Dlib C++ Library

Sunday, June 26, 2016

A Clean C++11 Deep Learning API

Dlib 19.0 is out and it has a lot of new features, like new elastic net and quadratic program solvers. But the feature I'm most excited about is the new deep learning API. There are a lot of existing deep learning frameworks, but none of them have clean C++ APIs. You have to use them through a language like Python or Lua, which is fine in and of itself. But if you are a professional software engineer working on embedded computer vision projects you are probably working in C++, and using those tools in these kinds of applications can be frustrating.

So if you use C++ to do computer vision work then dlib's deep learning framework is for you. It makes heavy use of C++11 features, allowing it to expose a very clean and lightweight API. For example, the venerable LeNet can be defined in pure C++ with a using statement:

LeNet

```
using LeNet = loss_multiclass_log<
                 fc<10,
                 relu<fc<84,
                 relu<fc<120,
                 max_pool<2,2,2,2,relu<con<16,5,5,1,1,
                 max_pool<2,2,2,2,relu<con<6,5,5,1,1,
                 input<matrix<unsigned char>>>>>>>>>>>>>>;
```

Then, using it to train and test a neural network looks like this:

```
LeNet net;
dnn_trainer<LeNet> trainer(net);
trainer.set_learning_rate(0.01);
trainer.set_min_learning_rate(0.00001);
trainer.set_mini_batch_size(128);
trainer.train(training_images, training_labels);
// Ask the net to predict labels for all the testing images
auto predicted_labels = net(testing_images);
```

Dlib will even automatically switch to lower learning rates when the training error stops improving, so you won't have to fiddle with learning rate schedules. The API will certainly let you do so if you want that control. But I've been able to train a number of state-of-the-art ImageNet models without any manual fiddling of learning rates, which I find to be very convenient.

Depending on how you compile dlib, it will use either the CPU or cuDNN v5. It also supports using multiple GPUs during training and has a "fast mode" and a "low VRAM" mode. Compared to Caffe, dlib's fast mode is about 1.6x times faster than Caffe but uses about 1.5x as much VRAM, while the low VRAM mode is about 0.85x the speed of Caffe but uses half the VRAM as Caffe. So dlib's new deep learning API is fast but can also let you run larger models in the same amount of VRAM if you are VRAM constrained.

It's also fully documented. The basics are covered in this tutorial and then more advanced concepts are covered in a follow on tutorial. These tutorials show how to define LeNet and ResNet architectures in dlib and another tutorial shows how to define Inception networks. And even more importantly, every function and class in the API is documented in the reference material. Moreover, if you want to define your own computational layers, loss layers, input layers, or solvers, you can because the interfaces you have to implement are fully documented.

I've also included a pretrained ResNet34A model and this example shows how to use it to classify images. This pretrained model has a top5 error of 7.572% on the 2012 imagenet validation dataset, which is slightly better than the results reported in the original paper Deep Residual Learning for Image Recognition by He, Zhang, Ren, and Sun. Training this model took about two weeks while running on a single Titan X GPU.

To use the new deep learning tools, all you need to install is cuDNN v5. Then you can compile the dlib example programs using the normal CMake commands. There are no other dependencies. In fact, if you don't install cuDNN CMake will automatically configure dlib to use only the CPU and the examples will still run (but much slower). You will however need a C++11 compiler, which precludes current versions of visual studio since they shamefully *still* lack full C++11 support. But any mildly recent version of GCC will work. Also, you can use visual studio with the non-DNN parts of dlib as they don't require C++11 support.

Finally, development of this new deep learning toolkit was sponsored by Systems & Technology Research, as part of the IARPA JANUS project. Without their support and feedback it wouldn't be nearly as polished and flexible. Jeffrey Byrne in particular was instrumental in finding bugs and usability problems in early versions of the API.

Library news

- dlib C++ Library: 19.4 Released
- dlib C++ Library: 19.3 Released
- dlib C++ Library: 19.2 Released
- dlib C++ Library: 19.1 Released
- dlib C++ Library: 19.0 Released

Blog Archive

Posted by Davis King at 8:29 AM

12 comments :

LUI said...

Excellent news. Thanks for sharing this

June 27, 2016 at 6:04 AM

Vladimir Yumatov said...

Are you sure that VS2015 doesn't support C++11? At least, your DNN examples work just fine for me. But I had to remove C++11 support check from use_cpp_11.cmake file because it will always fail with MSVC anyway.

July 22, 2016 at 7:24 AM

Davis King said...

Some of them work. Did you try to compile all the example programs?

July 22, 2016 at 8:12 AM

Vladimir Yumatov said...

Apparently, ImageNet examples can't be built. Building process goes on forever. Well, that's a shame. Anyway, thanks for the reply.

July 25, 2016 at 5:50 AM

mohanraj said...

Am trying to compile Dlib 19.0 using cmake. The following errors are occured. Am using Visual Studio 2012 to compile the dlib.

error C1083: Cannot open include file: 'initializer_list': No such file or directory hel

kindly help me to solve the errors.

July 26, 2016 at 6:29 AM

Davis King said...

What compiler are you using?

July 26, 2016 at 6:38 AM

Unknown said...

I think problem is Cudnn library is not compatible with higher than msvs 2013.

July 28, 2016 at 1:32 PM

Davis King said...

Yeah that's a problem. But visual studio also doesn't support C++11 so the CPU mode of dlib doesn't work either, regardless of any cuDNN considerations.

July 28, 2016 at 7:06 PM

Andreo said...

Davis, thank you for your great library! Did you ever thought about training your face landmark network on MUCT database?
https://github.com/StephenMilborrow/muct
It has 76 facial points on each photo and your network could be more precise with such training set.

December 13, 2016 at 5:00 AM

Davis King said...

I'm sure you could train it on that dataset and get a working model. I'm not going to do it though as I have other more pressing things to do :)

December 13, 2016 at 6:50 AM

Andreo said...

Thank you for reply)

December 15, 2016 at 12:04 AM

oğuz çetinol said...

Hello Mr. King

I want to ask few questions about dlib shape predictor training. Is this the right platform ?
Thank you in advance.
Oguz Cetinol

February 1, 2017 at 8:56 AM

Post a Comment

Newer Post   Home   Older Post

Subscribe to: Post Comments (Atom)

Powered by Blogger.