Accelerating Machine Learning using BLIS

Santanu Thangaraj, Kiran Varaganti, Kiran Puttur, Pradeep Rao

Advanced Micro Devices, Inc

Introduction:

Taking advantage of low latency and hierarchical memory architecture of x86 is critical to boost the performance of computational intensive applications such as deep learning algorithms in AMD platforms. Machine Learning (ML) algorithms are primarily built on top of basic linear algebra subprograms (BLAS). Hence performance of these linear algebra routines directly impact the performance of ML algorithms. In our experiments we use Caffe [4], a deep learning framework implementation and compare its performance by linking against BLAS libraries such as BLIS [7] and OpenBLAS [9].

Existing BLIS library performs poorly when benchmarked with Caffe’s handwritten digits recognition (MNIST challenge [1], [10]) deep layer model. We addressed this shortcoming in BLIS library and optimize the library to perform better for machine learning frameworks. We refer to optimized BLIS library as AMD optimized BLIS library

BLAS specifications:

BLAS is a specification that prescribes a set of low-level routines for performing common linear algebra operations such as vector addition, scalar multiplication, dot products, linear combinations, and matrix multiplication. They are the de facto standard low-level routines for linear algebra libraries; the routines have bindings for both C and FORTRAN. Although the BLAS specification is general, BLAS implementations are often optimized for speed on a particular machine, so using them can bring substantial performance benefits. BLAS implementations will take advantage of floating point hardware such as vector registers and SIMD instructions. Examples of BLAS libraries include: OpenBLAS, University of Texas Austin’s BLAS-like Library Instantiation Software Framework (BLIS) and Intel Math Kernel Library (MKL).

Basic linear algebraic operations exposed by BLAS libraries forms the crucial component of Machine Learning algorithms. Many machine learning frameworks including Caffe [10], depend on BLAS libraries to provide the required linear algebra functionality and can link to any of the standard BLAS libraries.

AMD adopted BLIS as its new BLAS library. AMD will provide optimized BLIS library for their microprocessors based on the new x86 architecture codenamed “Zen”.
BLIS framework was designed to isolate essential kernels of computation that, when optimized, immediately enable optimized implementations of most of its commonly used and computationally intensive operations. BLIS is written in ISO C99 and available under a new/modified/3-clause BSD license.

Deep Learning with Caffe

Convolutional Neural Networks (CNNs) are successful class of DNNs. CNNs are computed using dense kernels that differ from traditional dense linear algebra routines. Accordingly, modern deep learning frameworks such as Caffe provides suites of custom kernels that implement basic operations such as tensor convolutions, activation functions and pooling. These routines represent the bulk of the computations when training a CNN, and thus account for the majority of its execution time. The deep learning community has been successful in finding optimized implementations of these kernels, but as the underlying architectures evolve, these kernels must be re-optimized, which is a significant investment. Optimizing these kernels requires a deep understanding of the underlying processor architecture, with careful scheduling of data movement, on-chip memory placement, register blocking, and other optimizations in order to get acceptable performance.

Role of BLAS in DNN:

The most important computational primitive in CNNs is a special form of batched convolution called spatial convolution [1] [5].

There are two inputs to the convolution: \( D \in \mathbb{R}^{NCHW} \), which forms the input data, and \( F \in \mathbb{R}^{KCRS} \), which forms the convolutional filters. The input data ranges over \( N \) images in a mini batch, \( C \) input feature maps, \( H \) rows per image, and \( W \) columns per image. The filters range over \( K \) output feature maps, \( C \) input feature maps, \( R \) rows per filter, and \( S \) columns per filter. Computing this convolution involves a seven-way nested loop, with four independent loops and three accumulation loops [5]. There are many ways of implementing this computation. The Caffe MNIST benchmark training algorithm implements by lowering the convolutions onto a matrix multiplication (GEMM). The GEMM gets invoked for small matrix sizes. Therefore the performance of small matrix GEMM directly impacts the performance of the training algorithm. The optimized GEMM implementations are provided by BLAS libraries.

Small matrix GEMM optimization:

The BLIS library has six loops [12] around the GEMM computation, with the outer loop parameters dependent on L3 cache size while the inner loops dependent on L1/L2 cache sizes. The packing of data, required by inner loops, is done to avoid TLB misses.
This approach gives better performance for really large matrices which does not fit entirely in the cache system but introduces unnecessary overhead for small matrix computations. We have optimized GEMM specifically for small matrix cases and observed significant performance improvements (refer figure 1).

For our benchmarks we have used the Caffe version 1.0.0.rc3, OpenBLAS 0.2.20, BLIS 0.2.1 public open-source repository and AMD optimized BLIS version x.y (TBD). The experiments were run on Ubuntu 15.04 operating system.

![Small Matrix - SGEMM](image)

**Figure 1.** Results of single thread SGEMM performance with BLIS public, OpenBLAS and BLIS optimized. Machine: AMD Naples, 64 cores, 256 GB RAM @ 3.2 GHz.

**Caffe MNIST performance improvement:**

The optimization of GEMM the performance of the Caffe MNIST has improved, as shown in Figure 2. The performance of the forward pass shows significant improvement (Lower is better) and Caffe performance as a whole has improved by 17%.
Looking at the backward pass performance numbers there is a room for further improvement. The reason is the significant amount of GEMM calls made during the backward pass requires transpose of the input matrices. This support is not supported yet by the small matrix code.

**Conclusion:**

Machine learning, Deep Neural Networks are significantly gaining traction across the Industries for its application in automating every day chores and bringing AI into everyday life. Most of the Machine learning frameworks links with BLAS libraries during compilation. BLAS forms the de facto standard low-level routines for linear algebra libraries. It is the layer upon which lot of other high level Dense Linear Applications (DLA) are based. Having highly optimized BLAS library is essential for accelerating the ML frameworks.

By optimizing the level 1 and level 2 subroutines and small matrix GEMM we are able to achieve significant boost in performance of Caffe run with MNIST. We could observe performance benefit is not limited to only Caffe, but also for the LAPACK [8] routines such as LU, QR and Cholesky.
Reference:


