Ultrafast Processing of Pixel Detector Data with Machine Learning

SLAC NATIONAL ACCELERATOR LABORATORY

G. Blaj*, C.-E. Chang, C. J. Kenney

* blaj@slac.stanford.edu

Introduction

- Modern photon science was enabled by advances in 2D pixel detectors
- Free Electron Laser (FEL) requirements rapidly increasing:
  - Currently around 120 Hz, 1 GB/s (typical LCLS experiment)
  - Increasing rates towards 100 kHz (LCLS-II)
  - Increasing amounts of data, towards ∼1 TB/s (LCLS-II)
  - Accelerator, detector developments well underway
- LCLS-II data needs acquisition, storage, analysis:
  - Online analysis necessary (experiment optimization)
  - Storage and offline analysis necessary, expensive
  - Online noise reduction and compression necessary
- An LCLS-II 'Data Reduction Pipeline' for online processing:
  - First prototype designed
  - Servers with 'classical' multi-CPU processing
  - Requires hundreds or thousands of parallel servers
  - Parallel CPU code: is architecture dependent, complex
  - GPU code: strongly architecture dependent, complex

Fig. 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.

Machine Learning

- Similar challenges in computer vision a decade ago: led to current explosion of machine learning (ML) hardware and software and ongoing rapid development
- Tensorflow [1], a software framework for machine learning:
  - Compact, elegant code, enabling rapid development
  - Hardware agnostic code enabling scalable deployment
  - Native support for e.g., GPUs enabling high performance
  - Seamless integration with Python, Numpy
- We developed novel models[1] for pixel data processing in Tensorflow (implementing all steps in Fig. 1):
  - 1-2 orders of magnitude faster on modest hardware;
  - Optimization: expected orders of magnitude speed-up
  - Basis for future deep learning for radiation pixel detectors
  - Each layer, value: clear meaning (not a black box)
  - Effortlessly benefits from future ML advances

[1] Existing machine learning approaches to computer vision are not applicable to x-ray photon detection due to very different features, signal and noise statistics, etc.; while, e.g., autoencoders or deep learning models currently lose photon information, the models presented here preserve full photon information.

Methods

- ePix100 camera [2] intentionally operated in challenging conditions to maximize photon charge sharing:
  - Spectroscopic sensor with narrow 25 µm × 100 µm pixels
  - 500 µm thick, underbiased (100 V) Si sensor
  - 120 Hz, 704 × 768 pixels, 16(14) bit, 120 MB/s
  - ^{55}Fe source (emitting mainly Mn Kα at 5.9 keV)
- Large data set of 50 000 frames at low photon occupancy
- Processing computers:
  - CPU system: Macbook Pro, 2.9 GHz Intel Core i7, 16 GB 2133 MHz LPDDR3, Numba 0.38, Tensorflow 1.1
  - GPU system: Nvidia GTX 1060, Intel Xeon X3480, DDR3-1066 CAS latency 7, Tensorflow 1.4
  - Processing time excludes time to read raw data and buffer in RAM (similar to real time data acquisition systems which receive raw data with little overhead)

Dark, Gain Correction in Tensorflow

- Dark and gain correction are simple, see diagram in Fig. 2:

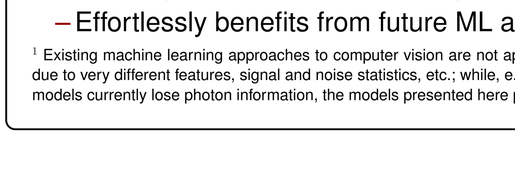
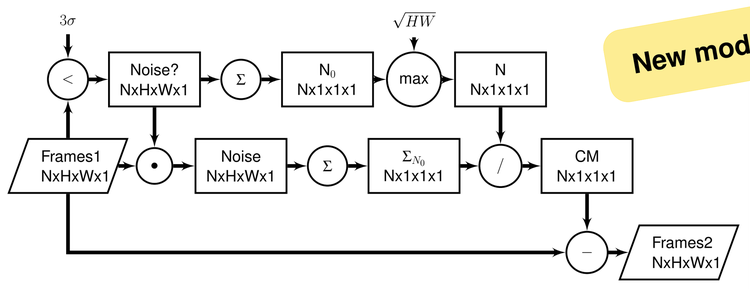


Fig. 2: The Tensorflow model for dark and gain correction is trivial. We provide raw data in 'Frames0' input, formatted in a rank 4 tensor with N frames (minibatch size), image size HxW=704 × 768 pixels, and 'feature size' 1 (correspoding to pixel intensity). The dimensions of each tensor are indicated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

Common Mode in Tensorflow

- Fig. 3 shows a robust estimator for frame common mode:



New model

Fig. 3: The 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 measure zero noise (within 3σ); this design can be simplified when operating only at low photon occupancy (i.e., sparse photons).

- This model can also reduce row and column common mode by summing along the corresponding dimensions.

Photon Charge Summing in TF

- At low occupancies, charge summing reverts the loss of spectroscopic information due to charge sharing of a photon signal over multiple pixels (inherent to pixel detectors)
- Uses two convolution kernels (see F0, F1 in Fig. 4)

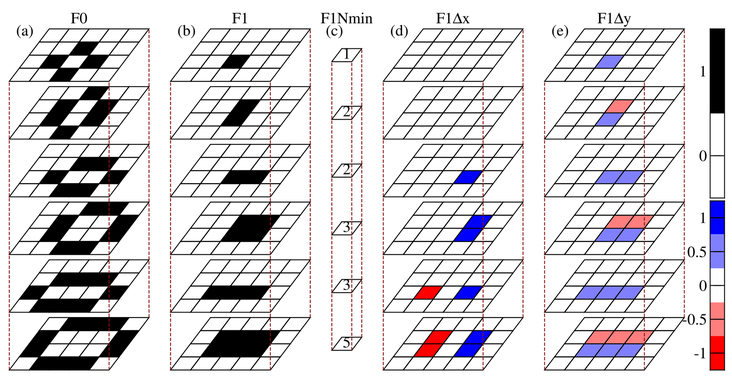
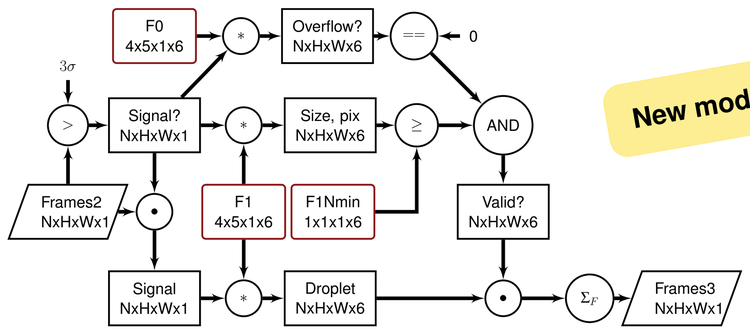


Fig. 4: Photon finding relies in matching the shape of the photon signal (distributed over one or more neighboring pixels) with one of the 6 features in filter F1 (b). The shape is identified by sufficient overlap with filter F1, as determined by the F1Nmin tensor (c), while not extended over the edge 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.

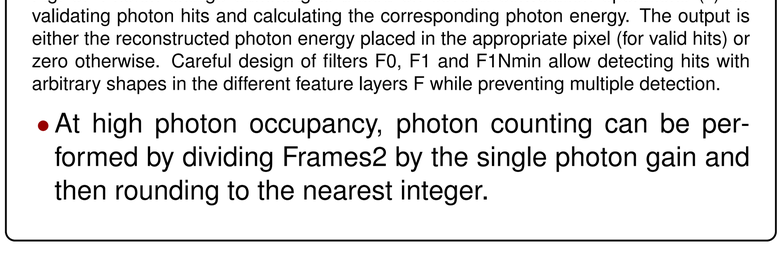


New model

Fig. 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.

- At high photon occupancy, photon counting can be performed by dividing Frames2 by the single photon gain and then rounding to the nearest integer.

Subpixel Resolution in Tensorflow



New model

Fig. 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 in Tensorflow

- Tensorflow functions for histogramming, sparsification, averaging over different dimensions, etc., enable reducing massive data sets to small summaries in a fraction of the time required by, e.g., similar compiled Numba functions
- These functions enable both rapid online analysis and online data reduction, before buffering and storage in the data acquisition system, without loss of photon data.

Results: Performance

- 1 to 2 orders of magnitude speed increase easily obtained, despite intentionally challenging images, inexpensive GPU, and inefficient first implementation:

Breakthrough

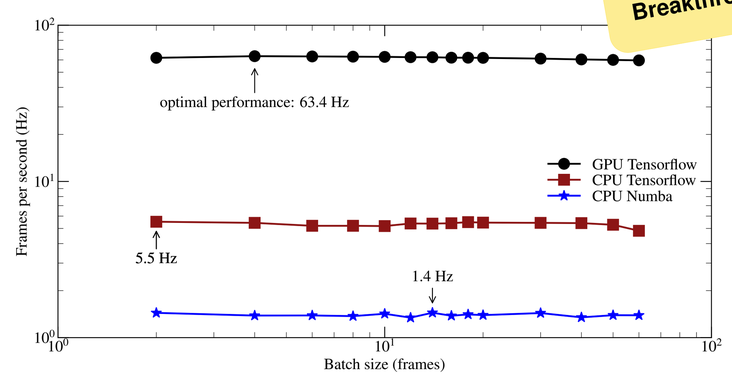
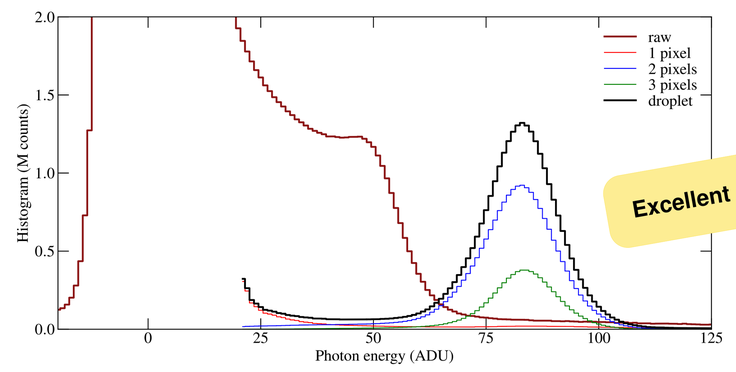


Fig. 7: Performance chart of the tensorflow model, shown on a log-log plot. Blue stars indicate the performance of "classical" CPU compiled code (using Numba with careful performance optimizations) with a processing rate of 1.4 Hz (decreasing at higher occupancy, as each droplet is calculated separately). 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.

Results: Accuracy

- Excellent performance, despite operating with (intentionally) challenging charge sharing, see Fig. 8 and 9.



Excellent

Fig. 8: 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.

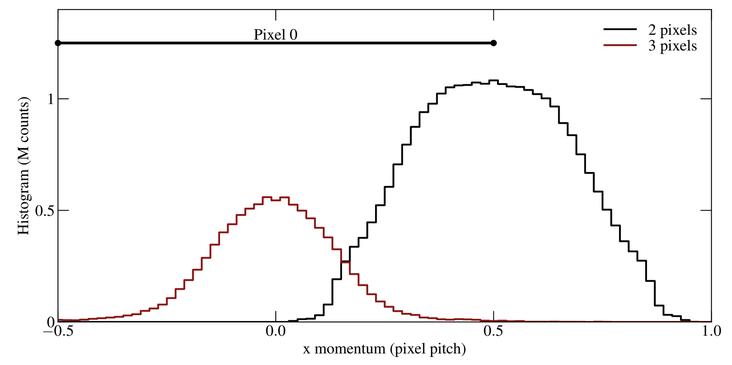


Fig. 9: Histogram of subpixel centroids in the x direction; after linearizarion, photon position resolution in the order of µm is possible at low photon occupancy.

Conclusions

- Tensorflow enables orders of magnitude faster data processing than standard CPU even with a consumer GPU
- Tensorflow code is compact, elegant, hardware agnostic, and will continue to benefit from advances in ML
- Our novel models greatly reduce the cost of online analysis and compression without photon loss at modern FELs and inspire future deep learning models for radiation imaging.

References

[1] Martín Abadi et al. Software available from tensorflow.org. 2015.
[2] G. Blaj et al. In: *SPIE Proceedings* 9968 (2016), 99680J–99680J–10.

Acknowledgements

U.S. DEPARTMENT OF ENERGY | Office of Science

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.