RESEARCH ARTICLE | JANUARY 15 2019

Ultrafast processing of pixel detector data with machine learning frameworks FREE

G. Blaj ✉; C.-E. Chang; C. J. Kenney

AIP Conf. Proc. 2054, 060077 (2019)
https://doi.org/10.1063/1.5084708

Ultrafast Processing of Pixel Detector Data with Machine Learning Frameworks

G. Blaj[1,a)], C.-E. Chang[1] and C. J. Kenney[1]

[1]*SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA 94025, U.S.A.*

[a)]Corresponding author: blaj@slac.stanford.edu

Abstract. Modern photon science performed at high repetition rate free-electron laser (FEL) facilities and beyond relies on 2D pixel detectors operating at increasing frequencies (towards 100 kHz at LCLS-II) and producing rapidly increasing amounts of data (towards TB/s). This data must be rapidly stored for offline analysis and summarized in real time for online feedback to the scientists. While at LCLS all raw data has been stored, at LCLS-II this would lead to a prohibitive cost; instead, enabling real time processing of pixel detector data (dark, gain, common mode, background, charge summing, subpixel position, photon counting, data summarization) allows reducing the size and cost of online processing, offline processing and storage by orders of magnitude while preserving full photon information. This could be achieved by taking advantage of the compressibility of sparse data typical for LCLS-II applications. Faced with a similar big data challenge a decade ago, computer vision stimulated revolutionary advances in machine learning hardware and software. We investigated if these developments are useful in the field of data processing for high speed pixel detectors and found that typical deep learning models and autoencoder architectures failed to yield useful noise reduction while preserving full photon information, presumably because of the very different statistics and feature sets in computer vision and radiation imaging. However, the raw performance of modern frameworks like Tensorflow inspired us to redesign in Tensorflow mathematically equivalent versions of the state-of-the-art, "classical" algorithms used at LCLS. The novel Tensorflow models resulted in elegant, compact and hardware agnostic code, gaining 1 to 2 orders of magnitude faster processing on an inexpensive consumer GPU, reducing by 3 orders of magnitude the projected cost of online analysis and compression without photon loss at LCLS-II. The novel Tensorflow models also enabled ongoing development of a pipelined hardware system expected to yield an additional 3 to 4 orders of magnitude speedup, necessary for meeting the data acquisition and storage requirements at LCLS-II, potentially enabling acquiring every single FEL pulse at full speed. Computer vision a decade ago was dominated by hand-crafted filters; their structure inspired the deep learning revolution resulting in modern deep convolutional networks; similarly, our novel Tensorflow filters provide inspiration for designing future deep learning models for ultrafast and efficient processing and classification of pixel detector images at FEL facilities.

INTRODUCTION

Modern photon science was enabled by advances of 2D pixel detector technology in the last decades [1]. After the initial success of hard X-ray free-electron laser (FEL) facilities the last decade, two high repetition rate facilities are under development: EuXFEL (27 kHz) [2] and LCLS-II (100 kHz) [3]. Ultrafast data acquisition and processing will be critical in multiple applications: for online monitoring, data reduction (photon counting), sparsification, storage, and offline analysis.

A typical LCLS large area detector [4] (2.2 Mpixel/frame, 2 bytes/pixel, 120 Hz) produces ~0.5 GB/s and all raw data is saved, resulting in tens of petabytes of data requiring storage and analysis for the first 8 years of operation of LCLS [5, 6, 7]. Similar detectors operating at 100 kHz in LCLS-II will produce a massive 430 GB/s [8]. The accelerator and detector [8] developments to meet this high rate are well underway. The data transfer and storage requirements can be met with multiple parallel lanes and larger versions of the existing data acquisition (DAQ) system [9].

Raw images include noise, leading to inefficient lossless compression (typically a factor of ≈ 2); typical LCLS-II images will be sparse, allowing a much higher compression rate after photon counting. If image processing (photon counting and sparsification) can take place in real time, sparse photon data is much easier to compress and use in online and offline analysis, resulting in dramatically lower requirements on real time storage and data analysis. For

Proceedings of the 13th International Conference on Synchrotron Radiation Instrumentation – SRI2018
AIP Conf. Proc. 2054, 060077-1–060077-9; https://doi.org/10.1063/1.5084708
Published by AIP Publishing. 978-0-7354-1782-3/$30.00

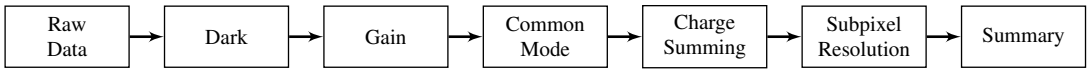

FIGURE 1. Typical steps in processing data from X-ray pixel detectors. Usually, Photon Finding and Subpixel Position are not extracted due to the high computational complexity, however, extracting them greatly increases data quality, enabling sparsification and efficient compression without photon loss. In this paper, we present complete Tensorflow models for all steps.

this reason, the prohibitive cost of storage and analysis of raw data at LCLS-II led to the development of a first version of the LCLS-II Data Reduction Pipeline [7]. However, an impractically large number of servers would still be required for real time reduction of 2D pixel detector data at 100 kHz.

Another area with rapid development in the last decade, driven by large increases of available data sets and stringent speed and efficiency requirements, is machine learning (especially computer vision applications). Several frameworks have been developed to allow scaling and optimization of system performance, leveraging advances in GPU technology while hiding system complexity, enabling simultaneously ultra rapid prototyping and efficient production systems; one such framework is Tensorflow, used for tensor (i.e., n-dimensional array) computation on CPUs and GPUs. Tensorflow allows a high level of abstraction, with compact, device-agnostic code, and orders of magnitude higher performance on GPUs compared to CPUs.

We present Tensorflow versions of usual, state-of-the-art algorithms used at LCLS for all steps of processing images from X-ray integrating pixel detectors and reducing it to photons. Due to the short pulse duration at FELs (in the order of femtoseconds), photon counting detectors can't be used; instead, integrating detectors must be used [10]. Raw data from integrating pixel detectors requires (or at least benefits from) the following processing steps (Fig. 1): dark subtraction, common mode correction, gain (mode) calibration, single photon reconstruction (charge summing, subpixel resolution), and summarization (including histogramming, photon counting and sparsification, and data aggregation). We then compare their programming complexity and performance on both CPU and GPU with usual solutions using compiled Numba code.

Using inexpensive consumer GPUs, we show 2 orders of magnitude improvement in data processing speed compared to "classical" CPU code (or 3 orders of magnitude improvement in cost for the same speed), already enabling real time processing and high rates of lossless compression for sparse data at LCLS with modest hardware costs. The approach presented here covers only 3 of the 6 orders of magnitude cost reduction required to match the LCLS-II speed requirements; however, it enables the development of an optimized, pipelined hardware approach which turns a prohibitively expensive task in a merely challenging one.

METHODS

Similar to the big data challenges of upcoming high repetition rate FELs were the challenges in computer vision a decade ago, which led to the current explosion of machine learning (ML) hardware and software and ongoing rapid development. One of the software frameworks for machine learning is Tensorflow, presenting a number of compelling advantages: Native Python support and trivial installation; compact, elegant code, enabling rapid development; hardware agnostic code enabling scalable deployment; native support for e.g., GPUs enabling high performance (requires the Nvidia CUDA Deep Neural Network library). Many other frameworks exist now.

Note that existing machine learning approaches to computer vision are not directly applicable to X-ray photon detection due to very different features, signal and noise statistics; while, e.g., autoencoders or deep learning models currently lose photon information, the models presented here preserve full photon information. For this reason, we developed novel low-level filters for all steps required to process data from 2D X-ray pixel detectors, which are mathematically equivalent to the current state-of-the-art algorithms for photon processing in 2D integrating pixel detectors at FELs.

To evaluate the performance of our Tensorflow models in challenging conditions, we intentionally maximized charge sharing [11] by (1) using a Hammerhead ePix100 camera [12], built with a novel sensor with a thickness of 500 μm and high aspect ratio pixels of 25 μm × 100 μm, (2) operating the camera with the sensor underbiased (at 100 V, just over the minimum bias for full depletion at 90 V), and (3) using relatively soft X-rays at 5.9 keV generated with an ^{55}Fe radioactive source. The Hammerhead ePix100 camera benefits from the high signal to noise ratio of ePix100 cameras [6]. We acquired a large data set of 50 000 frames (each containing 1 MB of data from 176 × 3072 pixels, equivalent to the typical ePix100 camera module of 704 × 768 pixels).

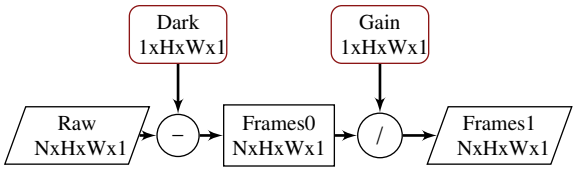

FIGURE 2. Tensorflow models for dark and gain correction are trivial. We provide raw data in "Raw" input, formatted in a rank 4 tensor with N frames (minibatch size), image size HxW=704 × 768 pixels, and "feature size" 1 (correspoding to pixel intensity). The dimensions of each tensor are indicated in the diagrams. After dark subtraction and gain correction, we obtain the corrected minibatches Frames0 and Frames1, respectively. Accomodating multiple darks and gains in auto-ranging detectors is also straightforward (not shown here).

We used two different computers: (1) CPU system: Macbook Pro, 2.9 GHz Intel Core i7, 16 GB 2133 MHz LPDDR3, Numba 0.38, Tensorflow 1.1; and (2) GPU system: Nvidia GTX 1060, Intel Xeon X3480, DDR3-1066 CAS latency 7, Tensorflow 1.4. We ran the Tensorflow models on both the CPU and GPU systems, comparing the performance with compiled Numba code (carefully optimized) running on the GPU system. The processing times we measured exclude the time to read data files, similar to real time data acquisition systems (receiving data from detectors over multiple 10 Gbps links simultaneously in real time).

NOVEL TENSORFLOW MODELS

Dark and Gain

While dark and gain corrections are trivial [13], we present diagrams in Fig. 2 to introduce the basics of tensor computation in Tensorflow. Stacks of frames called "minibatches" are submitted using rank 4 tensors: the first dimension, N, represents the number of frames in a minibatch; second and third dimensions represent the image size H x W, where H and W are the number of rows and columns, respectively. The last dimension represents a "feature" size (which is 1 for monochrome images, 3 for red-green-blue images); features can also represent other properties, e.g., matching a list of particular shapes). Subtraction and division are trivial and Tensorflow broadcasts the singleton dimensions automatically (similar to, e.g., Numpy).

Common Mode

The common mode model in Fig. 3 appears complex, however, its operation is relatively simple: the pixels without signal (i.e., pixel signal smaller than 3σ) are segmented, the number of segmented pixels is calculated in the top branch, their total signal is calculated in the middle branch, and their ratio (i.e., the mean pixel background) is obtained and subtracted from the input signal [13]. Note however the $\sqrt{HW}$ term which guarantees that the model works correctly also at high photon occupancy (when too few pixels measure the background and estimate a common mode correction based on only a few pixels, thus introducing spurious noise). If the camera is known to operate only at low photon occupancy, a much simpler and faster model can be used instead. While the model we present calculates the frame common mode, other types of common mode (row, column, block) can be easily extracted by reshaping tensors accordingly and summing only along the relevant dimensions.

Photon Charge Summing

At low photon occupancy (sparse photons) and sufficient signal to noise ratio (typically the case for modern radiation imaging pixel detectors), individual photons can be measured. Their signal is often split over several pixels, leading to charge sharing and changes in the spectrum of deposited energy. To minimize the effect of charge sharing [14], first we match the different possible charge cloud shapes with the detected shape. Typical computer vision applications achieve this by using a convolution with matching filters represented by 4-rank tensors, as shown in Fig. 4: F0 and F1 have a shape of [4, 5, 1, 6], corresponding here to the 4×5 kernel that is used to convolve ($*$) at each pixel location; the dimension 1 must match the number of features of the input data set (in our case, 1, see Fig. 5), and the last dimension is the number of features that we are extracting (i.e., the different images shown in each horizontal plane of F0 and F1).

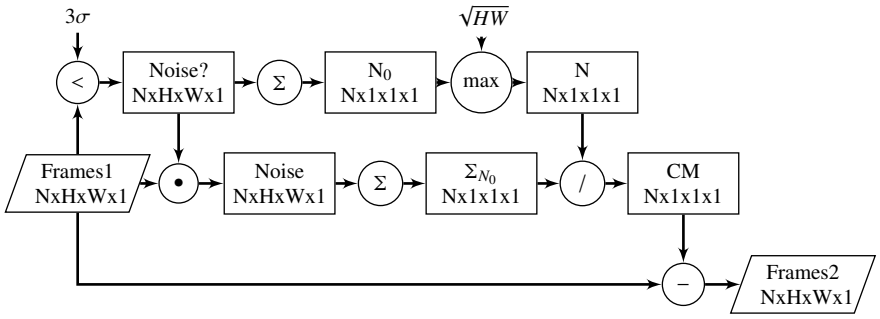

FIGURE 3. Tensorflow model for common mode correction appears complex, however, most of the diagram performs a more robust noise segmentation and average, which scales to high beam intensity without introducing spurious noise when only a small fraction of pixels sample the noise (within 3σ of 0); this design can be simplified at low photon occupancy (i.e., sparse photons).

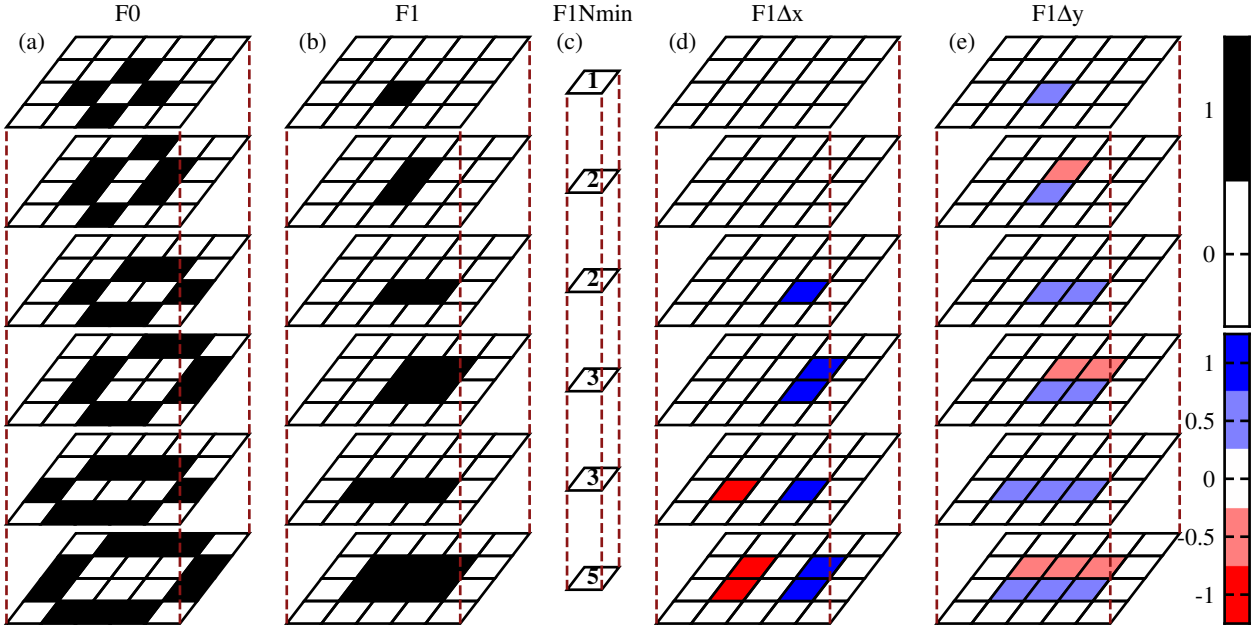

FIGURE 4. Photon finding relies in matching the shape of the photon signal (distributed over one or more neighboring pixels) with one of the 6 features in filter F1 (b). The shape is identified by sufficient overlap with filter F1, as determined by the F1Nmin tensor (c), while not extended over the edge defined by filter F0 (a). The full convolutional model is shown in Fig. 5. Subpixel resolution can be obtained as shown in Fig. 6, using convolutional filters F1Δx and F1Δy depicted in (d) and (e), respectively. Typically only the first 4 features are used (we used 6 to match the charge sharing resulting from narrow 25 μm pixels and underbiasing). F0, F1 and F1Nmin must be judiciously matched to detect all shapes of interest while preventing multiple feature matching.

Typical computer vision applications for feature segmentation (and object identification when using a deep learning model, i.e., a deep stack of convolutional layers) use a relatively fuzzy approach, with individual features obtained by training. Unlike in the deep learning approach, we designed two model branches with fixed weights (requiring no training and providing a clear physical meaning to each value) that strictly verify if the photon cloud shape conforms to one of the features in F1 while remaining within the boundaries defined by F0. Finally, the logical AND operation ($\wedge$) determines if there is a valid match. F0, F1 and F1Nmin must be chosen judiciously to ensure that any photon cloud shape will match only one feature, or none.

The top 4 layers of F0 and F1 unambiguously identify each of the 13 "event types" identified in [14] (the key is the particular choice of values in test vector F1Nmin). With successful shape matching, the total charge of the photon cloud is assigned to the pixel with the highest signal (using the subpixel information extracted in next subsection to

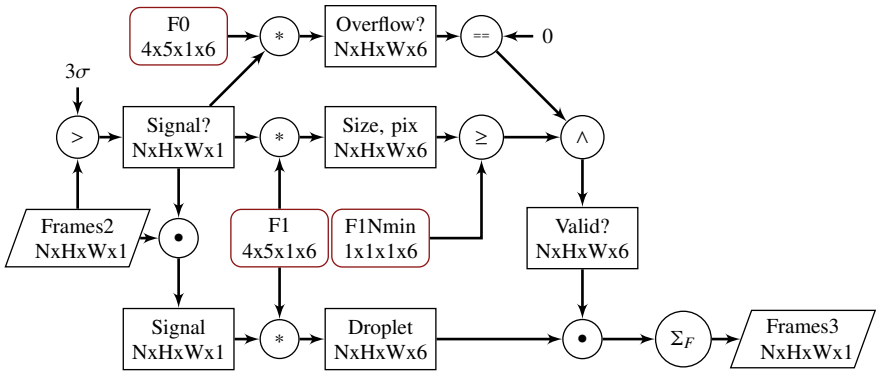

FIGURE 5. Photon charge summing is based on several convolution operations (∗) for validating photon hits and calculating the corresponding photon energy. The output is either the reconstructed photon energy placed in the appropriate pixel (for valid hits) or zero otherwise. Careful design of filters F0, F1 and F1Nmin allow detecting hits with arbitrary shapes in the different feature layers F while preventing multiple detection. This entire model is implemented in eight lines of Tensorflow code.

correctly assign the charge in the ambiguous case of even number of rows or columns for the charge cloud). The additional two layers elegantly extend the model to process charge clouds up to 2×3 pixels (and lead to a large increase in the number of event types), in order to process larger charge clouds. For typical pixel detectors, the first 4 features are sufficient.

This algorithm is mathematically equivalent with the one described in [14], which is also the state-of-the-art in charge summing with integrating detectors at LCLS. Our implementation is a generalization to charge clouds up to 2×3 pixel clusters and tests all possible charge cloud shapes at every pixel; the computation does not depend on the number of photons, and the maximum number of photons is returned. Typical droplet algorithms use an extra threshold parameter for detecting pixels where all the possible shapes are searched, thus missing some of the photons, and the computation time increases linearly with the number of photons; our model avoids both of these problems (while being 1 to 2 orders of magnitude faster).

Photon charge summing only works with sparse photons (typical images at LCLS-II); areas with high photon fluxes can be converted easily to photon counting images by division with the expected single photon gain followed by rounding to the nearest integer.

Tensorflow enables designing complex models with very compact and elegant code; we described the entire model in Fig. 5 using only eight lines of code.

Photon Subpixel Resolution

The subpixel centroid Δx of photon charge clouds, measured in pixel pitches, can be easily calculated using the F1Δx filters as shown in Fig. 6. The centroid position in the y direction can be calculated similarly, just by using the F1Δy filter. Note that the subpixel centroid has a nonlinear dependence on the actual position where the photon is detected.

This algorithm is mathematically equivalent to the "classical" algorithm described in [15], which is the state-of-the-art for subpixel resolution with integrating detectors at LCLS; our implementation includes a supplementary feature of zeroing the signal of pixels within noise, resulting in increased signal-to-noise ratio for some photons.

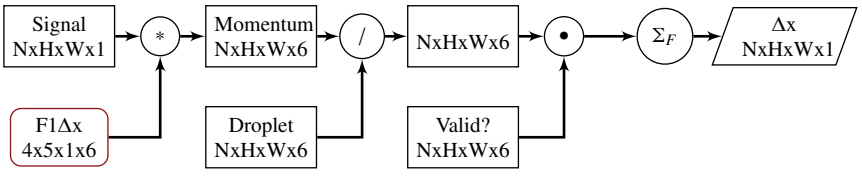

FIGURE 6. Extracting the Δx subpixel photon centroids (expressed in pixel pitches) is straightforward and fast, reusing intermediary layers from Fig. 5. Similarly, Δy can be obtained by using F1Δy. The centroids require linearization to yield actual positions.

Summarization

Tensorflow already has functions for histogramming, segmentation, sparsification, averaging and summing, etc. over arbitrary (combinations of) tensor dimensions; they are much faster than similar compiled functions, reducing massive data sets in real time and enabling online analysis with a minimum of overhead. Further functionality can involve, for example, identifying diffraction spots in real time. Once the input data is reduced to either lists of sparse photons, or photon counting images, this data is much smaller, enabling faster computation and greatly increased compression rates for offline storage, without loss of photon information.

RESULTS

Accuracy

The "classical" CPU algorithms and novel Tensorflow algorithms yield exactly the same results, photon by photon. The only parameters that can be changed are the two noise thresholds shown in Fig. 3 and Fig. 5, displayed as "3σ" and typically set between 3σ and 5σ [15]. Figure 7 shows a spectrum obtained in (intentionally challenging) conditions: thick sensor (500 μm), underbiased at 100 V (just over the minimum bias for full depletion, 90 V), with a small pitch in one direction (25 μm) and relatively soft X-ray photons (5.9 keV). The thick red line shows the raw histogram, before charge summing; the black line shows the result of charge summing, with the peak at its expected location (83 adu for 5.9 keV photons); the effect of charge sharing almost completely removed. Also the individual components with a charge cloud width of 1, 2 and 3 pixels are shown with thinner lines. Note that most detected photons have charge cloud widths of 2 pixels, some have 3 pixels, and almost none have a cloud size of 1 pixel. In Fig. 8 we show the subpixel centroids with the same detector settings (100 V bias). The majority of photon charge clouds are wider than 1 pixel, enabling recovery of subpixel position from subpixel centroids [12].

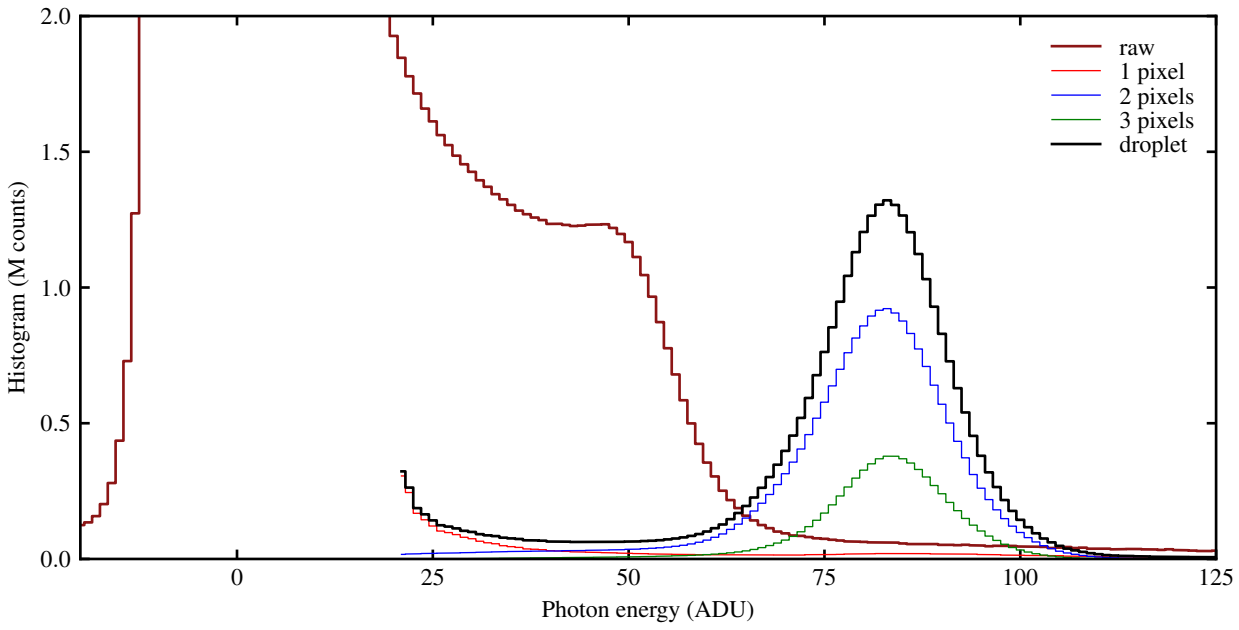

FIGURE 7. Performance of photon charge summing model: the thick red line shows a histogram of Frames2 (before); the thick black line depicts a histogram of Frames3 (after charge summing). Note the radically improved spectrum (matching expected single photon gain of 83 adu). The individual features obtained by summation over 1, 2 or 3 pixels are indicated by thin lines. The system is dominated by 2 and 3 pixel events.

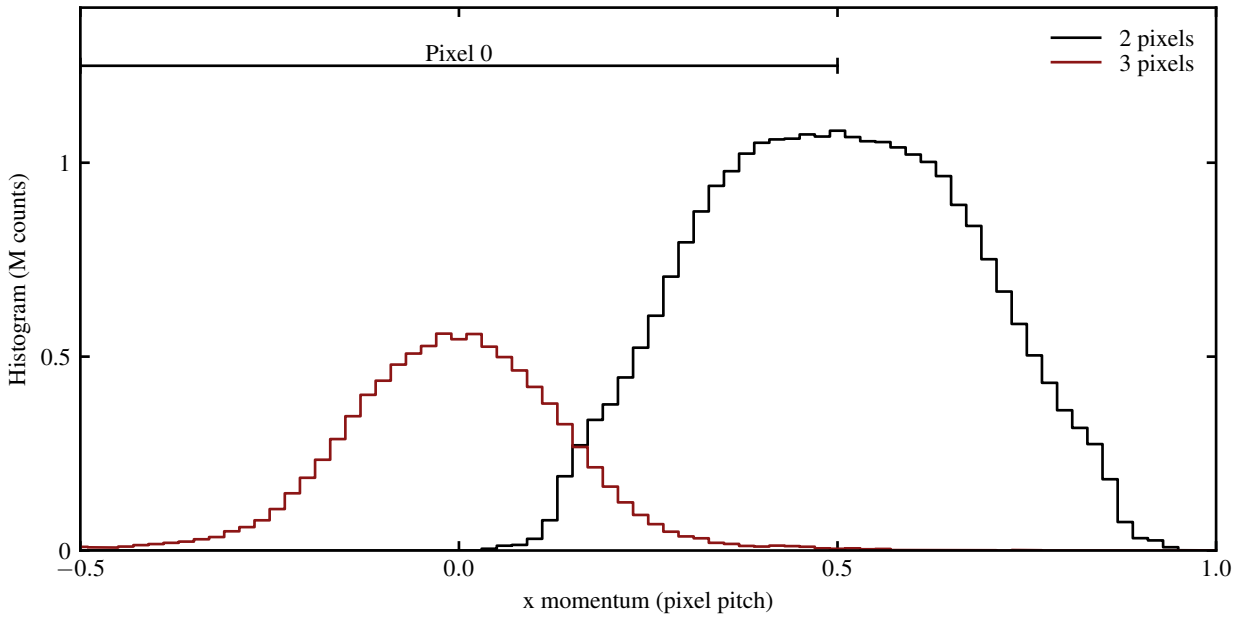

FIGURE 8. Histogram of subpixel centroids in the x direction; after linearizarion, photon position resolution in the order of μm is possible at low photon occupancy.

Performance

To compare the performance of the novel Tensorflow model with previously developed Numba compiled code (complex and carefully optimized), we built a model consisting of: a dark and gain stage, 3 stacked common mode stages, a charge summing stage with 6 feature kernels, subpixel centroiding on both x and y direction and full summarization (photon sparsification, accumulating photon counting maps, histograms). The results are shown in Fig. 9.

Tensoflow running on a CPU was 4 times faster than the Numba code running on the same computer. However, running the Tensorflow model on a modest, inexpensive consumer GPU unleashed its performance, achieving a 46 times speed improvement over the "classical" CPU code.

The raw data of 50 000 frames is about 50 GB in size (about 1 MB per frame); as it consists mostly of noise and relatively sparse photons, lossless compression yields about 25 GB. However, extracting the full photon information (energy, coordinates with subpixel precision, cloud shape) and saving it into a file enables two orders of magnitude compression without loss of photon data. This compression ratio depends strongly on the statistics of individual FEL applications; most LCLS-II applications are expected to yield sparse data. Systematic tests with representative data sets are currently underway.

Writing the Tensorflow model required very little effort beyond the initial designing and testing work; currently it consists of a few tens of lines of code, due to the elegant, implicit handling of boundary conditions and power of calculating all possible features simultaneously.

CONCLUSIONS

Tensorflow code is compact and elegant, enabling extremely rapid development; it supports Python and Numpy natively; and it is simultaneously hardware agnostic and highly scalable. Last but not least, Tensorflow and other similar frameworks will continue to benefit from advances in machine learning hardware and software with minimal or no code changes. Faster GPUs, new tensor processing units, and other developments are likely to appear in the future, speeding up current models.

The performance of Tensorflow code running on a modest GPU now exceeds by far the execution speed of highly optimized compiled code, reaching 63 Hz now with ePix100 cameras, and over 240 Hz with ePix10K cameras (sufficient to process data from two ePix10K cameras in real time). While it might be possible to build even faster

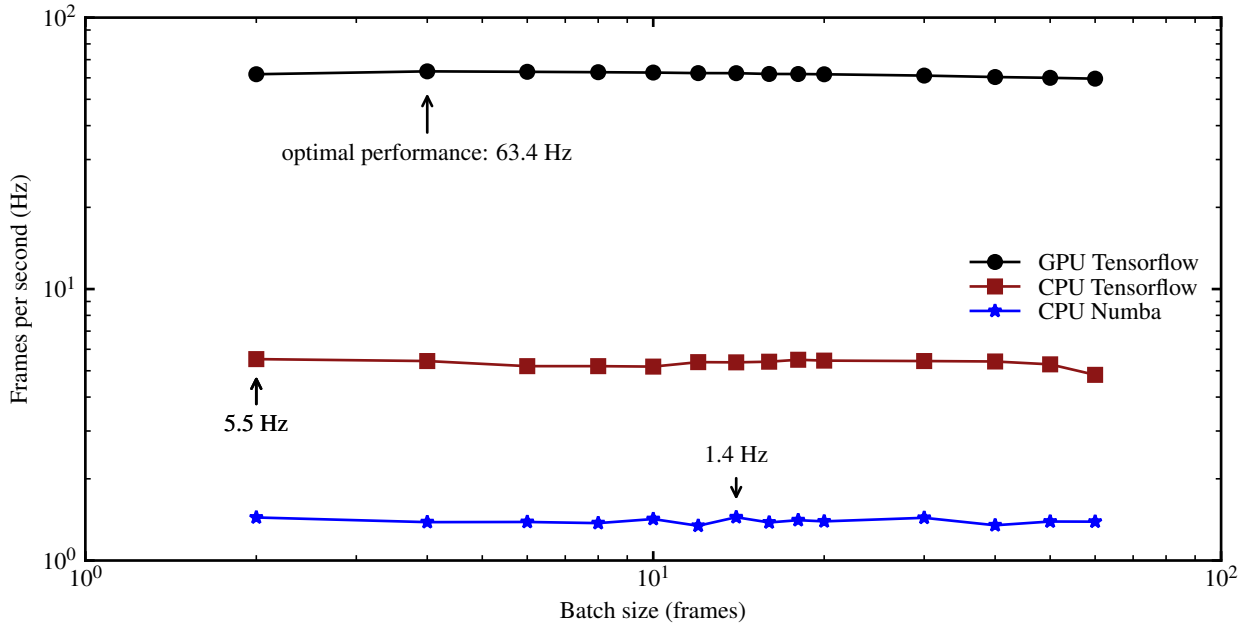

FIGURE 9. Performance chart of the Tensorflow model, shown on a log-log plot. Blue stars indicate the performance of "classical" CPU compiled code (using Numba with careful performance optimizations) with a processing rate of 1.4 Hz (decreasing at higher occupancy, as each droplet is calculated separately). Red squares depict the same hardware running Tensorflow on the CPU at 5.5 Hz, or 4x faster (note that higher occupancy does not increase computation time, as all features are calculated in parallel). Black dots show that the Tensorflow model is unleashed even by a modest consumer GPU, yielding 63.4 Hz, or a 46x speed increase over the "classical" CPU code, while producing identical results.

code using Nvidia CUDA, it would likely be difficult to develop and might depend strongly on the GPU architecture.

Our novel Tensorflow model already reduces by 3 orders of magnitude the projected cost of online analysis and compression without photon loss at LCLS-II, and enables the development of a pipelined hardware system which is expected to yield an additional 3 to 4 orders of magnitude cost reduction for processing the LCLS-II pixel detector data at full speed (supporting online analysis, reduction to photon data and compression). This will turn a prohibitively expensive task into a merely challenging one.

Computer vision a decade ago was dominated by hand-crafted filters; their structure inspired the deep learning revolution resulting in modern deep convolutional networks. Similarly, while each layer of our novel hand-crafted Tensorflow filters has a clear meaning, the filters will provide a starting point for development of future deep learning models for ultrafast and efficient processing and classification of pixel detector images at FEL facilities.

ACKNOWLEDGMENTS

Use of the Linac Coherent Light Source (LCLS), SLAC National Accelerator Laboratory, is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No. DE-AC02-76SF00515. Publication number SLAC-PUB-17277.

REFERENCES

1. P. Delpierre, Journal of Instrumentation **9**, p. C05059 (2014).
2. A. Koch, M. Kuster, J. Sztuk-Dambietz, and M. Turcato, Journal of Physics: Conference Series **425**, p. 062013 (2013).
3. R. Schoenlein *et al.*, *New science opportunities enabled by LCLS-II X-ray lasers* (SLAC Report, 2015) SLAC-R-1053.

4. P. A. Hart, S. Boutet, G. Carini, M. Dubrovin, B. Duda, D. Fritz, G. Haller, R. Herbst, S. Herrmann, C. J. Kenney, *et al.*, "The CSPAD megapixel X-ray camera at LCLS," in *SPIE Optical Engineering+ Applications* (International Society for Optics and Photonics, 2012), pp. 85040C–85040C.
5. G. Blaj, P. Caragiulo, G. Carini, S. Carron, A. Dragone, D. Freytag, G. Haller, P. A. Hart, R. Herbst, S. Herrmann, J. Hasi, C. J. Kenney, B. Markovic, K. Nishimura, S. Osier, J. Pines, J. Segal, A. Tomada, and M. Weaver, Synchrotron Radiation News **27**, 14–19 (2014).
6. G. Blaj, P. Caragiulo, G. Carini, S. Carron, A. Dragone, D. Freytag, G. Haller, P. Hart, J. Hasi, R. Herbst, S. Herrmann, C. J. Kenney, B. Markovic, K. Nishimura, S. Osier, J. Pines, B. Reese, J. Segal, A. Tomada, and M. Weaver, Journal of Synchrotron Radiation **22**, 577–583 (2015).
7. J. Thayer, D. Damiani, C. Ford, I. Gaponenko, W. Kroeger, C. O'Grady, J. Pines, T. Tookey, M. Weaver, and A. Perazzo, Journal of Applied Crystallography **49**, 1363–1369 (2016).
8. G. Blaj, P. Caragiulo, G. Carini, A. Dragone, G. Haller, P. Hart, J. Hasi, R. Herbst, C. J. Kenney, B. Markovic, K. Nishimura, J. Pines, J. Segal, C. Tamma, and A. Tomada, AIP Conference Proceedings **1741**, p. 040012 (2016).
9. R. Herbst, R. Claus, M. Freytag, G. Haller, M. Huffer, S. Maldonado, K. Nishimura, C. O'Grady, J. Panetta, A. Perazzo, *et al.*, "Design of the SLAC RCE Platform: A general purpose ATCA based data acquisition system," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2014 IEEE* (IEEE, 2014), pp. 1–4.
10. H. Graafsma, J. Becker, and S. M. Gruner, "Synchrotron light sources and free-electron lasers," (Springer International Publishing, Switzerland, 2014) Chap. Integrating Hybrid Area Detectors for Storage Ring and Free-Electron Laser Applications,, pp. 1–24.
11. G. Blaj, J. Segal, C. Kenney, and G. Haller, (2017), arXiv:1706.01429 [physics.ins-det], arXiv:1706.01429 [physics.ins-det] .
12. G. Blaj, D. Bhogadi, C.-E. Chang, D. Doering, C. Kenney, T. Kroll, J. Segal, D. Sokaras, and G. Haller, AIP Conference Proceedings (2018), submitted.
13. G. Blaj, P. Caragiulo, A. Dragone, G. Haller, J. Hasi, C. J. Kenney, M. Kwiatkowski, B. Markovic, J. Segal, and A. Tomada, SPIE Proceedings **9968**, 99680J–99680J–10June (2016).
14. A. Abboud, S. Send, N. Pashniak, W. Leitenberger, S. Ihle, M. Huth, R. Hartmann, L. Strüder, and U. Pietsch, Journal of Instrumentation **8**, p. P05005 (2013).
15. S. Cartier, A. Bergamaschi, R. Dinapoli, D. Greiffenberg, I. Johnson, J. Jungmann, D. Mezza, A. Mozzanica, B. Schmitt, X. Shi, *et al.*, Journal of Instrumentation **9**, p. C05027 (2014).