## GPU版本

最终得到本地归档是caffe-gpu

Ubuntu16.04 caffe最新版 makefile

|  |
| --- |
| 安装依赖 hdf5 (apt 安装, 参考官网给出的依赖安装)  修改cmake.config  OPENCV\_VERSION := 3  USE\_CUDNN := 1  #CPU\_ONLY := 1  INCLUDE\_DIRS := $(PYTHON\_INCLUDE) /usr/local/include /usr/include/hdf5/serial  LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib /usr/lib /usr/lib/x86\_64-linux-gnu/hdf5/serial  DEBUG:=1  修改 MakeFile  CXXFLAGS += -pthread -fPIC $(COMMON\_FLAGS) $(WARNINGS) -std=c++11  NVCCFLAGS += -D\_FORCE\_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON\_FLAGS) -std=c++11  LINKFLAGS += -pthread -fPIC $(COMMON\_FLAGS) $(WARNINGS) -std=c++11  sudo make clean  sudo make all -j2  sudo make test  sudo make runtest  sudo make pycaffe |

## 调试运行

tools/caffe.cpp 入口文件

dnn@DNN:~/caffe$ ./examples/mnist/create\_mnist.sh

dnn@DNN:~/caffe$ ./examples/mnist/train\_lenet.sh

gdb --args build/tools/caffe train --solver examples/mnist/lenet\_solver.prototxt

CPU版本的这个调试不支持

./build/tools/caffe time --model='det/yolov3/yolov3.prototxt' --iterations=100 --gpu=0

Vscode 调试还不行  
(gdb) b src/caffe/layers/base\_conv\_layer.cpp:117

如果需要调试GPU程序，可以使用cuda-gdb

<https://github.com/leonardvandriel/caffe2_cpp_tutorial>

## Infer

./build/examples/cpp\_classification/classification.bin models/bvlc\_reference\_caffenet/deploy.prototxt models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel data/ilsvrc12/imagenet\_mean.binaryproto data/ilsvrc12/synset\_words.txt examples/images/cat.jpg

models/bvlc\_reference\_caffenet

## 源码解读

入口参考caffe.cpp

三个目录include、src和tools由于Caffe是深度学习模型，单步调试在跨越很多层的情况下显得过分繁琐且不易发现问题。个人建议，分层做单元测试，再组合各层通过日志输出来调试全局模型。个人的经验是，这种方式高效而且逻辑性强，得益于Caffe分层清晰,主要包括四个部分：

Blob:定义blob数据结构用来数据传输,包括输入输出,权重等

Layer:op

Net:网络骨架

Solver:网络求解优化策略

定义网络:rain.prototxt 配置训练参数solver.prototxt

优点:版本之间api稳

缺点:不支持自动求导,不支持模型并行,只支持数据并行

https://github.com/BUPTLdy/Caffe\_Code\_Analysis

工厂设计模式

## Caffe docker(cmake)

|  |
| --- |
| cmake\_minimum\_required(VERSION 2.8.7)  project(XORusingCAFFE C CXX)  set(Caffe\_INCLUDE\_DIRS /opt/caffe/include /opt/caffe/build/include /usr/local/cuda/include)  set(Caffe\_LIBRARIES caffe boost\_system glog)  include\_directories(${Caffe\_INCLUDE\_DIRS})  link\_directories(/opt/caffe/build/lib)  add\_executable(trainXOR trainXOR.cpp)  target\_link\_libraries(trainXOR ${Caffe\_LIBRARIES})  add\_executable(testXOR testXOR.cpp)  target\_link\_libraries(testXOR ${Caffe\_LIBRARIES}) |

其他

|  |
| --- |
| find\_package(OpenCV REQUIRED)  set(Caffe\_DIR "/home/kezunlin/program/caffe-wy/build/install/share/Caffe") # caffe-wy caffe  # for CaffeConfig.cmake/ caffe-config.cmake  find\_package(Caffe)# offical caffe : There is no Caffe\_INCLUDE\_DIRS and Caffe\_DEFINITIONS# refinedet caffe: OK.  add\_definitions(${Caffe\_DEFINITIONS})  MESSAGE( [Main] " Caffe\_INCLUDE\_DIRS = ${Caffe\_INCLUDE\_DIRS}")  MESSAGE( [Main] " Caffe\_DEFINITIONS = ${Caffe\_DEFINITIONS}")  MESSAGE( [Main] " Caffe\_LIBRARIES = ${Caffe\_LIBRARIES}") # caffe  MESSAGE( [Main] " Caffe\_CPU\_ONLY = ${Caffe\_CPU\_ONLY}")  MESSAGE( [Main] " Caffe\_HAVE\_CUDA = ${Caffe\_HAVE\_CUDA}")  MESSAGE( [Main] " Caffe\_HAVE\_CUDNN = ${Caffe\_HAVE\_CUDNN}")  include\_directories(${Caffe\_INCLUDE\_DIRS})  target\_link\_libraries(demo  ${OpenCV\_LIBS}  ${Caffe\_LIBRARIES}  ) |