<<** "Prefetch batch: " **<<** batch\_timer**.**MilliSeconds**()** **<<** " ms."**;**  DLOG**(**INFO**)** **<<** " Read time: " **<<** read\_time **/** 1000 **<<** " ms."**;**  DLOG**(**INFO**)** **<<** "Transform time: " **<<** trans\_time **/** 1000 **<<** " ms."**;**  **}** |

在load\_batch函数中transformed\_data\_相当于一个中介，它将数据指针设置到prefetch\_data **+** offset位置，使用**this->**data\_transformer\_**->**Transform将图像数据进行预处理并保存到transformed\_data上，其结果是将数据保存到了prefetch\_data **+** offset上，由此完成了数据从Mat类型到batch类型的转换，最终被push到prefetch\_full中。

除了以前关键步骤外，还有一些细节如，返回层的类型函数type，层的注册等，参见附录代码。

## 测试

为了测试自定义的层，将minist例子稍做修改.

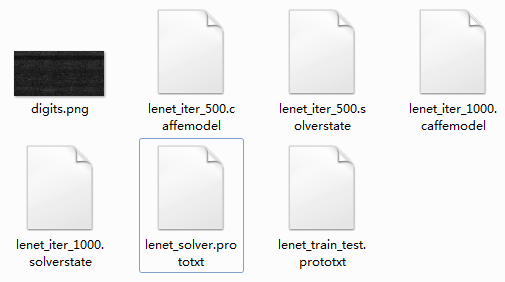
修改lenet\_train\_test.prototxt文件，数据层改成我们的OneImageData层，由于这里只有一张图像，我把测试集的图像也用训练集代替：

|  |
| --- |
| name: "LeNet"  layer {  name: "mnist"  type: "OneImageData"  top: "data"  top: "label"  include {  phase: TRAIN  }  transform\_param {  scale: 0.00390625  }  oneImage\_param {  iCols: 100  iRows: 50  iWidth: 20  iHeight: 20  iBatchSize: 20  strLoadAddress: "C:/caffe/examples/one\_image/digits.png"  }  }  layer {  name: "mnist"  type: "OneImageData"  top: "data"  top: "label"  include {  phase: TEST  }  transform\_param {  scale: 0.00390625  }  oneImage\_param {  iCols: 100  iRows: 50  iWidth: 20  iHeight: 20  iBatchSize: 9  strLoadAddress: "C:/caffe/examples/one\_image/digits.png"  }  }  //以下部分与原文件相同，略 |

修改lenet\_solve.prototxt文件：

|  |
| --- |
| # The train/test net protocol buffer definition  net: "C:/caffe/examples/one\_image/lenet\_train\_test.prototxt"  # test\_iter specifies how many forward passes the test should carry out.  # In the case of MNIST, we have test batch size 100 and 100 test iterations,  # covering the full 10,000 testing images.  test\_iter: 100  # Carry out testing every 500 training iterations.  test\_interval: 500  # The base learning rate, momentum and the weight decay of the network.  base\_lr: 0.01  momentum: 0.9  weight\_decay: 0.0005  # The learning rate policy  lr\_policy: "inv"  gamma: 0.0001  power: 0.75  # Display every 100 iterations  display: 100  # The maximum number of iterations  max\_iter: 1000  # snapshot intermediate results  snapshot: 500  snapshot\_prefix: "C:/caffe/examples/one\_image/lenet"  # solver mode: CPU or GPU  solver\_mode: CPU |

开始训练，训练结束后生成以下文件：



## 附录

### 4.1 one\_image\_data\_layer.hpp

|  |
| --- |
| #ifndef CAFFE\_ONE\_IMAGE\_DATA\_LAYER\_HPP\_  #define CAFFE\_ONE\_IMAGE\_DATA\_LAYER\_HPP\_  #include <string>  #include <utility>  #include <vector>  #include "caffe/blob.hpp"  #include "caffe/data\_transformer.hpp"  #include "caffe/internal\_thread.hpp"  #include "caffe/layer.hpp"  #include "caffe/layers/base\_data\_layer.hpp"  #include "caffe/proto/caffe.pb.h"  **namespace** caffe **{**  template **<**typename Dtype**>**  class OneImageDataLayer **:** public BasePrefetchingDataLayer**<**Dtype**>** **{**  public**:**  explicit OneImageDataLayer**(**const LayerParameter**&** param**)**  **:** BasePrefetchingDataLayer**<**Dtype**>(**param**)** **{}**  virtual **~**OneImageDataLayer**();**  virtual void DataLayerSetUp**(**const vector**<**Blob**<**Dtype**>\*>&** bottom**,**  const vector**<**Blob**<**Dtype**>\*>&** top**);**  virtual inline const char**\*** type**()** const **{** **return** "OneImageData"**;** **}**  virtual inline int ExactNumBottomBlobs**()** const **{** **return** 0**;** **}**  virtual inline int ExactNumTopBlobs**()** const **{** **return** 2**;** **}**  protected**:**  shared\_ptr**<**Caffe**::**RNG**>** prefetch\_rng\_**;**  virtual void ShuffleImages**();**  virtual void load\_batch**(**Batch**<**Dtype**>\*** batch**);**  vector**<**std**::**pair**<**cv**::**Mat**,** int**>** **>** dataSet**;**  int iImageIdx**;**  int iWidth**;**  int iHeight**;**  int iCols**;**  int iRows**;**  string strLoadAddress**;**  int iBatchSize;  };  } // namespace caffe  #endif // CAFFE\_IMAGE\_DATA\_LAYER\_HPP\_ |

### 4.2 one\_image\_data\_layer.cpp

|  |
| --- |
| #ifdef USE\_OPENCV  #include <opencv2/core/core.hpp>  #include <fstream> // NOLINT(readability/streams)  #include <iostream> // NOLINT(readability/streams)  #include <string>  #include <utility>  #include <vector>  #include "caffe/data\_transformer.hpp"  #include "caffe/layers/base\_data\_layer.hpp"  #include "caffe/layers/one\_image\_data\_layer.hpp"  #include "caffe/util/benchmark.hpp"  #include "caffe/util/io.hpp"  #include "caffe/util/math\_functions.hpp"  #include "caffe/util/rng.hpp"  #include <opencv2\highgui.hpp>  **namespace** caffe **{**  template **<**typename Dtype**>**  OneImageDataLayer**<**Dtype**>::~**OneImageDataLayer**<**Dtype**>()** **{**  **this->**StopInternalThread**();**  **}**  template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**DataLayerSetUp**(**const vector**<**Blob**<**Dtype**>\*>&** bottom**,**  const vector**<**Blob**<**Dtype**>\*>&** top**)** **{**  iWidth **=** **this->**layer\_param\_**.**oneimage\_param**().**iwidth**();**  iHeight **=** **this->**layer\_param\_**.**oneimage\_param**().**iheight**();**  iCols **=** **this->**layer\_param\_**.**oneimage\_param**().**icols**();**  iRows **=** **this->**layer\_param\_**.**oneimage\_param**().**irows**();**  strLoadAddress **=** **this->**layer\_param\_**.**oneimage\_param**().**strloadaddress**();**  iBatchSize **=** **this->**layer\_param\_**.**oneimage\_param**().**ibatchsize**();**  dataSet**.**clear**();**  //载入图像  cv**::**Mat fullImage **=** cv**::**imread**(**strLoadAddress**,** 0**);**  CHECK**(**fullImage**.**data**)** **<<** "Load Image Failed!"**;**  **for** **(**int idxY **=** 0**;** idxY **<** iRows**;** idxY**++)**  **{**  **for** **(**int idxX **=** 0**;** idxX **<** iCols**;** idxX**++)**  **{**  cv**::**Rect rcPatch **=** cv**::**Rect**(**idxX **\*** iWidth**,** idxY **\*** iHeight**,** iWidth**,** iHeight**);**  cv**::**Mat matPatch **=** cv**::**Mat**(**fullImage**,** rcPatch**);**  int label **=** idxY **/** 5**;**  dataSet**.**push\_back**(**make\_pair**(**matPatch**.**clone**(),** label**));**  **}**  **}**  LOG**(**INFO**)** **<<** "Total Image Number: " **<<** dataSet**.**size**();**  ShuffleImages**();**  iImageIdx **=** 0**;**  // data  vector**<**int**>** top\_shape**(**4**);**  top\_shape**[**0**]** **=** iBatchSize**;**  top\_shape**[**1**]** **=** 1**;**  top\_shape**[**2**]** **=** iWidth**;**  top\_shape**[**3**]** **=** iHeight**;**  **for** **(**int i **=** 0**;** i **<** **this->**PREFETCH\_COUNT**;** **++**i**)**  **{**  **this->**prefetch\_**[**i**].**data\_**.**Reshape**(**top\_shape**);**  **}**  top**[**0**]->**Reshape**(**top\_shape**);**  LOG**(**INFO**)** **<<** "output data size: " **<<** top**[**0**]->**num**()** **<<** ","  **<<** top**[**0**]->**channels**()** **<<** "," **<<** top**[**0**]->**height**()** **<<** ","  **<<** top**[**0**]->**width**();**  // label  vector**<**int**>** label\_shape**(**1**,** iBatchSize**);**  top**[**1**]->**Reshape**(**label\_shape**);**  **for** **(**int i **=** 0**;** i **<** **this->**PREFETCH\_COUNT**;** **++**i**)**  **{**  **this->**prefetch\_**[**i**].**label\_**.**Reshape**(**label\_shape**);**  **}**  **}**  template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**ShuffleImages**()** **{**  //shuffle(lines\_.begin(), lines\_.end(), prefetch\_rng);  shuffle**(**dataSet**.**begin**(),** dataSet**.**end**());**  **}**  // This function is called on prefetch thread  template **<**typename Dtype**>**  void OneImageDataLayer**<**Dtype**>::**load\_batch**(**Batch**<**Dtype**>\*** batch**)** **{**  CPUTimer batch\_timer**;**  batch\_timer**.**Start**();**  double read\_time **=** 0**;**  double trans\_time **=** 0**;**  CPUTimer timer**;**  vector**<**int**>** top\_shape**(**4**);**  top\_shape**[**0**]** **=** iBatchSize**;**  top\_shape**[**1**]** **=** 1**;**  top\_shape**[**2**]** **=** iWidth**;**  top\_shape**[**3**]** **=** iHeight**;**  **this->**transformed\_data\_**.**Reshape**(**top\_shape**);**  batch**->**data\_**.**Reshape**(**top\_shape**);**  Dtype**\*** prefetch\_data **=** batch**->**data\_**.**mutable\_cpu\_data**();**  Dtype**\*** prefetch\_label **=** batch**->**label\_**.**mutable\_cpu\_data**();**  **for** **(**int item\_id **=** 0**;** item\_id **<** iBatchSize**;** **++**item\_id**)**  **{**  // get a blob  timer**.**Start**();**  CHECK\_GT**(**dataSet**.**size**(),** iImageIdx**);**  //取出图像数据和标签数据  cv**::**Mat cv\_img **=** dataSet**[**iImageIdx**].**first**;**  int label **=** dataSet**[**iImageIdx**].**second**;**  iImageIdx**++;**  read\_time **+=** timer**.**MicroSeconds**();**  timer**.**Start**();**  //传输图像数据和标签  int offset **=** batch**->**data\_**.**offset**(**item\_id**);**  **this->**transformed\_data\_**.**set\_cpu\_data**(**prefetch\_data **+** offset**);**  **this->**data\_transformer\_**->**Transform**(**cv\_img**,** **&(this->**transformed\_data\_**));**  prefetch\_label**[**item\_id**]** **=** label**;**  trans\_time **+=** timer**.**MicroSeconds**();**  **if** **(**iImageIdx **>=** dataSet**.**size**())** **{**  DLOG**(**INFO**)** **<<** "Restarting data prefetching from start."**;**  iImageIdx **=** 0**;**  ShuffleImages**();**  **}**  **}**  batch\_timer**.**Stop**();**  DLOG**(**INFO**)** **<<** "Prefetch batch: " **<<** batch\_timer**.**MilliSeconds**()** **<<** " ms."**;**  DLOG**(**INFO**)** **<<** " Read time: " **<<** read\_time **/** 1000 **<<** " ms."**;**  DLOG**(**INFO**)** **<<** "Transform time: " **<<** trans\_time **/** 1000 **<<** " ms."**;**  **}**  INSTANTIATE\_CLASS**(**OneImageDataLayer**);**  REGISTER\_LAYER\_CLASS**(**OneImageData**);**  **}** // namespace caffe  #endif // USE\_OPENCV |