Caffe提供了在CPU以及GPU上的快速卷积神经网络实现，同时提供训练算法，使用NVIDIA K40或Titan GPU可以1天完成多于40,000,000张图片的训练。

Caffe：

Blod：基础数据结构

Layer：网络的基本单元

Net：用于各Layer层的组合搭建

caffe训练数据：

将图片数据集格式转为lmdb格式

搭建自己的网络MNIST在caffe上进行训练与学习

全过程需要有这么几个文件：训练数据集mnist\_train\_lmdb，测试数据集mnist\_test\_lmdb，层定义lenet.prototxt，层在训练时所用网络参数lenet\_train\_test.prototxt，层训练参数lenet\_solver.prototxt，网络开始训练的脚本train\_lenet.sh

1. 准备数据库:MNIST 手写字体库

运行以下指令下载:字体库

$ cd CAFFE\_ROOT/data/mnist

$ ./get\_mnist.sh

$ cd CAFFE\_ROOT/examples/mnist

$ ./create\_mnist.sh

这注意根据实际修改下create\_mnist.sh里的路径。并且需要将convert\_mnist\_data.bin改成convert\_mnist\_data

运行之后，CAFFE\_ROOT/examples/mnist 文件夹下会有 mnist\_train\_lmdb 和 mnist\_test\_lmdb两个文件夹

2. 训练数据

文件CAFFE\_ROOT/examples/mnist/lenet.prototxt中定义了训练时所用的层。

caffe采用Google Protobuf，主要作用是把某种数据结构的信息,以某种格式保存起来。主要用于数据存储、传输协议格式等场 合。

文件CAFFE\_ROOT/examples/mnist/lenet\_train\_test.prototxt中定义了训练时所用的网络参数

cnn网络一般分为10层，分别为：数据层DATA，卷积层CONVOLUTION，池化层POOLING， 卷积层CONVOLUTION，池化层POOLING，全连接层INNER\_PRODUCT，RELU， 全连接层INNER\_PRODUCT，ACCURACY，SOFTMAX\_LOSS

我们以其中一个单元为例来讲解，这是第一个卷积层的样子

name: “LeNet”

layers {

name: "mnist"

type: DATA

top: "data"

top: "label"

data\_param {

source: "examples/mnist/mnist\_train\_lmdb"

backend: LMDB

batch\_size: 64

}

transform\_param {

scale: 0.00390625

}

include: { phase: TRAIN }

}

layers {

name: "mnist"

type: DATA

top: "data"

top: "label"

data\_param {

source: "examples/mnist/mnist\_test\_lmdb"

backend: LMDB

batch\_size: 100

}

transform\_param {

scale: 0.00390625

}

include: { phase: TEST }

}

layers {

name: "conv1"

type: CONVOLUTION

bottom: "data"

top: "conv1"

blobs\_lr: 1

blobs\_lr: 2

convolution\_param {

num\_output: 20

kernel\_size: 5

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

初始的name用于给网络命名。

DATA层中，source指定了训练数据和测试数据的目录。

CONVOLUTION层中，name定义了这层的名字，type为这层的类型，bottom指明这层前面使用data层，top指明后面生成conv1的Blob空间，第一个blobs\_lr为权重学习率设置为与求解器给出的学习率相等，第二个blobs\_lr设置权重学习律是偏置 学习率的两倍。convolution\_param为卷积层参数，num\_output为输出单元数，即该层滤波器的个数，kernel\_size为卷积核的大小，stride为步长，weight\_filler设定网络允许用随机值初始化权重和偏置值，type为表示使用 xavier 算法自动确定基于输入和输出神经元数量的初始规模，bias\_filler设置偏置值初始化为常数,默认为 0。

文件CAFFE\_ROOT/examples/mnist/lenet\_solver.prototxt中定义了训练参数

net: "/home/visenze/Project/caffe/examples/mnist/lenet\_train\_test.prototxt"

test\_iter: 100 ，是训练的批次,设为 100,迭代次数 100 次,这样,就覆盖了10000 张(100\*100) 测试图片

test\_interval: 500 ，每迭代次数 500 次测试一次

base\_lr: 0.01 ，学习率

momentum: 0.9 ， 动量

weight\_decay: 0.0005 ，权重的递减

lr\_policy: "inv" ，学习政策 inv,注意的是,cifar10 类用固定学习率,imagenet 用每步递减学习率

gamma: 0.0001

power: 0.75

display: 100 ，每迭代 100 次显示一次

max\_iter: 10000 ，最大迭代次数 10000 次

snapshot: 5000 ，每 5000 次迭代存储一次数据到电脑

snapshot\_prefix: "examples/mnist/lenet" ，存储文件名字为 lenet

solver\_mode: CPU ，确定用 CPU 训练还是是用 GPU 训练

网络开始训练的脚本train\_lenet.sh

用于开始整个训练过程，会把执行的信息显示出来其内容为：

#!/usr/bin/env sh

/home/visenze/Project/caffe-build/tools/caffe train –solver=/home/visenze/Project/caffe/examples/mnist/lenet\_solver.prototxt

最后在终端执行指令，开始训练:

$ cd CAFFE\_ROOT

$ ./examples/mnist /train\_lenet.sh

然后出现类似以下的信息，这是搭建模型的相关信息：

I0424 12:21:32.370805 18961 net.cpp:56] Memory required for data: 0

I0424 12:21:32.370852 18961 net.cpp:67] Creating Layer mnist

I0424 12:21:32.370867 18961 net.cpp:356] mnist -> data

I0424 12:21:32.370893 18961 net.cpp:356] mnist -> label

I0424 12:21:32.370911 18961 net.cpp:96] Setting up mnist

I0424 12:21:32.371245 18961 data\_layer.cpp:81] Opening lmdb examples/mnist/mnist\_test\_lmdb

I0424 12:21:32.371721 18961 data\_layer.cpp:141] output data size: 100,1,28,28

I0424 12:21:32.371824 18961 base\_data\_layer.cpp:64] Initializing prefetch

I0424 12:21:32.371848 18961 base\_data\_layer.cpp:66] Prefetch initialized.

I0424 12:21:32.371860 18961 net.cpp:103] Top shape: 100 1 28 28 (78400)

I0424 12:21:32.371870 18961 net.cpp:103] Top shape: 100 1 1 1 (100)

I0424 12:21:32.371878 18961 net.cpp:113] Memory required for data: 314000

I0424 12:21:32.371897 18961 net.cpp:67] Creating Layer label\_mnist\_1\_split

I0424 12:21:32.371907 18961 net.cpp:394] label\_mnist\_1\_split <- label

I0424 12:21:32.371924 18961 net.cpp:356] label\_mnist\_1\_split -> label\_mnist\_1\_split\_0

I0424 12:21:32.371947 18961 net.cpp:356] label\_mnist\_1\_split -> label\_mnist\_1\_split\_1

I0424 12:21:32.371961 18961 net.cpp:96] Setting up label\_mnist\_1\_split

I0424 12:21:32.371975 18961 net.cpp:103] Top shape: 100 1 1 1 (100)

I0424 12:21:32.371984 18961 net.cpp:103] Top shape: 100 1 1 1 (100)

I0424 12:21:32.371991 18961 net.cpp:113] Memory required for data: 314800

I0424 12:21:32.372009 18961 net.cpp:67] Creating Layer conv1

I0424 12:21:32.372019 18961 net.cpp:394] conv1 <- data

I0424 12:21:32.372038 18961 net.cpp:356] conv1 -> conv1

I0424 12:21:32.372066 18961 net.cpp:96] Setting up conv1

I0424 12:21:32.372133 18961 net.cpp:103] Top shape: 100 20 24 24 (1152000)

I0424 12:21:32.372144 18961 net.cpp:113] Memory required for data: 4922800

I0424 12:21:32.372169 18961 net.cpp:67] Creating Layer pool1

I0424 12:21:32.372181 18961 net.cpp:394] pool1 <- conv1

I0424 12:21:32.372197 18961 net.cpp:356] pool1 -> pool1

I0424 12:21:32.372213 18961 net.cpp:96] Setting up pool1

I0424 12:21:32.372227 18961 net.cpp:103] Top shape: 100 20 12 12 (288000)

I0424 12:21:32.372236 18961 net.cpp:113] Memory required for data: 6074800

I0424 12:21:32.372266 18961 net.cpp:67] Creating Layer conv2

I0424 12:21:32.372277 18961 net.cpp:394] conv2 <- pool1

I0424 12:21:32.372294 18961 net.cpp:356] conv2 -> conv2

I0424 12:21:32.372311 18961 net.cpp:96] Setting up conv2

I0424 12:21:32.373957 18961 net.cpp:103] Top shape: 100 50 8 8 (320000)

I0424 12:21:32.373970 18961 net.cpp:113] Memory required for data: 7354800

I0424 12:21:32.373994 18961 net.cpp:67] Creating Layer pool2

I0424 12:21:32.374006 18961 net.cpp:394] pool2 <- conv2

I0424 12:21:32.374023 18961 net.cpp:356] pool2 -> pool2

I0424 12:21:32.374040 18961 net.cpp:96] Setting up pool2

I0424 12:21:32.374053 18961 net.cpp:103] Top shape: 100 50 4 4 (80000)

I0424 12:21:32.374061 18961 net.cpp:113] Memory required for data: 7674800

I0424 12:21:32.374075 18961 net.cpp:67] Creating Layer ip1

I0424 12:21:32.374084 18961 net.cpp:394] ip1 <- pool2

I0424 12:21:32.374101 18961 net.cpp:356] ip1 -> ip1

I0424 12:21:32.374119 18961 net.cpp:96] Setting up ip1

I0424 12:21:32.401000 18961 net.cpp:103] Top shape: 100 500 1 1 (50000)

I0424 12:21:32.401021 18961 net.cpp:113] Memory required for data: 7874800

I0424 12:21:32.401048 18961 net.cpp:67] Creating Layer relu1

I0424 12:21:32.401060 18961 net.cpp:394] relu1 <- ip1

I0424 12:21:32.401080 18961 net.cpp:345] relu1 -> ip1 (in-place)

I0424 12:21:32.401095 18961 net.cpp:96] Setting up relu1

I0424 12:21:32.401105 18961 net.cpp:103] Top shape: 100 500 1 1 (50000)

I0424 12:21:32.401113 18961 net.cpp:113] Memory required for data: 8074800

I0424 12:21:32.401131 18961 net.cpp:67] Creating Layer ip2

I0424 12:21:32.401141 18961 net.cpp:394] ip2 <- ip1

I0424 12:21:32.401160 18961 net.cpp:356] ip2 -> ip2

I0424 12:21:32.401177 18961 net.cpp:96] Setting up ip2

I0424 12:21:32.401463 18961 net.cpp:103] Top shape: 100 10 1 1 (1000)

I0424 12:21:32.401471 18961 net.cpp:113] Memory required for data: 8078800

I0424 12:21:32.401482 18961 net.cpp:67] Creating Layer ip2\_ip2\_0\_split

I0424 12:21:32.401489 18961 net.cpp:394] ip2\_ip2\_0\_split <- ip2

I0424 12:21:32.401500 18961 net.cpp:356] ip2\_ip2\_0\_split -> ip2\_ip2\_0\_split\_0

I0424 12:21:32.401514 18961 net.cpp:356] ip2\_ip2\_0\_split -> ip2\_ip2\_0\_split\_1

I0424 12:21:32.401523 18961 net.cpp:96] Setting up ip2\_ip2\_0\_split

I0424 12:21:32.401531 18961 net.cpp:103] Top shape: 100 10 1 1 (1000)

I0424 12:21:32.401536 18961 net.cpp:103] Top shape: 100 10 1 1 (1000)

I0424 12:21:32.401540 18961 net.cpp:113] Memory required for data: 8086800

I0424 12:21:32.401549 18961 net.cpp:67] Creating Layer accuracy

I0424 12:21:32.401556 18961 net.cpp:394] accuracy <- ip2\_ip2\_0\_split\_0

I0424 12:21:32.401564 18961 net.cpp:394] accuracy <- label\_mnist\_1\_split\_0

I0424 12:21:32.401573 18961 net.cpp:356] accuracy -> accuracy

I0424 12:21:32.401582 18961 net.cpp:96] Setting up accuracy

I0424 12:21:32.401592 18961 net.cpp:103] Top shape: 1 1 1 1 (1)

I0424 12:21:32.401595 18961 net.cpp:113] Memory required for data: 8086804

I0424 12:21:32.401604 18961 net.cpp:67] Creating Layer loss

I0424 12:21:32.401609 18961 net.cpp:394] loss <- ip2\_ip2\_0\_split\_1

I0424 12:21:32.401618 18961 net.cpp:394] loss <- label\_mnist\_1\_split\_1

I0424 12:21:32.401625 18961 net.cpp:356] loss -> loss

I0424 12:21:32.401634 18961 net.cpp:96] Setting up loss

I0424 12:21:32.401648 18961 net.cpp:103] Top shape: 1 1 1 1 (1)

I0424 12:21:32.401653 18961 net.cpp:109] with loss weight 1

I0424 12:21:32.401659 18961 net.cpp:113] Memory required for data: 8086808

I0424 12:21:32.401664 18961 net.cpp:170] loss needs backward computation.

I0424 12:21:32.401670 18961 net.cpp:172] accuracy does not need backward computation.

I0424 12:21:32.401675 18961 net.cpp:170] ip2\_ip2\_0\_split needs backward computation.

I0424 12:21:32.401680 18961 net.cpp:170] ip2 needs backward computation.

I0424 12:21:32.401685 18961 net.cpp:170] relu1 needs backward computation.

I0424 12:21:32.401690 18961 net.cpp:170] ip1 needs backward computation.

I0424 12:21:32.401695 18961 net.cpp:170] pool2 needs backward computation.

I0424 12:21:32.401700 18961 net.cpp:170] conv2 needs backward computation.

I0424 12:21:32.401705 18961 net.cpp:170] pool1 needs backward computation.

I0424 12:21:32.401720 18961 net.cpp:170] conv1 needs backward computation.

I0424 12:21:32.401726 18961 net.cpp:172] label\_mnist\_1\_split does not need backward computation.

I0424 12:21:32.401731 18961 net.cpp:172] mnist does not need backward computation.

I0424 12:21:32.401734 18961 net.cpp:208] This network produces output accuracy

I0424 12:21:32.401741 18961 net.cpp:208] This network produces output loss

I0424 12:21:32.401757 18961 net.cpp:467] Collecting Learning Rate and Weight Decay.

I0424 12:21:32.401767 18961 net.cpp:219] Network initialization done.

接着：

I0424 12:21:32.401770 18961 net.cpp:220] Memory required for data: 8086808

I0424 12:21:32.401806 18961 solver.cpp:41] Solver scaffolding done.

I0424 12:21:32.401814 18961 solver.cpp:160] Solving LeNet

I0424 12:21:32.401819 18961 solver.cpp:161] Learning Rate Policy: inv

I0424 12:21:32.401841 18961 solver.cpp:264] Iteration 0, Testing net (#0)

之后，训练开始

I0424 12:21:32.401850 18961 net.cpp:652] Copying source layer mnist

I0424 12:21:32.401855 18961 net.cpp:652] Copying source layer conv1

I0424 12:21:32.401866 18961 net.cpp:652] Copying source layer pool1

I0424 12:21:32.401871 18961 net.cpp:652] Copying source layer conv2

I0424 12:21:32.401876 18961 net.cpp:652] Copying source layer pool2

I0424 12:21:32.401880 18961 net.cpp:652] Copying source layer ip1

I0424 12:21:32.401947 18961 net.cpp:652] Copying source layer relu1

I0424 12:21:32.401954 18961 net.cpp:652] Copying source layer ip2

I0424 12:21:32.401959 18961 net.cpp:652] Copying source layer loss

I0424 12:21:40.045995 19069 data\_layer.cpp:242] Restarting data prefetching from start.

I0424 12:21:40.202261 18961 solver.cpp:315] Test net output #0: accuracy = 0.14

I0424 12:21:40.202314 18961 solver.cpp:315] Test net output #1: loss = 2.30171 (\* 1 = 2.30171 loss)

I0424 12:21:40.327651 18961 solver.cpp:209] Iteration 0, loss = 2.30095

I0424 12:21:40.327702 18961 solver.cpp:224] Train net output #0: loss = 2.30095 (\* 1 = 2.30095 loss)

I0424 12:21:40.327728 18961 solver.cpp:445] Iteration 0, lr = 0.01

I0424 12:21:51.620446 18961 solver.cpp:209] Iteration 100, loss = 0.333897

I0424 12:21:51.620501 18961 solver.cpp:224] Train net output #0: loss = 0.333897 (\* 1 = 0.333897 loss)

I0424 12:21:51.620517 18961 solver.cpp:445] Iteration 100, lr = 0.00992565

I0424 12:22:02.953106 18961 solver.cpp:209] Iteration 200, loss = 0.159022

I0424 12:22:02.953199 18961 solver.cpp:224] Train net output #0: loss = 0.159022 (\* 1 = 0.159022 loss)

I0424 12:22:02.953217 18961 solver.cpp:445] Iteration 200, lr = 0.00985258

I0424 12:22:14.099150 18961 solver.cpp:209] Iteration 300, loss = 0.222133

I0424 12:22:14.099205 18961 solver.cpp:224] Train net output #0: loss = 0.222132 (\* 1 = 0.222132 loss)

I0424 12:22:14.099221 18961 solver.cpp:445] Iteration 300, lr = 0.00978075

I0424 12:22:25.620846 18961 solver.cpp:209] Iteration 400, loss = 0.114809

I0424 12:22:25.620977 18961 solver.cpp:224] Train net output #0: loss = 0.114809 (\* 1 = 0.114809 loss)

I0424 12:22:25.621038 18961 solver.cpp:445] Iteration 400, lr = 0.00971013

I0424 12:22:36.577474 18961 solver.cpp:264] Iteration 500, Testing net (#0)

最后

I0424 17:14:19.226507 20188 data\_layer.cpp:242] Restarting data prefetching from start.

I0424 17:14:19.583953 14272 solver.cpp:315] Test net output #0: accuracy = 0.9894

I0424 17:14:19.585851 14272 solver.cpp:315] Test net output #1: loss = 0.0343618 (\* 1 = 0.0343618 loss)

I0424 17:14:19.839411 14272 solver.cpp:209] Iteration 9500, loss = 0.00570374

I0424 17:14:19.839494 14272 solver.cpp:224] Train net output #0: loss = 0.00570366 (\* 1 = 0.00570366 loss)

I0424 17:14:19.839530 14272 solver.cpp:445] Iteration 9500, lr = 0.00606002

I0424 17:14:35.312465 14272 solver.cpp:209] Iteration 9600, loss = 0.00309575

I0424 17:14:35.312504 14272 solver.cpp:224] Train net output #0: loss = 0.00309566 (\* 1 = 0.00309566 loss)

I0424 17:14:35.312515 14272 solver.cpp:445] Iteration 9600, lr = 0.00603682

I0424 17:14:53.018973 14272 solver.cpp:209] Iteration 9700, loss = 0.00394589

I0424 17:14:53.019070 14272 solver.cpp:224] Train net output #0: loss = 0.0039458 (\* 1 = 0.0039458 loss)

I0424 17:14:53.019089 14272 solver.cpp:445] Iteration 9700, lr = 0.00601382

I0424 17:15:10.420929 14272 solver.cpp:209] Iteration 9800, loss = 0.0313377

I0424 17:15:10.421025 14272 solver.cpp:224] Train net output #0: loss = 0.0313376 (\* 1 = 0.0313376 loss)

I0424 17:15:10.421063 14272 solver.cpp:445] Iteration 9800, lr = 0.00599102

I0424 17:15:45.175144 14272 solver.cpp:209] Iteration 9900, loss = 0.00742963

I0424 17:15:45.175288 14272 solver.cpp:224] Train net output #0: loss = 0.00742955 (\* 1 = 0.00742955 loss)

I0424 17:15:45.175330 14272 solver.cpp:445] Iteration 9900, lr = 0.00596843

I0424 17:16:17.482460 14272 net.cpp:799] Serializing 9 layers

I0424 17:16:17.488373 14272 solver.cpp:334] Snapshotting to examples/mnist/lenet\_iter\_10000.caffemodel

I0424 17:16:17.510210 14272 solver.cpp:342] Snapshotting solver state to examples/mnist/lenet\_iter\_10000.solverstate

I0424 17:16:17.660923 14272 solver.cpp:246] Iteration 10000, loss = 0.00491011

I0424 17:16:17.660964 14272 solver.cpp:264] Iteration 10000, Testing net (#0)

I0424 17:16:17.660976 14272 net.cpp:652] Copying source layer mnist

I0424 17:16:17.660985 14272 net.cpp:652] Copying source layer conv1

I0424 17:16:17.660996 14272 net.cpp:652] Copying source layer pool1

I0424 17:16:17.661005 14272 net.cpp:652] Copying source layer conv2

I0424 17:16:17.661013 14272 net.cpp:652] Copying source layer pool2

I0424 17:16:17.661026 14272 net.cpp:652] Copying source layer ip1

I0424 17:16:17.661036 14272 net.cpp:652] Copying source layer relu1

I0424 17:16:17.661043 14272 net.cpp:652] Copying source layer ip2

I0424 17:16:17.661052 14272 net.cpp:652] Copying source layer loss

I0424 17:16:38.622433 3322 data\_layer.cpp:242] Restarting data prefetching from start.

I0424 17:16:39.204105 14272 solver.cpp:315] Test net output #0: accuracy = 0.9912

I0424 17:16:39.205910 14272 solver.cpp:315] Test net output #1: loss = 0.0280239 (\* 1 = 0.0280239 loss)

I0424 17:16:39.205943 14272 solver.cpp:251] Optimization Done.

I0424 17:16:39.205960 14272 caffe.cpp:165] Optimization Done.

visenze@visenze-Latitude-E7450:~/Project/caffe$

出现Optimizatiion Done表示训练成功