MyCaffe: A Complete C# Re-Write of Caffe, for Windows Programmers

David W. Brown
daveb@signalpop.com
SignalPop LLC.

10/4/2017

Abstract

Over the past few years Caffe [1]*, from Berkeley AI Research, has gained a strong following in the deep learning community with over 10k forks on the github.com/BLVC/Caffe site. With its well organized, very modular C++ design it is easy to work with and very fast. However in the world of Windows development C# has helped accelerate development with many of the enhancements that it offers over C++, such as garbage collection, a very rich .NET programming framework and easy database access via Entity Frameworks* [2]*. So how can a C# developer use the advances of C# to take full advantage of the benefits offered by the Berkeley Caffe deep learning system? The answer is the newly released, fully open source, 'MyCaffe' for Windows .NET programmers. MyCaffe is an open source, complete re-write of Berkeley's Caffe, in the C# language.*

This article describes the general architecture of MyCaffe, how it closely follows the C++ Caffe down to each and every comment and automated test, yet does so while talking efficiently to the low level NVIDIA CUDA hardware to offer a high performance, highly programmable deep learning system for Windows .NET programmers.

Introduction

The goal of MyCaffe is not to replace the C++ Caffe, but instead to augment the overall platform by expanding its footprint to Windows C# programmers in their native programming language so that this large group of software developers can more easily develop Caffe models that then both the C++ Caffe and C# MyCaffe communities can benefit from. In addition, MyCaffe allows Windows developers to easily expand MyCaffe by writing their own new layers using the C# language.

The following table compares the similarities and differences between the C++ Caffe and C# MyCaffe software.

	C++ Caffe	**C# MyCaffe**
All vision layers	x	x
All recurrent layers	x	x
All neuron layers	x	x
All cuDnn layers	x	x
All common layers	x	x
All loss layers	x	x
All data layers	x	All but 2
All auto tests	x	x
All solvers	x	x
All fillers	x	x
Net object	x	x
Blob object	x	x
SyncMem object	x	x
DataTransform object	x	x
Parallel object(s)	x	x
NCCL (Nickel)	x	x
Caffe object	x	CaffeControl
Database	Levledb, Imdb	MSSQL (+Express)
Low-level Cuda	C++	C++
ProtoTxt	Text	Compatible objects
Weights	Binary	Compatible
GPU Support	1-8 gpus	1-8 gpus*
CPU Support	x	GPU only

Table 1 Support Comparison

* Multi-GPU support only on headless GPUs.

From Table 1 above you can see that MyCaffe is a very close match to the C++ Caffe, with the main differences being in the database support and the way that MyCaffe accesses the low-level C++ CUDA code. In addition, MyCaffe is designed to only run on GPU bases systems and supports a wide range of NVIDIA cards from the NVIDIA GeForce 750ti on up to multi-GPU Tesla based systems running in TCC mode.

Like the C++ Caffe, MyCaffe can be complied to run using either a 'float' or 'double' as its base type.

General Architecture

Four main modules make up the MyCaffe software: MyCaffe Control, MyCaffe Image Database, Cuda Control and the low-level Cuda C++ Code DLL.

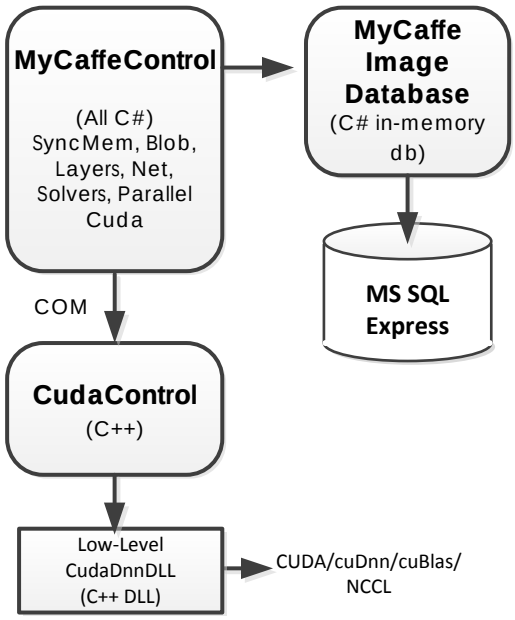

Table 2 MyCaffe Architecture

The MyCaffe Control is the main interface to software that uses MyCaffe. It mainly plays the role of the 'caffe' object in C++ Caffe but does so in a way more convenient for C# Windows programmers for the control is easily dropped into a windows program and used.

The MyCaffe Image Database is an in-memory database used to load datasets or portions of datasets from a Microsoft SQL or Microsoft SQL Express database into memory. In addition to providing the typical database functionality, the MyCaffe Image Database adds several useful features to deep learning, namely: label balancing [3] and image boosting [4]. When using label balancing the up-sampling occurs at the database for the database itself organizes all data by label when loaded thus allowing a random selection of the label group first and then a random selection of the image from the label group so as to ensure that labels are equally represented when training and not skew training toward one label or another. Image boosting allows the user to mark specific images with a higher boost value to increase the probability that the marked image is selected during training. These are optional features that we have found helpful when training on sparse datasets that tend to have imbalanced label sets.

The Cuda Control is a C++ COM Control that supports COM/OLE Automation communication. Using the COM support built into C#, the MyCaffe Control communicates to the Cuda Control seamlessly by passing parameters and a function index which is then used internally to call the appropriate CUDA based function within the Low-Level CudaDnn DLL. All parameters are converted from the COM/OLE Automation format and into native C types within the Cuda Control thus allowing the Low-Level CUDA C++ Code to focus on the low-level programming such as calling CUDA kernels and/or calling cuDnn and NCCL functions.

The Low-Level CudaDnn DLL (different from, but uses the cuDnn [5] library from NVIDIA) contains all of the low-level functionality such as the low level MyCaffe math functions, calls to cuDnn, calls to NCCL and several low level t-SNE [6] and PCA [7] functions provided for speed. In order to manage the CUDA memory, CUDA Streams and special object pointers such as the Tensor Descriptors

used by cuDnn, the CudaDnn DLL uses a set of look-up tables and keeps track of each and every memory allocation and special pointer allocated. A look-up index into each table acts as a handle that is then passed back on up to and used by the C# code within the MyCaffe Control. Thanks to the way GPU's operate with a (mainly) separate memory space from the CPU, a look-up table works well for once the memory loaded on the GPU, you generally want to keep it there for as long as possible so as to avoid the timing hit caused by transferring between the GPU and host memory.

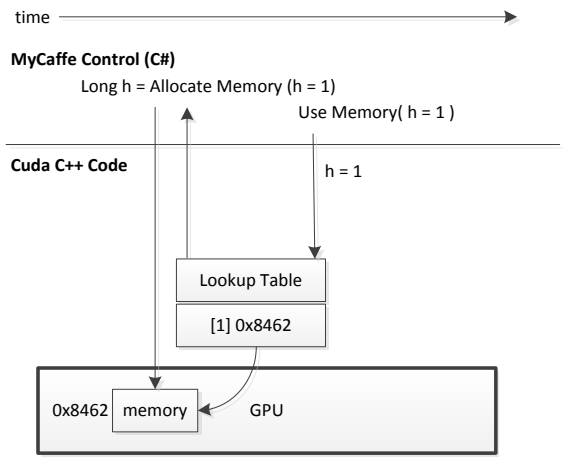

Table 3 Fast Lookup Table

MyCaffe also employs this same concept to manage all special data pointers used by the Cuda and cuDnn libraries where each pointer to an allocated descriptor or stream is also placed in its own look-up table that is referenced using an index (or handle) up in the MyCaffe Control software. To support some sub-systems such as NCCL, t-SNE and PCA, internal objects are allocated to manage the sub-system and stored in a look-up table thus allowing the CudaDnn DLL to manage the state of these subsystems while they are being used.

Benchmarks

In deep learning, speed is important. For that reason, we have run the MyCaffe on several Windows configurations to show how well the technology stacks up[1].

The following benchmarks were run on an i7-5960 8-core 3.00GHZ Dell/Alienware Area-51 R2 Computer running Windows 7 with 32GB of memory loaded with an NVIDIA Titan X (Maxwel) GPU in WDM[2] mode and CUDA 9.

Model	GPU Memory Used	Image Size	Batch Size	Fwd/Bwd Average Time (ms.)
GoogleNet	n/a	224x224	*128[3] est*	*800 ms*
GoogleNet	9.01 GB	224x224	64	400 ms
GoogleNet	5.32 GB	224x224	32	260 ms
GoogleNet	3.48 GB	224x224	16	230 ms
VGG-16	8.57 GB	224x224	32	580 ms
VGG-16	5.51 GB	224x224	16	312 ms
GoogleNet	2.36 GB	56x56	24	166 ms
AlexNet	1.32 GB	32x32	128	35 ms

The following benchmarks were run on the same Dell/Alienware Area-51 R2 Computer (i7-5960 8-core 3.00GHZ) running Windows 7 with 32GB of memory but this time loaded with a NVIDIA Titan Xp (Pascal) GPU in TCC mode and CUDA 9.

Model	GPU Memory Used	Image Size	Batch Size	Fwd/Bwd Average Time (ms.)
GoogleNet	n/a	224x224	*128[4] est*	*570 ms*
GoogleNet	8.91 GB	224x224	64	285 ms
GoogleNet	4.73 GB	224x224	32	140 ms
GoogleNet	3.24 GB	224x224	16	65 ms
VGG-16	7.56 GB	224x224	32	280 ms
VGG-16	5.28 GB	224x224	16	138 ms
GoogleNet	2.09 GB	56x56	24	38 ms
AlexNet	1.05 GB	32x32	128	37 ms

Comparison benchmarks (run in 2014) show that C++ Caffe using an older version of cuDnn running GoogleNet on an NVIDIA K40c with a

[1] The SignalPop AI Designer version 0.9.0 using the LOAD_FROM_SERVICE mode was used for all tests with the dataset fully loaded in memory.
[2] Run on GPU with monitor plugged in.
[3] Estimated by multiplying batch 64 timing x2.
[4] Estimated by multiplying batch 64 timing x2.

128 batch size had a running total time of 1688.8 ms. for the forward + backward pass [8].

This truly shows the great strides that NVIDIA has made both in hardware improvements with their Titan Xp (Pascal) GPU and software improvements to their CUDA and cuDnn libraries with CUDA 9.0 and cuDnn 7.0. And the new Volta hardware should only be better - what a time to be involved in AI!

But how does MyCaffe stack up on every day GPU's such as the 1060 that is in many new laptops for the highly used 750TI? The following benchmarks show what these more standard PC systems can do as well.

The following benchmarks were run on an i7 6700HQ 6-core 2.60GHZ Alienware 15 Laptop running Windows 10 with 8 GB of memory loaded with an NVIDIA GTX 1060 GPU in WDM[5] mode with CUDA 9.

Model	GPU Memory Used	Image Size	Batch Size	Fwd/Bwd Average Time (ms.)
GoogleNet	3.01 GB	56x56	24	88 ms
AlexNet	1.98 GB	32x32	128	48 ms

The following benchmarks were run on an i7 x980 6-core 3.33GHZ HP Pavilion Elite running Windows 7 with 24 GB of memory loaded with an NVIDIA 750ti GPU in WDM[6] mode with CUDA 9.

Model	GPU Memory Used	Image Size	Batch Size	Fwd/Bwd Average Time (ms.)
GoogleNet	2.03 GB	56x56	24	268 ms
AlexNet	0.94 GB	32x32	128	134 ms

Compatibility

Since our goal is to expand the target market for the C++ Caffe platform to the general C# Windows programmer, we have strived to make sure that MyCaffe maintains compatibility with the C++ Caffe platform prototxt scripts and binary weight file formats. To do this MyCaffe performs its own parsing of the prototxt files into internal C# parameter objects that closely match those generated by Googles ProtoTxt software. In addition each parameter object is designed to print itself out in the very same ProtoTxt format from which it was parsed. Using this methodology allows MyCaffe to maintain compatibility with the same prototxt files used by the C++ Caffe.

With regard to the weigh files, MyCaffe stores weight files using the same Google ProtoBuf binary format as the C++ Caffe allowing MyCaffe to use existing *.caffemodel* files created using the C++ Caffe.

Summary

It is great to see fantastic software such as Caffe out in the open source community. As a thank you, we wanted to contribute back to that same community with the 'MyCaffe' project to help an even larger group of programmers use this great technology.

For more information on the C++ Caffe, see the Berkeley AI Research site at http://caffe.berkeleyvision.org/.

For more information on the Windows version of MyCaffe written in C#, see us on GitHub at https://github.com/mycaffe. For easy integration into your existing C# applications, just search for MyCaffe on NuGet or go to https://www.nuget.org/packages?q=MyCaffe. And for more information on innovative products that make designing, editing, training, testing and debugging your AI models easier, see us at https://www.signalpop.com.

[5] Run on GPU with monitor connected.

[6] Run on GPU with monitor connected.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolution Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093,* 2014.

[2] B. Johnson, Professional Visual Studio 2015, Indianapolis: John Wiley & Sons, Inc., 2015.

[3] F. Provost, "Machine Learning from Imbalanced Data Sets 101," 2000.

[4] R. E. Schapire and Y. Freund, Boosting: Foundations and Algorithms, Cambridge: The MIT Press, 2012.

[5] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen and J. Tran, "cuDNN: Efficient Primitives for Deep Learning," *arXiv:1410.0759v2* , pp. 1-8, 2104.

[6] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* no. 9, pp. 1-27, 2008.

[7] M. Andrecut, "Parallel GPU Implementation of Iterative PCA Algorithms," *arXiv:0811.1081 [q-bio.QM],* pp. 1-45, 2008.

[8] sguada, "Caffe Timings for GoogleNet, VGG, AlexNet with cuDNN," [Online]. Available: https://github.com/Bvlc/caffe/issues/1317.