This repository | Search Pull requests Issues Marketplace Gist

naibaf7 / **caffe**
forked from BVLC/caffe

Watch 18 Star 59 Fork 11,624

Code Issues 16 Pull requests 0 Projects 0 Wiki Insights

Caffe: a fast open framework for deep learning. With OpenCL and CUDA support. http://caffe.berkeleyvision.org/

5,101 commits **7** branches **10** releases **275** contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 1067 commits ahead of BVLC:master. Pull request Compare

gongzg Move half.hpp's license to 3rdparty/half. Latest commit e38c619 4 days ago

.github	Add Github issue template to curb misuse.	8 months ago
android	android compilation support	5 months ago
cmake	Lint fix.	4 days ago
data	Added powershell scripts to mimic the .sh script to download and crea…	5 months ago
docker	Merge branch 'master' of github.com:BVLC/caffe	3 months ago
docs	Merge branch 'master' of github.com:BVLC/caffe	2 months ago
examples	Enable model fuse script to generate merged-model and adding an examp…	5 days ago
include	Move half.hpp's license to 3rdparty/half.	4 days ago
matlab	Merge branch 'windows' of github.com:BVLC/caffe	2 months ago
models	Add one infernece optimized model file for AlexNet.	5 days ago
python	Netgen min shape update.	a month ago
scripts	Merge branch 'master' of github.com:BVLC/caffe	2 months ago
src	Lint fix.	4 days ago
tools	Add negative slope support for relu fusion.	5 days ago
.Doxyfile	update doxygen config to stop warnings	3 years ago
.gitattributes	Add support for windows build	a year ago
.gitignore	Deconv layer improvements.	3 months ago
.travis.yml	Stop setting cache timeout in TravisCI	a year ago
CMakeLists.txt	Add option to disable host unified memory in CMake.	2 months ago
CONTRIBUTING.md	[docs] add CONTRIBUTING.md which will appear on GitHub new Issue/PR p…	2 years ago
CONTRIBUTORS.md	BVLC -> BAIR	3 months ago
INSTALL.md	installation questions -> caffe-users	2 years ago
LICENSE	Merge	5 months ago
Makefile	Merge branch 'master' of github.com:BVLC/caffe	3 months ago
Makefile.config.example	Add possibility to disable host unified memory in Makefile build.	4 months ago
README.md	Updated Readme	5 days ago
appveyor.yml	Merge	5 months ago
caffe.cloc	[fix] stop cloc complaint about cu type	3 years ago
protoc_generator.sh	Updated MergeCrop layer parameters.	2 years ago

README.md

OpenCL Caffe

This is an experimental, community-maintained branch led by Fabian Tschopp (@naibaf7). It is a work-in-progress.

Custom distributions

- Intel Caffe (Optimized for CPU and support for multi-node), in particular Xeon processors (HSW, BDW, Xeon Phi).
- OpenCL Caffe e.g. for AMD or Intel devices.
- Windows Caffe

Community

gitter join chat

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR)/The Berkeley Vision and Learning Center (BVLC) and community contributors.

Caffe is released under the BSD 2-Clause license. The BAIR/BVLC reference models are released for unrestricted use.

For error reports, please run and include the result of `./build/test/test_all.testbin --gtest_filter=*OpenCLKernelCompileTest* X` where `X` is the OpenCL device to test (i.e. `0`). This test is available after a build with `make all`, `make runtest`.

This branch of Caffe contains an OpenCL backend and additional layers for fast image segmentation. This work is partially supported by:

- AMD
- HHMI Janelia
- UZH, INI
- ETH Zurich
- Intel
- DIY Deep Learning for Vision with Caffe
- Tutorial Documentation
- BAIR reference models and the community model zoo
- Installation instructions

OpenCL Backend

The backend is supposed to work with all vendors. Note however there may be problems with libOpenCL.so provided by nVidia. It is therefore recommended to install another OpenCL implementation after installing nVidia drivers. Possibilities are:

- Intel OpenCL, see https://github.com/01org/caffe/wiki/clCaffe for details.
- AMD APP SDK (OpenCL), recommended if you have an AMD GPU or CPU.

Technical Report

Available on arXiv: http://arxiv.org/abs/1509.03371

Windows Caffe

This is an experimental, communtity based branch led by Guillaume Dumont (@willyd). It is a work-in-progress.

This branch of Caffe ports the framework to Windows.

build passing Travis (Linux build)

build passing AppVeyor (Windows build)

Prebuilt binaries

Prebuilt binaries can be downloaded from the latest CI build on appveyor for the following configurations:

- Visual Studio 2015, CPU only, Python 3.5: Caffe Release, ~~Caffe Debug~~
- Visual Studio 2015, CUDA 8.0, Python 3.5: Caffe Release
- Visual Studio 2015, CPU only, Python 2.7: Caffe Release, Caffe Debug
- Visual Studio 2015,CUDA 8.0, Python 2.7: Caffe Release
- Visual Studio 2013, CPU only, Python 2.7: Caffe Release, Caffe Debug

Windows Setup

Requirements

- Visual Studio 2013 or 2015
- CMake 3.4 or higher (Visual Studio and Ninja generators are supported)

Optional Dependencies

- Python for the pycaffe interface. Anaconda Python 2.7 or 3.5 x64 (or Miniconda)
- Matlab for the matcaffe interface.
- CUDA 7.5 or 8.0 (use CUDA 8 if using Visual Studio 2015)
- cuDNN v5

We assume that `cmake.exe` and `python.exe` are on your `PATH`.

Configuring and Building Caffe

The fastest method to get started with caffe on Windows is by executing the following commands in a `cmd` prompt (we use `C:\Projects` as a root folder for the remainder of the instructions):

```
C:\Projects> git clone https://github.com/BVLC/caffe.git
C:\Projects> cd caffe
C:\Projects\caffe> git checkout windows
:: Edit any of the options inside build_win.cmd to suit your needs
C:\Projects\caffe> scripts\build_win.cmd
```

The `build_win.cmd` script will download the dependencies, create the Visual Studio project files (or the ninja build files) and build the Release configuration. By default all the required DLLs will be copied (or hard linked when possible) next to the consuming binaries. If you wish to disable this option, you can by changing the command line option `-DCOPY_PREREQUISITES=0`. The prebuilt libraries also provide a `prependpath.bat` batch script that can temporarily modify your `PATH` envrionment variable to make the required DLLs available.

Below is a more complete description of some of the steps involved in building caffe.

Install the caffe dependencies

By default CMake will download and extract prebuilt dependencies for your compiler and python version. It will create a folder called `libraries` containing all the required dependencies inside your build folder. Alternatively you can build them yourself by following the instructions in the caffe-builder README.

Use cuDNN

To use cuDNN the easiest way is to copy the content of the `cuda` folder into your CUDA toolkit installation directory. For example if you installed CUDA 8.0 and downloaded cudnn-8.0-windows10-x64-v5.1.zip you should copy the content of the `cuda` directory to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0`. Alternatively, you can define the CUDNN_ROOT cache variable to point to where you unpacked the cuDNN files e.g. `C:/Projects/caffe/cudnn-8.0-windows10-x64-v5.1/cuda`. For example the command in scripts/build_win.cmd would become:

```
cmake -G"!CMAKE_GENERATOR!" ^
      -DBLAS=Open ^
      -DCMAKE_BUILD_TYPE:STRING=%CMAKE_CONFIG% ^
      -DBUILD_SHARED_LIBS:BOOL=%CMAKE_BUILD_SHARED_LIBS% ^
      -DBUILD_python:BOOL=%BUILD_PYTHON% ^
      -DBUILD_python_layer:BOOL=%BUILD_PYTHON_LAYER% ^
      -DBUILD_matlab:BOOL=%BUILD_MATLAB% ^
      -DCPU_ONLY:BOOL=%CPU_ONLY% ^
      -DCUDNN_ROOT=C:/Projects/caffe/cudnn-8.0-windows10-x64-v5.1/cuda ^
      -C "%cd%\libraries\caffe-builder-config.cmake" ^
      "%~dp0\.."
```

Alternatively, you can open `cmake-gui.exe` and set the variable from there and click `Generate`.

Building only for CPU

If CUDA is not installed Caffe will default to a CPU_ONLY build. If you have CUDA installed but want a CPU only build you may use the CMake option `-DCPU_ONLY=1`.

Using the Python interface

The recommended Python distribution is Anaconda or Miniconda. To successfully build the python interface you need to add the following conda channels:

```
conda config --add channels conda-forge
conda config --add channels willyd
```

and install the following packages:

```
conda install --yes cmake ninja numpy scipy protobuf==3.1.0 six scikit-image pyyaml pydotplus graphviz
```

If Python is installed the default is to build the python interface and python layers. If you wish to disable the python layers or the python build use the CMake options `-DBUILD_python_layer=0` and `-DBUILD_python=0` respectively. In order to use the python interface you need to either add the `C:\Projects\caffe\python` folder to your python path of copy the `C:\Projects\caffe\python\caffe` folder to your `site_packages` folder.

Using the MATLAB interface

Follow the above procedure and use `-DBUILD_matlab=ON`. Change your current directory in MATLAB to `C:\Projects\caffe\matlab` and run the following command to run the tests:

```
>> caffe.run_tests()
```

If all tests pass you can test if the classification_demo works as well. First, from `C:\Projects\caffe` run `python scripts\download_model_binary.py models\bvlc_reference_caffenet` to download the pre-trained caffemodel from the model zoo. Then change your MATLAB directory to `C:\Projects\caffe\matlab\demo` and run `classification_demo`.

Using the Ninja generator

You can choose to use the Ninja generator instead of Visual Studio for faster builds. To do so, change the option `set WITH_NINJA=1` in the `build_win.cmd` script. To install Ninja you can download the executable from github or install it via conda:

```
> conda config --add channels conda-forge
> conda install ninja --yes
```

When working with ninja you don't have the Visual Studio solutions as ninja is more akin to make. An alternative is to use Visual Studio Code with the CMake extensions and C++ extensions.

Building a shared library

CMake can be used to build a shared library instead of the default static library. To do so follow the above procedure and use `-DBUILD_SHARED_LIBS=ON`. Please note however, that some tests (more specifically the solver related tests) will fail since

both the test exectuable and caffe library do not share static objects contained in the protobuf library.

Troubleshooting

Should you encounter any error please post the output of the above commands by redirecting the output to a file and open a topic on the caffe-users list mailing list.

Known issues

- The `GPUTimer` related test cases always fail on Windows. This seems to be a difference between UNIX and Windows.
- Shared library (DLL) build will have failing tests.
- Shared library build only works with the Ninja generator

Further Details

Refer to the BVLC/caffe master branch README for all other details such as license, citation, and so on.