This repository Search Pull requests Issues Marketplace Gist

hughperkins / **DeepCL**

Watch 59 Star 565 Fork 156

Code Issues 20 Pull requests 0 Projects 0 Insights

OpenCL library to train deep convolutional neural networks

2,097 commits **28** branches **50** releases **11** contributors MPL-2.0

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

hughperkins committed on **GitHub** Merge pull request #120 from merceyz/master … Latest commit `dd0bbdd` 13 days ago

EasyCL @ 48feac4	update to latest easycl	a year ago
cl	remove lots of opencl build warnings on Mac	2 years ago
clMathLibraries	pass args to initkernelarg by reference, to avoid issue with alignmen…	11 months ago
cmake	run unit tests from travis	2 years ago
cog-batteries	add license header to stringify.py	2 years ago
data	remove cifar10 from git	2 years ago
doc	tweak doxy doc slightly	9 months ago
jenkins	switch to libjpeg on windows	11 months ago
prototyping	formatting of spaces around ()	a year ago
python	tweak build for windows for upload pypir; also switch to c++11	11 months ago
src	SoftMaxLayer: Added missing include	13 days ago
test	add test for backprop exception in dropout	7 months ago
thirdparty	try adding wildcards to cog, so works on windows	2 years ago
travis	add __str__ and __unicode__ to neuralnet.pyx	a year ago
.gitignore	doxygen notes/readme	9 months ago
.gitmodules	added clblas to makelists	2 years ago
.travis.yml	change xcode version in travis.yml, and put back the std= option on m…	a year ago
CMakeLists.txt	add test for backprop exception in dropout	7 months ago
DeepCLConfig.cmake.in	Added DeepCL project config file generation and fixed manifest txt lo…	2 years ago
LICENSE	Initial commit	3 years ago
README.md	update readme that unit tests build/run on mac	2 months ago
files.txt	reorg a bit, like packges	2 years ago

README.md

DeepCL

DeepCL

OpenCL library to train deep convolutional networks

- C++
- OpenCL
- Deep convolutional
- Python wrappers
- Lua wrappers
- Q-learning

APIs:

- Python
- c++
- command-line

Layer types:

- convolutional
- max-pooling
- normalization
- activation
- dropout
- random translations
- random patches
- loss

Loss layer types:

- softmax
- cross-entropy (synonymous with multinomial logistic, etc)
- square loss

Trainers:

- SGD
- Anneal
- Nesterov
- Adagrad
- Rmsprop
- Adadelta

Activations:

- tanh
- scaled tanh (1.7519 * tanh(2/3x))
- linear
- sigmoid
- relu
- elu (new!)

Loader formats:

- jpegs
- mnist
- kgsv2
- norb

Weight initializers:

- original

- uniform
- more possible...

Multicolumn net also possible, as in McDnn

Example usages

- obtained 37.2% test accuracy, on next move prediction task, using 33.6 million training examples from kgsgo v2 dataset
  - commandline used `./deepclrun dataset=kgsgoall netdef=12*(32c5z-relu)-500n-tanh-361n numepochs=15 learningrate=0.0001`
  - 2 epochs, 2 days per epoch, on an Amazon GPU instance, comprising half an NVidia GRID K520 GPU (about half as powerful as a GTX780)
- obtained 99.5% test accuracy on MNIST, using `netdef=rt2-8c5z-relu-mp2-16c5z-relu-mp3-150n-tanh-10n numepochs=20 multinet=6 learningrate=0.002`
  - epoch time 99.8 seconds, using an Amazon GPU instance, ie half an NVidia GRID K520 GPU (since we are learning 6 nets in parallel, so 16.6seconds per epoch per net)

Installation

Native library installation

This section installs the native libraries, and the command-line tools. You always need to do this part, even if you will use the Python wrappers.

Windows

Pre-requisites:

- OpenCL-enabled GPU or APU, along with appropriate OpenCL driver installed
- Tested using Windows 2012 RC2, and (New!) Visual Studio 2015, this is how the CI builds run

Procedure:

- Download latest binary zip file from http://deepcl.hughperkins.com/Downloads/ (eg from v8.0.0rc8)
- unzip it, which creates the `dist` folder
- To test it:
  - open a cmd
  - run `call dist\bin\activate.bat` (adjusting the path appropriately for wherever you downloaded deepcl binaries to)
  - now, eg try `deepcl_unittests`
  - (New!), you can choose which gpu to run tests on now, eg: `deepcl_unittests gpuindex=1`

Note that you need to "activate" the installation each time you open a new cmd prompt (or you could add appropriate environment variables permanently, using Control Panel | System | Advanced System Settings | Environment Variables)

Linux

Pre-requisites:

- OpenCL-enabled GPU or APU, along with appropriate OpenCL driver installed (can check by running `clinfo` , which should show your desired GPU device)
- Tested using Ubuntu 14.04 32-bit/64-bit

Procedure:

- Download latest tar file from http://deepcl.hughperkins.com/Downloads/ (eg from v8.0.0rc8)
- untar it, which creates the `dist` sub-folder
- in a bash prompt, run `source dist/bin/activate.sh` (adjust the path appropriate for wherever you untarred the binaries tar file to)
- test by doing, from the same bash prompt, eg `deepcl_unittests`
  - (New!), you can choose which gpu to run tests on now, eg: `deepcl_unittests gpuindex=1`

Note that you need to "activate" the installation each time you open a new bash prompt (or you can call activate.sh from your `.bashrc` file)

Python wrappers

- make sure you already installed the native library, and "activate"d it, by doing `call dist\bin\activate.bat` , or `source dist/bin/activate.sh`
- run `pip install --pre DeepCL`
- test by doing `python -c "import PyDeepCL; cl = PyDeepCL.DeepCL()"`

To build from source

Building from source is only needed if installing from binaries doesn't work for your configuration, or if you want to modify DeepCL.

See Build.md

What if it doesn't run?

- Check if you have an OpenCL-enabled device on your system
  - ideally a GPU, or accelerator, since there is no attempt to optimize DeepCL for CPUs (at least, not currently, could change, feel free to submit a pull request :-))
- Try running `gpuinfo` (from EasyCL, but built as part of this project too, for ease of use)
  - it should output at least one OpenCL-enabled device
  - if it doesn't, then you need to make sure you have an OpenCL-enabled device, and that appropriate drivers are installed, and that the ICD is configured appropriately (registry in Windows, and `/etc/OpenCL/vendors` in linux)

What if I need a new feature?

Please raise an issue, let me know you're interested.

- If it's on my list of things I was going to do sooner or later anyway (see below), I might do it sooner rather than later.
- If it's to do with usability, I will try to make that a priority

What if I want to contribute myself?

- please feel free to fork this repository, tweak things, send a pull request. Or get in contact. Or both :-)

Third-party libraries

- EasyCL
- clew
- libpng++
- lua
- cogapp

Hardware/driver specific issues

- If you're using Clover, you might want to look at:
  - this thread https://github.com/hughperkins/DeepCL/issues/35
  - this branch https://github.com/hughperkins/DeepCL/tree/clover-compatibility

Related projects

- kgsgo-dataset-preprocessor Dataset based on kgsgo games; 33 million data points
- cltorch

- clnn

License

Mozilla Public License 2.0

Recent changes

- 2017 May 2nd:
  - branch `update-easycl-mac` updated to latest EasyCL, and unit-tests tested on Mac Sierra against:
    - Intel HD Graphics 530 GPU
    - Radeon Pro 450 GPU
  - This latest EasyCL lets you use environment variable `CL_GPUOFFSET` to select gpus, eg set to `1` for second GPU, or `2` for third
  - Thank you to my employer ASAPP for providing me use of said Mac Sierra :-)
- 7th August 2016:
  - "standard" version of windows compiler changed from msvc2010 to msvc2015 update 3 (no change to linux/mac)
  - "standard" version of python 3.x on windows changed from 3.4 to 3.5 (no change to linux/mac)
  - (note: python2.7 continues to work as before on all of Windows 32/64, linux, Mac)
  - standard c++ version on linux/mac changed from c++0x to c++11
- 29th July 2016:
  - python fixes:
    - CHANGE: must use numpy tensors now, `array.array` no longer accepted
    - New feature: can provide numpy tensors as 4d tensors now, no longer have to be 1d tensors
    - Bug fix: q-learning working again now (hopefully)
- 26th July 2016:
  - fixed some bugs in manifest loader
  - no longer need to specify the number of images in the first line of the manifest file
  - added `gpuindex=` option to `deepcl_unittests` (quite beta for now...)
- 4th January 2016:
  - fixed a number of build warnings on Mac, both in OpenCL build, and C++ build
- 3rd January 2016:
  - create Mac OS X build on Travis, and fix the build, https://travis-ci.org/hughperkins/DeepCL
- 27th November:
  - added ELU
- Week of 26th October:
  - created branch `clblas-2.8.0` , which works with Visual Studio 2015. It uses the latest 2.8.x release of clBLAS. Thank you to jakakonda for helping to test this and get it working.
- Aug 28th:
  - merged 8.x branch to master, will release first version of 8.x shortly
  - installation of 8.x from binaries on Windows works now, by doing, eg on 32-bit Windows 7, and assuming you already activated an appropriate python environment (assumes 7-zip is installed, in default location, otherwise do the unzip by hand):

```
powershell Set-ExecutionPolicy unrestricted
rem following command is like `wget` in linux:
powershell.exe -Command (new-object System.Net.WebClient).DownloadFile('http://deepcl.hughperkins.com/Downl
rem following command is like `tar -xf` in linux:
"c:\program files\7-Zip\7z.exe" x deepcl-win32-v8.0.0rc8.zip
call dist\bin\activate.bat
pip install --pre DeepCL
python -c "import PyDeepCL; cl = PyDeepCL.DeepCL()"
# (last line is just to check works ok)
```

- Aug 26th: installation of 8.x from binaries on linux works now, by doing, eg on 64-bit Ubuntu 14.04:

```
mkdir 8.0.0rc4
cd 8.0.0rc4
```

```
wget http://deepcl.hughperkins.com/Downloads/deepcl-linux64-v8.0.0rc4.tar.bz2
tar -xf deepcl-linux64-v8.0.0rc4.tar.bz2
virtualenv env
source env/bin/activate
source dist/bin/activate.sh
pip install --pre DeepCL
python -c "import PyDeepCL; cl = PyDeepCL.DeepCL()"
```

(last line is just to check works ok)

- Aug 21st-24th:
  - 8.x finally builds again on all CI tested configurations!
    - ubuntu 14.04 32-bit Python 2.7
    - ubuntu 14.04 32-bit Python 3.4
    - ubuntu 14.04 64-bit Python 2.7
    - ubuntu 14.04 64-bit Python 3.4
    - visual studio 2010 32-bit python 2.7
    - visual studio 2010 32-bit python 3.4
    - visual studio 2010 64-bit python 2.7
    - visual studio 2010 64-bit python 3.4
- Aug 19th-20th:
  - Python wrappers now built using a very thin setup.py layer, on top of the standard native DeepCL build
- Aug 18th:
  - added BackwardIm2Col layer, which uses im2col for backward propagation
  - added BackpropWeightsIm2Col layer, which uses im2col for weight update
  - added BackwardAuto layer, which automatically selects fastest Backward layer
  - added BackpropWeightsAuto layer, which automatically selects faster weight update layer
  - under the covers:
    - created ClBlasHelper, to handle Gemm and Gemv
    - factorized im2col into Im2Col class
- week up to Aug 17th:
  - added forward and backward im2col layer
  - forward im2col automatically used during forward propagation, where appropriate
  - backwards has yet to be integrated
  - under the covers:
    - added clBLAS
    - migrated the Python build process to use cmake, rather than setup.py (whether this turns out to be good or bad is a bit up in the air for now)
- June 22nd:
  - removed lua wrappers
  - if you want to use lua with OpenCL, please consider using cltorch and clnn

To get in contact

Just create an issues, in github, in the top right of this page. Don't worry about whether you think the issue sounds silly or anything. The more feedback the better!

Note that I'm currently focused 100.000% on cuda-on-cl, so please be patient during this period.