Announcing the Compute Library 17.6

Not long has passed since we made the Compute Library publicly available on March 14th. Today we are pleased to announce the second public release of the library, adding many new functions to accelerate computer vision and machine learning on Arm Cortex-A CPUs and Arm Mali GPUs.

We are also going to share with you some of our most immediate plans for the future.

What is in this new release

The first step in the development of the Compute Library focused on building a rich collection of common low-level functions for image processing, computer vision and machine learning that were portable and performant across Arm processors. Our aim was to reduce time to market and enable developers and manufacturers to focus on high level use-cases and differentiation and not on re-implementing common functionality. In this last quarter, our team has continued developing new functions for the library. We have also extended the set of target platforms and testing workloads which helped us improve real life performance as well as improve reliability. Of course, we also fixed bugs and continued performance optimizations of key functions. If you are keen on the details here is our changelog.

Feedback from developers and partners in the last few months has played a driving role in prioritising our work and in summary, the key focus ares for us this last quarter have been:

- Machine Learning

- Lower precision
- Sub-tensors
- Histogram of Gradients (HOG)
- Optimization of functions for complex real-life workloads and popular networks

Machine learning is hot, and whilst this used to be a server and big data play in the past, it is clear today that mobile is the de-facto new ML platform. In addition to running inference on the client there are also good arguments in the long run for training to happen on device. With ML on device you can get your results faster, use less data, and can even complete certain tasks without being connected to a network. Training on the client can also enhance security: if your voice data, pictures, and location never leave your phone, it can't be intercepted. ML right now is a really hot area for device manufactures looking to add value to their products and for developers looking to innovate. Hence the focus on development of our functions has now shifted to machine learning primitives.

Lower-precision. When we originally developed the Compute Library we were targeting only the GPU and therefore our initial implementation was tuned for FP32 operations. As developers and partners gained knowledge and experience it became clear that machine learning could live with lower precision in order to boost performance, and that in some cases 16-bit and even 8-bit fixed point or integer was sufficient. Others agree on this too, some good reads on the topics include Peter Warden's blog on quantisation here as well as this site on SqueezeNet (here the same accuracy as AlexNet is achieved using just 6-bit weights!). Inspired by this we have started porting key functions to lower precision, and in this release, we have added support for fixed point signed 8 bit (QS8) to the various NEON machine learning kernels. Furthermore, several of our partners have indicated that some key image processing functions such as image segmentation benefit from lower precision than 32-bit, whilst 8-bit and fixed-point compromises accuracy and therefore FP16 is the best choice. For this reason, we are also planning to invest in FP16 ports of key functions going forward.

The following functions now support 8-bit for CPU:

- Activation Layer
- Batch normalization layer
- Convolution Layer
- Pooling Layer
- Soft max Layer
- Fully connected layer
- Normalization

Sub tensors. Another useful functionality we have added to the library is the support for sub-tensors, which allows the creation of tensors as a subset of a bigger tensor. The main reason is AlexNet. You can see the pipeline split in two after the first layer and join again at the end. By using sub-tensors, we can split and join without copying the data. It's also useful for computer vision if you want to process only a portion of the image without copying it.

Histogram of Gradients (HOG) is a dense feature extraction algorithm useful to determine shapes within an image. This is commonly used alongside a classifier such as for example SVM (Support Vector Machine) and has been a popular request from developers. Compute Library already supported HOG for NEON/CPU, we have now extended support for OpenCL/GPU by adding the following functions:

- CLHOGOrientationBinningKernel
- CLHOGBlockNormalizationKernel
- CLHOGDetectorKernel / CLHOGDescriptor
- CLHOGDetector
- CLHOGGradient
- CLHOGMultiDetection

There is also a component of HOG that cannot be parallelised and therefore we supply as a vanilla C++ kernels:

- CPPDetectionWindowNonMaximaSuppressionKernel

Complex workloads. Our work analysing complex workloads/networks such as Google Inception, Resnet and Facebook DeepFace has also enabled us to pin-point some further optimization hotspots for ML so we added the following functions to the Compute Library for both OpenCL/GPU and NEON/CPU:

- CLBatchNormalizationLayerKernel / CLBatchNormalizationLayer (Resnet)
- CLDepthConcatenateKernel / CLDepthConcatenate (Google Inception)
- CLLocallyConnectedMatrixMultiplyKernel / CLLocallyConnectedLayer (Facebook Deep Face)

We have also added the option for the user to re-shape the weights of the network as they are being loaded so that this step can take place ahead of execution. This can improve performance in some cases:

- CLWeightsReshapeKernel / CLConvolutionLayerReshapeWeights

New NEON kernels / functions:

- NEDirectConvolutionLayerKernel / NEDirectConvolutionLayer

The runtime/scheduler. The library includes a basic scheduling component which is useful to enable the NEON code to be distributed across multiple CPU cores (using threads). The primary purpose of the runtime (which is just a basic example) is to: chain kernels to achieve complex functions (example: HOG/SVM, Canny edges etc.), memory management, multi-threading. In this release, we have improved some of the functionality of the scheduler and its interface as well as support for OpenMP.

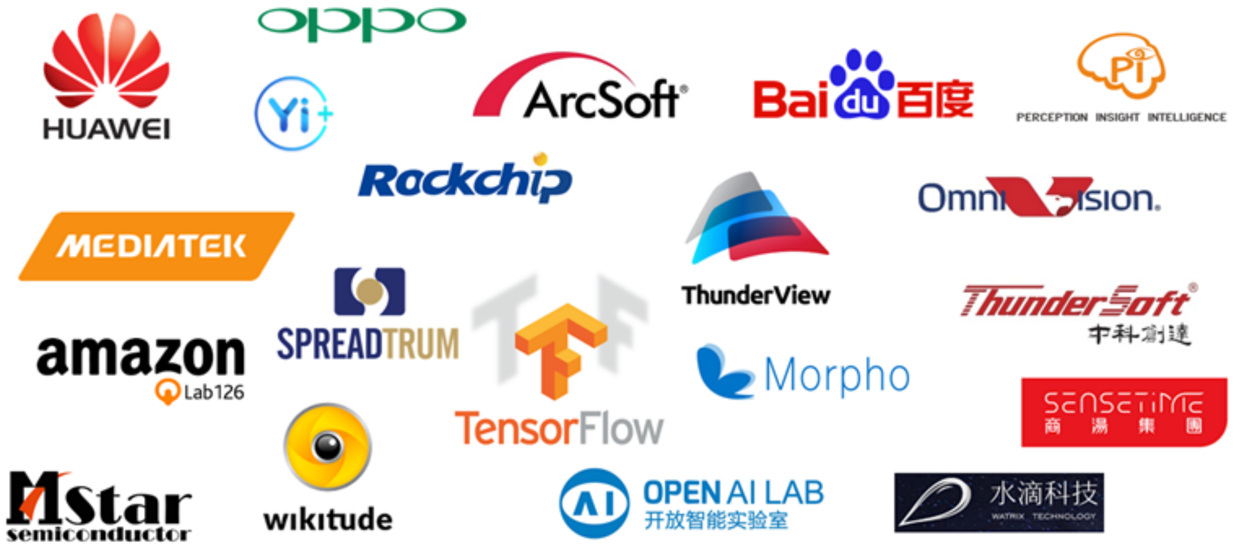
Exceptional adoption

Computer Vision and Machine Learning are hot topics, and we expected good traction for the library on release, however adoption was exceptional!

Some numbers:

- 576 clones (of which 420 unique clones) up from 260+ clones 2 months ago
- 4432 unique code viewers (up from 2.5k+ two months ago)
- 23k+ reads of launch blog

Since launch, a large number of Arm Partners have been evaluating or taking advantage of the Compute Library in their development. Here is a taste of some of them:



The list highlights the breadth of applicability of the Compute Library, as it has been used by partners in various market segments, and across the supply chain, from start-ups to established larger technology suppliers and verticals. The use of the Compute Library has enabled several new ventures: PerceptIn, a Chinese start-up developing next-generation robotics platform being incubated in Arm Accelerator, recently announced their case-study of enabling embedded deep learning inference engine with the Arm Compute Library. Read more about this in Evens Pan's blog.

Open AI Lab, a joint venture between Arm and ThunderSoft, has been working on porting the Caffe open source framework to take advantage of the Compute Library primitives, and the resulting work has been published in the CaffeOnACL project. We expect this effort eventually to make it upstream in the main Caffe branch.

Just getting started?

Want to learn more about the library? I can recommend this introduction blog from Gian Marco Iodice who explains how to use the Compute Library on Raspberry Pi

We have also recently released a webinar that discuss some computer vision and machine learning optimization use-cases using the Compute Library: here.

What's next

We endeavour to make one public release each quarter. The next release is scheduled for later in September 2017.

Key focus areas for next quarter are:

- The integration and optimization of key functions for popular ML frameworks (including the new Neural Network API for TensorFlow Lite announced by

Google at I/O - read more about it in Dave Burke's blog here - as well as Facebook's Caffe2 framework, maybe others too)

- Further complex workloads optimizations
- Further support for low precision: fixed point 8-bit and 16-bit, floating point 16-bit support for each of CPU and GPU
- Initial integration of CPU micro architectural optimizations for selected processors and for key functions
- Initial dot product support (Armv8.2), a feature that will improve CPU performance for key ML workloads (read more about the Arm-v8 architecture evolution here)

0 comments 0 members are here

ed

Cartoonifying Images on Raspberry Pi with the Compute Library

Gian Marco Iodice
2 months ago
Graphics & Multimedia
Graphics & Multimedia blog

Hi folks! Here we are! For the first hands-on guide of the new Computer Vision and Machine Learning software library developed at Arm: Compute Library! Compute Library is a rich collection of functions for image processing , computer vision and machine...

Arm Compute Library for computer vision and machine learning now publicly available!

Roberto Mijat
6 months ago
Graphics & Multimedia
Graphics & Multimedia blog

Emerging technologies like machine learning (ML), virtual reality (VR), augmented reality (AR), and computer vision (CV) are providing fantastic opportunities for innovation and business growth across the whole Arm ecosystem and in all market segments...

Google and the quest to bring Artificial Intelligence to the

Pablo fraile
3 months ago
Graphics & Multimedia
Graphics & Multimedia blog

masses

I recently attended the 11 th Google I/O, a conference where 7000 developers gather near Google’s headquarters to learn about, and celebrate, everything new that Google announces across its increasing range of products and services. The announcements...