P1360R0: Towards Machine Learning for C++: Study Group 19

Date: 2018-11-26(Post-SAN mailing): 10 AM ET

Project: ISO JTC1/SC22/WG21: Programming Language C++

Audience SG19, WG21

Authors : Michael Wong (Codeplay),
Vincent Reverdy (University of Illinois at Urbana-Champaign, Paris Observatory),
Robert Douglas (Epsilon),
Emad Barsoum (Microsoft),
Sarthak Pati (University of Pennsylvania)
Peter Goldsborough (Facebook)
Franke Seide (MS)

Contributors

Emails: michael@codeplay.com
vreverdy@illinois.edu
rwdougla@gmail.com
ebarsoum@gmail.com
sarthak.pati@uphs.upenn.edu
psag@fb.com
fseide@microsoft.com

Reply to: michael@codeplay.com

Introduction

This paper proposes a WG21 SG for Machine Learning with the goal of:

- Making Machine Learning a first-class citizen in ISO C++

It is the collaboration of a number of key industry, academic, and research groups, through several connections in CPPCON BoF[reference], LLVM 2018 discussions, and C++ San Diego meeting. The intention is to support such an SG, and describe the scope of such an SG. This is in terms of potential work resulting in papers submitted for future C++ Standards, or collaboration with other SGs. We will also propose ongoing teleconferences, meeting frequency and locations, as well as outreach to ML data scientists, conferences, and liaison with other Machine Learning groups such as at Khronos, and ISO.

As of the SAN meeting, this group has been officially created as SG19, and we will begin teleconferences immediately, after the US thanksgiving, and after NIPS. Michael Wong will be chair.

Motivation

We feel we can leverage C++ strengths in generic programming, optimization and acceleration, as well as code portability. We also can use such an SG to address and improve on its ability to support fast iteration, better support for array, matrix, linear algebra, in memory passing of data for computation, scaling, and graphing, as well as optimization for graph programming. This section will describe some of the strengths of C++ for ML. The next section on scope will outline areas that need to address through this SG.

Machine Learning, AI, and Neural Networks have entered almost every aspect of research, industry and academia. There is a widely perceived notion that this is always done in some other language than C++ (perhaps Python, or R). Even though Python is one of the most used languages for machine learning, all of the frameworks built with Python are taking advantage of a set of highly optimized C++ libraries under the hood. Ultimately if you want to deploy your application to execute on hardware that is embedded or in a more constrained environment you will ultimately need to develop the final solution using C++ (even if the deployment is using a

cloud-based virtual machine). There are a set of C++ libraries that offer fast mathematical operations and are the tools of the trade for many machine learning developers.

As a user of machine learning frameworks, C++ offers some major advantages. C++ code provides the engine for machine learning frameworks delivering powerful, high-performance operations that are used to build and execute deep neural networks in the most efficient way possible. Also, using C++ virtually ensures the user will go from prototype to production level code much quicker. Machine learning applications, in particular deep learning programs make heavy use of linear algebra operations involving geometric transformations, numerical solvers and other related algorithms. These complex algorithms operate on matrices and vectors requiring huge numbers of additions and multiplications, and C++ is used to implement these so that the execution can be accelerated by heterogeneous systems that offer multi core processors such as GPUs and AI-specific chipsets such as Intel's Nervana [Nervana], Google's Tensor Processing Unit [TPU] and Microsoft's Project Catapult [Catapult], among others. Compared to execution on traditional CPUs, the performance per watt is greatly improved. Additionally, in big clusters (cloud computing for example) it is easier to install multiple GPUs in a single node than multiple CPUs. This means that machine learning applications can take advantage of many GPUs to accelerate their execution. There are already multiple consumer products that have dedicate processing units to leverage neural network computing (for example, Google Pixel Visual Core [Pixel]). Using C++ makes it possible to write custom and flexible machine learning applications fit for any purpose and hardware architecture that use the same set of accelerated C++ libraries that machine learning frameworks like TensorFlow, CNTK and Caffe2 use. The C++ libraries include those used to perform normalization, convolution and matrix multiplication operations and it is these that give developers the building blocks to make custom machine learning applications tailored for specific needs and models.

All major machine learning frameworks have some support for accelerator processors such as GPUs and this is essential. Otherwise, machine learning applications would only be able to process a very limited data set. Many machine learning developers are using GPUs from Nvidia. Because of this, TensorFlow offers the CUDA acceleration option by default. It is, however, possible to use other hardware, and TensorFlow interfaces have been built for other desktop GPUs such as those from AMD [AMD], as well as embedded platforms like the Raspberry Pi [Rasperry] and Arm Mali GPUs [Mali].

It is possible to rapidly prototype machine learning models and train data sets using high level frameworks such as TensorFlow and Caffe2 using Python. As a language it does an excellent job of delivering a rapid, iterative development environment. In production, however, when the hardware available is more limited, for example in a mobile environment such as an autonomous vehicle or other battery operated environments, developers need to deploy the same applications and trained data sets without the power hungry systems and processors. Frameworks such as TensorFlow are too heavy-weight for this purpose consuming lots of energy and requiring large amounts of memory, and this is where C++ development is essential.

It is necessary to convert the same instructions developed using high-level frameworks into efficient and performant applications targeted at specific hardware.
The basis of modern machine learning, often described as "deep neural networks" due to their use of a vast networks of nodes, is a fast linear algebra implementation. The **Eigen library** [Eigen] is probably the world's most popular C++-based, high-performance dense linear algebra library. It implements matrix and vector operations, geometric transformations, numerical solvers, and related algorithms. Eigen uses template meta-programming techniques to allow developers to express complex linear algebra in a domain-specific language (DSL). The DSL allows expression trees [Expression] to be built at compile-time, and used to process data at run-time. The Eigen *tensor* (n-dimensional array) operations are used as part of the **TensorFlow** [TensorFlow] framework along with other Eigen operations. The library benefits from acceleration using heterogeneous hardware, since matrix and vector operations require many parallel calculations and are thus suited to parallelization on GPUs. Eigen can also be optimized for specific chipsets so that computing primitives (BLAS/LAPACK, etc.) supplied by the specific manufacturer can be used instead of something generic, thereby giving the developer the capability to maximize performance with relative ease. Other libraries include Armadillo [Armadillo] but there is numerous prior art. In fact SG14 Low latency has an SIG on Linear Algebra support in C++.

Developers will find other existing C++ libraries available to them that help in developing these applications for more constrained environments. ARM has developed their own inference engine [Inference] giving developers a set of optimized common operations used in machine learning applications built specifically for their range of GPUs. Other more generic frameworks are available with Nvidia having developed CuDNN [CuDNN] and in development is the SYCL-DNN [SYCL-DNN] project. These provide optimized C++ implementations of various operations such as highly tuned convolutions for CUDA and OpenCL devices respectively and can be deployed to a wider set of processors. The BLAS specification defines operations to perform basic vector and matrix operations and has been used to perform linear algebra for decades. Now the accelerated C++ based cuBLAS [cuBLAS] and syclBLAS [syclBLAS] libraries can be used for many of the operations used by machine learning developers.
While neural networks and deep learning seem to be dominating modern machine learning because of their flexibility and ease of use, there are situations when applying classical machine learning could be an advantage. For example, classical machine learning can perform better than deep learning on smaller data sets. Classical machine learning can also often be less computationally expensive than deep learning, which often requires high-end GPUs. Since neural networks typically follow a "black box" approach it is difficult to interpret the trained networks. However, classical machine learning employs mathematical and physical models that are easier to understand.
Classical machine learning algorithms are typically written in C++ and many benefit from acceleration offered by GPUs and other specifically designed processors. For example, the SYCL-ML [SYCL-ML] project is a prototype library showing what a pure modern C++ classical machine learning framework with acceleration using SYCL would look like. The project

implements several machine learning related algorithms including "Principal Component Analysis" (PCA), "Gaussian Mixture Model" (GMM) and "Support Vector Machine" (SVM).
The PCA can be seen as the equivalent of the few first layers of a neural network: it compresses the input data in a lower dimensional space. The GMM is a classifier based on the Expectation Maximization (EM) algorithm. It tries to fit every instance with the same label into any number M of Gaussian distributions. These distributions can then be used during the inference to compute which label is the closest to the new unseen data. The SVM is another classifier whose main idea is to select which samples in the training data represent the best the boundaries between each labels. These samples are then used to compute separators between classes.
The GMM is very efficient on a GPU as the most expensive operations are matrix multiplication and inversion. PCA and SVM are harder to parallelize but the GPU still shows some benefits over a CPU. Using C++ also allows the implementation to easily fuse the kernels that need to be fused giving further performance improvements.

Scope

In a teleconference held in the week before the San Diego meeting (Oct 30, 10 AM ET), we gathered industry and academic experts to discuss the need for C++ SG. The following response and summary of that meeting offers a strong case to support the creation of such a group. During the call, we discussed some of the following scope that could be covered by the ML SG for improvement in C++:

- fundamental arrays, matrix, vectors, tensors, linear algebra
- facilitate better support for interchange of in memory information/data between packages
- basic graphing
- optimization , quantization, parallelism, batching computations of vector, matrix, tensors
- packaging to allow adding computation/data manipulation/scaling packages + dependency
- supporting portability to various hardware embedded inference engines, and up down convert of different FP sizes between training and inference.
- support of exchange formats (ONNX, NNEF)
- support for kernel fusion on training and inference
- support of accelerator dispatch to inference engines, GPUs, FPGAs, MPSoC, Tensor Processing Units, Coarse Grain Reconfigurable Arrays, many of the newer ML boards from Xilinx, Google, ARM, Wave Computing, Nvidia
- do we need a graph extraction pass on top of C++
- support interoperability with data formats from Python and R packages
- Other C++ ML/Data Analysis libs : shark, MLpack, dlib, root [root]
- Integration Xla, tvm, tensor-rt, glow
- lazy evaluation execution graphs/workflows
- graph and tree data structures

These form an initial scope for the SG but there could be other additions.

The attendees of the call included the following and we highlight some of their comments. In this call we also discussed the sub-categories including Scope, Frequency of meeting, Outreach and Liaison. These are described and summarized in subsequent sections. The end result was that we submitted to the Convener, Herb Sutter who was also on the call, to start such an SG.

Emad Barsoum, MS; make a script that allows iteration
Need support for differentiable programming
Franke Seide, MS: working on framework on pytorch C++, language and library items, need Deferred computation expression with language support as first class citizen
Herb Sutter, MS: need to begeneral language features, evolving C++, please don't make it narrow, Reflection work in progress
Peter Goldsborough, FB: FP 16 is needed as well as even more reduced precision type
C++ is a good language, almost like python, lack of ecosystem for plotting, notebook jupiter,
Ralph Potter, Codeplay
Sarthak Pati, U. Penn: concern if the next generation of coder don't use what C++ put forth, I see now in my research centre, their first go to is python, integrating with academic researchers, need to target academics, interoperate with numpy and panda libraries, Linear algebra, eigen, mdspan, map-reduce
Sebastien Messmer, FB: pytorch onnx, need generic language support for lib for low resolution quantization types, new types in futures, better device abstraction, target audience machine learning libraries (quantization types is available) or users (easy to write code)
Simon Layton, Nvidia
Michael Wong, Codeplay: differentiable computing merge with probabilistic programming;
Need dynamic graph generation for RNN, look at the scope list above and adds xla, tvm, tensor-rt as well as existing C++ ML libs
Yes, intend to reach out to Data Scientists who are now building towards a Machine Learning Language such as Michael I Jordan who has a language called RAY
John Lawson, Codeplay
Others:
Power POV;
We Need data scientists, finance, and medical imaging people added to the SG
Product are in C++, dynamic graph cannot work with python framework
C++ static typing helps with finding errors

In a poll, the group feel that we will be able to draw in experts from data science and AI community to have sufficient quorum for maintaining momentum in such a study group.

Since the call, at the SAN meeting where the intention to form the ML for C++ SG was announced, there has been significant additional interest.

Meeting frequency and means

If formed, we initially suggest that the SG would meet online to enable outreach to data scientists who cannot attend C++ Standard meetings. This would imply telecons/webex either weekly, bi-weekly, or monthly, in addition to F2F meeting at C++ Standard meetings

In a poll online, there seems to be enough for a quorum for a F2F meeting of this SG in Kona at the next C++ Standard meeting.

Outreach to ML/AI/Data Science community

Similar to SG14 meeting at GDC where gamers meet, we would propose BoFs/panels/tutorials at ML conferences to reach out to the ML/AI/Data Science community. These include CVPR, ICML, NIPS, and ICCV/ECCV as well as framework specific conferences for tensorflow, pytorch/Caffe, CNTK, and others.

We have intention to start an SG20@NCSA (National Center for Supercomputing Applications), as well as showcase our work at Medical Imaging conferences (MICCAI, SPIE, etc.).

We also intend to collaborate to publish papers to these and other conferences, as a means to reach out to groups interested in language support for ML.

Liaison with other groups

We intend to liaise with external groups such as:

Khronos Machine Learning group:Michael chairs SYCL within this group
https://www.khronos.org/machine-learning/

ISO JTC1/SC 42 for which there is a Canadian Mirror Committee
https://www.iso.org/committee/6794475.html

Future meetings

We propose the next meeting will be on selected based on a doodle poll to start on the week of Dec 10 after NIPS or NeurIPS.
https://doodle.com/poll/fdz2t4hn5qs3rs5x
I will close the poll on Sunday Dec, 9 after NeurIPS.
I will book these calls from DST-next DST change to avoid confusion during DST changes between Europe and NA.

Thanks all for all your interest. Mailing list is also coming, but is awaiting ISO setup.

Conclusion

Due to the increasing performance requirements of machine learning and AI, not just for self driving vehicles but in many other industries, it is foreseeable that the C++ interfaces of these machine learning libraries will continue to gain importance as applications are deployed to real world situations.
The hardware landscape in this space is growing and expands as more vendors turn their focus to machine learning systems and applications. While Nvidia is currently capturing the mind share of data scientists and developers who are currently working on prototypes using desktop and cloud GPUs, there are many challenges still to overcome in order to deploy these solutions into mass markets. AMD has its HIP and ROCm solution. There also increasing number of embedded AI solutions in the forms of hardware inference engines. Developers are beginning to have a choice on the solutions they can work with, and C++ development is clearly an important part of producing the end product offering flexibility, portability and the ability to deploy complex machine learning applications into embedded environments.

FInally, the purpose of this ML group in C++ is not to compete with other language. If anything, it is to ensure we work well with them through interoperability. We invite interested party to join and solicit researchers, academics, scientists, industry, engineers, and practitioners who support its outcome.

Acknowledgements

Vincent Reverdy's work has been made possible thanks to NSF Awards CCF-1647432 and SI2-SSE-1642411.

Michael Wong's work is thanks to Codeplay Software Ltd, ISOCPP Foundation, Khronos, and Standards Council of Canada.

References

[AMD] https://gpuopen.com/rocm-tensorflow-1-8-release/

[Armadillo] http://arma.sourceforge.net/

[Catapult] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN20Whitepaper.pdf

[CuBLAS] https://developer.nvidia.com/cublas

[CuDNN] https://developer.nvidia.com/cudnn

[Eigen] http://eigen.tuxfamily.org

[Expression] https://www.codeplay.com/portal/05-22-17-implementing-opencl-support-for-eigen-using-sycl-and-computecpp

[Inference] https://developer.arm.com/products/processors/machine-learning/arm-nn

[Maili] https://developer.arm.com/technologies/machine-learning-on-arm/developer-material/software-for-machine-learning-on-arm

[Nervana] https://spectrum.ieee.org/tech-talk/computing/software/nervana-systems-puts-deep-learning-ai-in-the-cloud

[Pixel] https://www.blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

[Rasperry] https://medium.com/tensorflow/tensorflow-1-9-officially-supports-the-raspberry-pi-b91669b0aa0

[root] https://root.cern.ch

[SYCL-DNN] https://github.com/codeplaysoftware/SYCL-DNN

[SYCL-ML] https://github.com/codeplaysoftware/SYCL-ML

[syclBLAS] https://github.com/codeplaysoftware/sycl-blas

[TensorFlow] https://www.tensorflow.org/

[TPU] https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip