Matrix Sketching for Online Analysis of LCLS Imaging Datasets

John Winnicki
Institute for Computational and Mathematical Engineering
Stanford University
Stanford, USA
winnicki@stanford.edu

Frédéric Poitevin
Linac Coherent Light Source
SLAC National Accelerator Laboratory
Stanford, USA
fpoitevi@slac.stanford.edu

Haoyuan Li
Linac Coherent Light Source
SLAC National Accelerator Laboratory
Stanford, USA
hyli16@stanford.edu

Eric Darve
Institute for Computational and Mathematical Engineering, Mechanical Engineering
Stanford University
Stanford, USA
darve@stanford.edu

***Abstract*—X-ray light source facilities such as the Linac Coherence Light Source (LCLS) at SLAC National Accelerator Laboratory generate massive amounts of data that need to be analyzed quickly to inform ongoing experiments. The analysis of data streams coming from various parts of the instrument has potential to feed back into instrument operation or experiment steering. For example, shot-to-shot images of the beam profile inform on the quality of the beam delivery while downstream data read from large area detectors inform on the state of diffraction experiments carried on samples of interests at various beamlines. However, the high repetition rate and high dimensionality of these data streams make their analysis challenging, both in terms of scalability and interpretability. In this work, we propose an image monitoring and classification framework that follows a three-stage process: dimensionality reduction using principal component analysis on a matrix sketch, visualization using UMAP, and clustering using OPTICS. In the dimensionality reduction step, we combine the Priority Sampling algorithm with a modified Frequent Directions algorithm to produce a rank-adaptive accelerated matrix sketching (ARAMS) algorithm, wherein practitioners specify the target error of the sketch as opposed to the rank. Furthermore, the framework is parallel, enabling real-time analysis of the underpinning structure of the data. This framework demonstrates strong empirical performance and scalability. We explore its effectiveness on both beam profile data and diffraction data from recent LCLS experiments.**

***Index Terms*—matrix sketching, dimension reduction, parallel processing, approximation, rank-adaptive, data exploration**

I. Introduction

The Linac Coherent Light Source (LCLS) facility at SLAC National Accelerator Laboratory generates X-ray pulses by spontaneous undulator radiation from electrons. Through these pulses, X-ray images of atoms and molecules in action are generated, offering precise atomic-level insights into ultrafast events. These observations unveil essential phenomena in various domains, including materials science, energy sciences, and biology. By stitching these images together, the LCLS creates dynamic "molecular movies" that depict chemical reactions unfolding in real-time [3].

The facility generates images from multiple detectors synchronized by a timing system that timestamps images and other readouts across the instrument and pools them all into event objects corresponding to individual shots. Processing of the shot-to-shot events can have two usages: for instrument diagnostic or scientific analysis. The analysis of upstream diagnostic detector data, which are used to monitor the beam shape, or profile, enables labeling events as good or bad, thus informing the analysis of downstream measurement detectors that are collecting diffraction images of studied samples. For example, events with poor beam shape can be discarded from the downstream analysis. Or events might be grouped according to some beam profile characteristics, and downstream analysis can be performed on the different groups separately. Beam profiling can also be used directly as a diagnostic that helps operators improve the instrument's performance. And diffraction data from scientific samples can also be analyzed on its own merits.

A common challenge for both categories of detector image analysis is the rate at which the events are built. Typically, LCLS detectors read out at a frame-rate of 120 frame-per-second, generating gigabits of detector data each second. And the recent coming online of LCLS-II will increase these rates by a few order of magnitudes over the next decade. The sheer amount of data generated thus calls for scalable analysis methods that are able to present actionable information to the instrument operator or user in a timely manner. Examples comprise approaches to conveniently visualize these high-volume high-dimensional datasets or to accurately cluster them. Here,

979-8-3503-5554-3/24/$31.00 ©2024 IEEE
DOI 10.1109/SCW63240.2024.00269

we propose matrix sketching as a promising strategy to that end.

Matrix Sketching techniques find low-rank approximations to large matrices in a computationally efficient manner which are significantly smaller and simpler but still approximate it well. Recent advancements to Matrix Sketching have made it popular in online data processing and dimensionality reduction across a broad range of fields. The Frequent Directions matrix sketching algorithm has stood out in particular for its theoretical and practical error bounds, though lags behind other matrix sketching techniques in run-time performance [5].

II. Contributions

Our contributions are three-fold. First, we describe a two-stage rank-adaptive online matrix sketching algorithm to monitor LCLS imaging datasets, combining the priority sampling and frequent directions matrix sketching algorithms to balance run-time with error. This algorithm finds a low-rank approximation of the data. Second, we present a scheme for parallelization of this algorithm and provide strong scaling studies for this scheme. To our knowledge, our proposed modifications of the frequent directions algorithm in acceleration via priority sampling, applying rank adaptation, and parallelization in this branching fashion have not been studied. Third, using this low rank approximation to project the original data into latent space, we propose a technique for monitoring imaging datasets in a scalable and efficient manner by visualizing and clustering the images using UMAP and OPTICS clustering. We demonstrate the effectiveness of this technique in monitoring X-ray beam profiles corresponding to a Boron Mirror diffraction sample as well as X-ray diffraction images from large area detectors.

III. Background

A. Experiment Overview

LCLS is the first hard X-ray free electron laser facility (FEL) [7] developed in history and is still pioneering the development of novel ultrafast X-ray techniques. On average, FELs are one billion times brighter than synchrotrons which makes it possible to develop novel coherent diffraction imaging techniques (CDI) [4] such as single particle imaging [14] and to adopt laser techniques to the hard X-ray regime such as X-ray photon correlation spectroscopy [13] (XPCS).

During the development and practice of these measurements, however, it is realized that the stochastic temporal and spatial profile of the hard x-ray pulses from FEL is significantly hindering the analysis of the signal measurement. For example, in ptychography, large variation of the X-ray beam profile will require more sophisticated algorithm to produce high resolution reconstruction of the sample structure. The X-ray beam profile change leads to large uncertainty in speckle contrast measurement in XPCS. While there is a continuous effort to improve the X-ray beam and its profile stability, one possible alternative solution is to classify the X-ray pulses according to their profiles.

Another important aspect of these experiments is related to the ability of the experimenter to make decisions based on actionable information readily extracted from the imaging data as it is being diffracted from streaming samples. Online image analysis techniques applied to both beam profile and sample diffraction image data would readily enable optimizing FEL and instrument parameters.

B. Matrix Sketching

Matrix sketching is a technique used in linear algebra and numerical computing to approximate large matrices with smaller, more manageable low-rank representations while preserving important properties of the original matrix [11]. The fundamental objective of matrix sketching is to represent an $n \times d$ matrix A as a sketch $k \times d$ matrix B such that $\|A^T A - B^T B\|$ is minimized. Doing so enables practitioners to use the sketch matrix in place of the original matrix in many operations, speeding up what would be costly matrix operations.

1) Sampling Methods: One prominent class of matrix sketching techniques is sampling methods: the fundamental idea behind sampling methods is to select a subset of rows or columns from the original matrix, creating a sketch that retains important characteristics while significantly reducing the matrix's size. Typically this is achieved by assigning a probability to each row of the matrix and then iterating over the rows making selections based on the probabilities [9]. This subset selection process is often guided by various considerations, such as leverage scores or spectral properties, to ensure that the sketch captures key information. In particular, priority sampling works by assigning a weight to each row given by $w_i = \|A_i\|$, and then a priority is calculated as $p_i = \frac{w_i}{u_i}$ where u_i is a random number from the uniform distribution [6].

2) Frequent Directions: A different approach to sketching is known as Frequent Directions (FD) [9]. The Frequent Directions Algorithm works in a similar way to the Frequent Items algorithm [11]: similar to how the frequent items algorithm shrinks ℓ distinct elements by the same magnitude once its buffer is full, the Frequent Directions algorithm "shrinks" ℓ orthogonal vectors by approximately the same magnitude once its buffer is full. To do so, at each step:

1) Initially, a buffer representing the matrix sketch of size $\ell \times d$ is filled with $\ell - 1$ rows of data, with the ℓ row being left zero.
2) At each iteration, row ℓ is filled with a row from the data.
3) To "shrink" the ℓ orthogonal vectors, the matrix sketch is first rotated from the left such that the rows of the buffer are orthogonal and in descending magnitude order. This is accomplished by computing the singular value decomposition $U\Sigma V^T$ and selecting V^T.
4) Then, the norm of sketch rows are shrunk so that the smallest direction (the last row) is set to 0. In practice, this is achieved by subtracting the singular values in Σ by the last entry, followed by left-multiplication of V^T.

Since computing the singular value decomposition (SVD) is very expensive, in practice, the fast frequent directions algorithm is used instead, whereby the same iteration is applied, but for a buffer of size $2\ell \times d$, and shrinking the last ℓ rows of the buffer to be zero. In this way, the SVD is only computed once every ℓ iterations.

IV. Matrix Sketching

A. Rank Adaptive Frequent Directions

1) Motivation: In online scenarios, estimating the required number of components can pose challenges for practitioners, especially when dealing with stochastic processes such as Self-Amplified Spontaneous Emission mode of the X-ray beam. Additionally, practitioners are often more concerned with ensuring the model meets a certain error threshold rather than precisely determining the number of components to retain, such as in the case of data compression applications. Furthermore, when dealing with high-resolution data, such as 2-megapixel images, across hundreds of computational cores, the memory and storage demands for the sketching process become significant. Therefore, there is a preference, when possible, to minimize the sketch size.

2) Estimating Reconstruction Error: Motivated by these demands, we propose a heuristic approach to determining and adaptively modifying the number of components at each batch of images during the sketching process based on the estimated reconstruction error of the current sample. After each rotation step, we will compute a low-memory estimation of the reconstruction error of the freshly processed sample and determine whether to increase the rank based on this estimation.

Computing the true reconstruction error of the sketch up to the most recent time would require storing all the data. To avoid this, we explored cheaper heuristics. However, even computing the reconstruction error for part of the data is non-trivial: a low-memory estimation of reconstruction error is necessary here since it is very expensive to explicitly compute the actual reconstruction error given by $\|X - UU^T X\|_F^2$ for $X = U\Sigma V^t$, a 2-megapixel image. Notice that $I - UU^T$ forms a $2M \times 2M$ matrix, which cannot be easily stored in memory. It has been shown that an efficient estimation of the Frobenius Norm can be computed via random matrix multiplication [2]. We will use this estimation of the Frobenius Norm for simplicity, but other more effective estimates are currently being studied, such as Stochastic Trace Estimation in [6] or GKL Estimator in [10]. These improvements to the algorithm could be studied in future work and have the potential to significantly improve runtime and error rates for rank adaptivity.

Using this fact, we come up with the following heuristic in Algorithm 1, where ν is an arbitrary constant representing the number of random matrix multiplications (in practice, we have found a decrease in error at roughly 10% for every 10 multiplications), X is d features by n samples, and ϵ is a user-defined error threshold. In theory, this heuristic should guide the algorithm into increasing the rank if the current most prominent features of the beam (which is represented by the most recent set of samples) are not being captured in the matrix sketch.

Algorithm 1 Rank Adaption Heuristic

Input: X, U, ν, ϵ

Output: $RankAdapt$ (Heuristic indicating whether or not to increase the rank)

1: **function** RankAdaptHeur(X, U, ν)
2: $\quad Avg \leftarrow 0$
3: $\quad$ **for** $k \leftarrow 1$ to ν **do**
4: $\quad\quad V \leftarrow Generate\_Random\_Gauss\_Matrix()$
5: $\quad\quad V \leftarrow X \times V$
6: $\quad\quad \widehat{V} \leftarrow U^T \times V$
7: $\quad\quad \widehat{V} \leftarrow U \times \widehat{V}$
8: $\quad\quad Avg \leftarrow Avg + ||\widehat{V} - V||_2^2$
9: $\quad RankAdapt \leftarrow \frac{n}{\nu} \cdot Avg < \epsilon$
10: $\quad$ **return** $RankAdapt$

3) Applying the Heuristic to Frequent Directions: Combining the heuristic with the frequent directions algorithm, we come up with the Rank Adaptive Frequent Directions Algorithm 2. We start by filling the buffer with data. Then we iterate for k from l to n. At each step, we check if the buffer is full by checking if $nextZeroRow$ is at $2l$, and if there are also sufficient rows left to process. If in addition we have previously determined that the rank must be increased, then we increase the rank. Instead, if there are too few rows left to process or the reconstruction error is low, we proceed with the FD algorithm as usual. When the buffer is full, the reconstruction error is computed to determine if the rank should be adapted in the next cycle. It should be noted that on line 17, the value under the square root will never be negative, due to the indicator I_ℓ, where it is assumed that the diagonal values beyond the first ℓ-th terms are zero. At the end of each iteration, we add new data to our buffer and perform bookkeeping. Note that this heuristic should not increase the overall runtime of the algorithm by a significant margin, since the SVD is already computed during the rotation step. In addition, one must ensure that there are no zero-row entries in the sketch between rotations (especially before merging sketches during the parallelization step discussed in the upcoming section) to ensure sketching accuracy. Therefore, care should be taken during the rank adaption step to ensure that a sufficient number of images are left in the batch before increasing the size of the sketch.

B. Chaining Priority Sampling with Frequent Directions

Although Frequent Directions provides excellent theoretical and empirical error bounds, its runtime lags behind competitors such as sampling methods and random-projection methods [5]. Therefore, in order to accelerate Frequent Directions, while still attaining strong error bounds, we propose chaining the faster method of Priority Sampling, described in [6], with Frequent Directions, in order to select which data points in

Algorithm 2 Rank Adaptive Frequent Directions

Input: X, ν, ϵ, ℓ
Output: $Sketch$ (matrix sketch of data)

function RANKADAPTFD(X, ν, ϵ)
  Buffer $\leftarrow 0^{2\ell \times d}$
  $rowsLeft \leftarrow n$
  Buffer$[0:\ell] \leftarrow X[0:\ell]$
  $nextZeroRow \leftarrow \ell, increaseEll \leftarrow False$
  for $k \leftarrow \ell$ to n **do**
    if $nextZeroRow == 2\ell$ **then**
      $canRankAdapt \leftarrow rowsLeft > \ell + \nu$
      if $increaseEll \wedge canRankAdapt$ **then**
        $\ell \leftarrow \ell + \nu$
        Buffer $\leftarrow$ [Buffer, $0^{2\nu \times d}$]
        $increaseEll \leftarrow False$
      else
        Buffer$_\ell \leftarrow X[k,:]$
        $[U, \Sigma, V] \leftarrow svd$(Buffer)
        $\delta \leftarrow \sigma_\ell^2$
        Buffer $\leftarrow \sqrt{\Sigma^2 - \delta I_\ell} \cdot V^T$
        $nextZeroRow \leftarrow \ell$
        if $canRankAdapt$ **then**
          $RE \leftarrow RE(X[k-\ell:k])$
          if $RE > \epsilon$ **then**
            $increaseEll \leftarrow True$
    Buffer$[nextZeroRow] \leftarrow X[k]$
    $nextZeroRow \leftarrow nextZeroRow + 1$
    $rowsLeft \leftarrow rowsLeft - 1$
  return $Sketch$

each batch are the most important to the frequent directions sketch. The main idea is to bring down the number of samples by a significant fraction using sampling methods, such as 80%, but not down to a low-dimensional latent space such as 50 components, as one would sacrifice too much accuracy for speed and memory in this case. This addition to the sketching process will be especially important in extreme data generation scenarios, such as in the case of LCLS-II generating 1 million 16-megapixel images per second. We now write the full Accelerated Rank Adaptive Matrix Sketching (ARAMS) in Algorithm 3.

Algorithm 3 Accelerated Rank Adaptive Matrix Sketching (ARAMS)

Input: X, β, ϵ, ℓ
Output: $Sketch$ (matrix sketch of data)

function ARAMS(X, ν, β, ϵ)
  $PQ \leftarrow PriorityQueue(\beta n)$
  for $k \leftarrow 1$ to n **do**
    $PQ.push(X[k])$
  $Sketch \leftarrow RankAdaptFD(PQ, \nu, \epsilon)$
  return $Sketch$

C. Parallelization

One of the key properties of the Frequent Directions Algorithm is that the resulting sketch forms a mergeable summary: suppose we have two sketches of disjoint datasets of equal size, called data A_1 with sketch B_1, and data A_2 with sketch B_2, where A_1 and A_2 form the full dataset $A = [A_1; A_2]$. It can be shown that to create a single summary B which approximates A using only B_1 and B_2, one can simply run the Frequent Directions algorithm on the matrix $B' = [B_1; B_2]$ of size 2ℓ to generate a new sketch B of size ℓ, which preserves the same theoretical space/error tradeoff as each B_i with respect to A_i [9].

In many use cases, however, sequential merging of matrix sketches turns out to be the main bottleneck. For example, in the beam profile monitoring scenario, a global matrix sketch may be desired after only a dozen rotation operations, across hundreds of cores in parallel. Sequential merging to a single core would cost hundreds of rotation operations, increasing the run time by an order of magnitude.

We propose a tree-merge parallelization scheme in which we successively merge sketches summarizing data subsets of equivalent magnitude. The resulting sketch after each merge step summarizes data subsets of identical sizes, and as this property is invariable across the merge steps, the global sketch enjoys the same space/error trade-off as the local sketches. We formalize this in the appendix. Furthermore, since each step reduces the number of sketches by an order of magnitude, the merging step performs a logarithmic number of rotations.

V. BENCHMARK PERFORMANCE: ABLATION STUDY

We now compare the rank adaptive accelerated sketching algorithm on synthetic data across varying parameters to understand the performance of this algorithm and its scaling properties.

1) Synthetic Data: Each of the three synthetic data sets is an $n = 15000 \times d = 1000$ random matrix, with varying singular value decay rates. The datasets are generated by first generating an $n \times r$ and an $r \times m$ orthogonal matrix, where r represents the intended rank of the matrix. Random orthogonal matrices with normal distribution can be generated via a number of matrix decompositions as described in [8], such as via QR decomposition of a standard normal distribution matrix. If we are generating data over multiple cores, each core starts with the same random orthogonal matrices and we then perturb these random orthogonal matrices by a unique perturbation for each core. This is supposed to simulate how we would expect similar but not identical data to look like in beam profile analysis. We then generate the vector of singular values, scaling according to the desired weights of each component. In our case, we used sub-exponential, exponential, and super-exponential decaying sets of singular values. The matrix is then assembled in the same way as an SVD. The top-left panel of figure 1 plots the singular values of the resulting matrices, with the legend indicating the function used to generate the singular values.

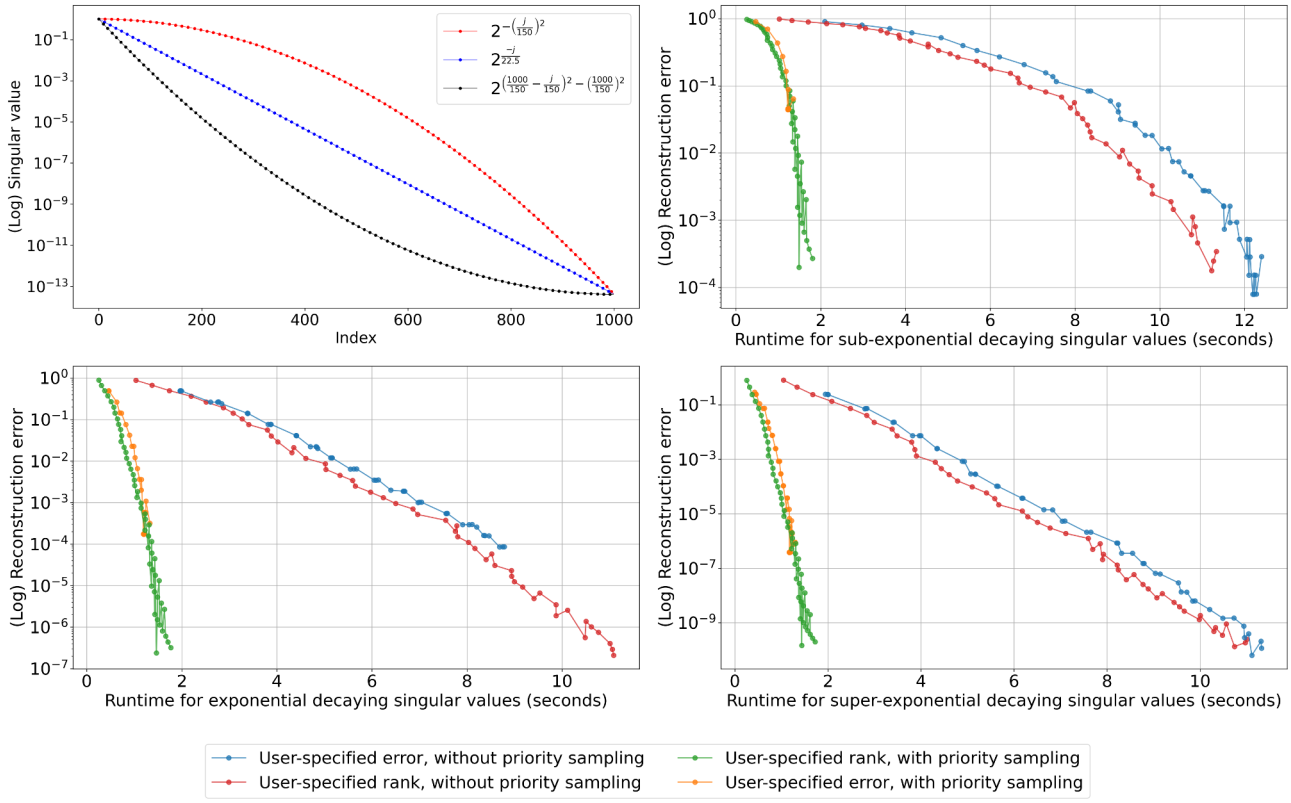

Fig. 1. The upper-left panel plots the singular values of each synthetic dataset: the black plots super-exponential, the blue plots exponential, and the red plots sub-exponential singular value decay. The remaining 3 panels are semilogy plots of reconstruction error versus runtime for each of the datasets. Four variants of the algorithm are tested: rank-adaptive versus non-rank adaptive frequent directions (user-specified error versus user-specified rank), and with versus without priority sampling.

2) Impact of Rank Adaptive and Priority Sampling Modifications: We test four variants of the Frequent Direction Algorithm on a single core: with/without Rank Adaptivity and with/without Priority Sampling. It should be noted that when Rank Adaptivity is enabled, the number of components dynamically increases to match the user-specified minimum error tolerance as the algorithm runs, whereas when Rank Adaptivity is not enabled, the number of rank stays fixed throughout the runtime of the algorithm. We shall denote the former as "User-Specified Error" and the latter as "User-Specified Rank". In this experiment, we vary the minimum-error tolerance in the rank-adaptive case or the rank of the sketch in the non-rank-adaptive case from 0–500, and record the corresponding runtime and error.

In figure 1, we plot the reconstruction error versus runtime of the four variants of the FD algorithm. We observe three broad trends in this experiment. First, we notice significant improvements in runtime and in the time/error trade-off by using the priority sampling variants of the FD algorithm. This is to be expected, since we are processing significantly less data in these variants, and therefore would expect an improvement in runtime due to lower amounts of computation. In addition, since we prioritize "higher-quality" data points via the priority sampling step, one would also expect the error/time tradeoff to be improved.

The second trend we observe is that the normal (User-Specified Error) and rank adaptive (User-Specified Rank) variants of the FD algorithm track each other quite closely (with maximum difference of 2 seconds in runtime for the most part), with the rank-adaptive variant performing only slightly worse in most cases. The drop in performance is even less noticeable in the priority sampling variants of the algorithm. Importantly, even though the user is specifying the error rather than the rank, the performance of the matrix sketching does not suffer too much in both the case where one uses priority sampling and where one does not, and thus the rank-adaptive variant may be useful in cases where the rank cannot be easily ascertained.

The third trend we observe is across the different decay rates of singular values. We observe that the slower the decay of the singular values, the worse the performance of the rank-adaptive methods. Specifically, we see the error/time tradeoff between the user-specified error and rank for non-PS variants are closest in the super-exponential case, with gradual improvement from sub-exponential to exponential and from exponential to super-exponential. This may be due to the fact

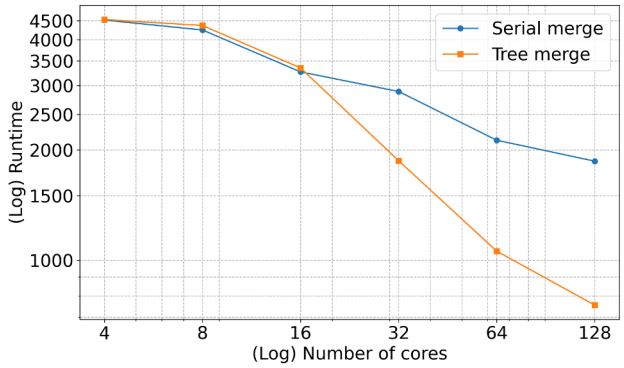

Fig. 2. Log log efficiency plot. The parallel variant of the algorithm demonstrates clear benefits compared to the serial variant.

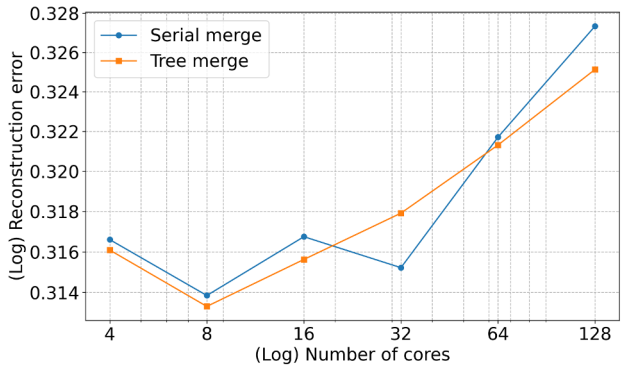

Fig. 3. Log log plot of error versus number of cores. The error of the parallel variant of the algorithm tracks the serial variant's error closely.

that for matrices with singular values more evenly dispersed (as in the sub-exponential decay), it becomes more difficult to estimate the reconstruction error due to a broader range of data, and the algorithm therefore tends to over or undershoot the correct rank. We also notice that these differences are greatly reduced in the priority sampling cases, which may be explained by the improved efficiency of the heuristic and lower data runtimes.

3) Parallelization: We now investigate the strong scaling properties of the accelerated matrix sketching algorithm.

In this experiment, we use a matrix of size 2000×1658880 with cubically decaying singular values generated using the technique described above and run the vanilla FD algorithm with sketch size 200 with the proposed parallelization framework, comparing the FD algorithm using our proposed tree-merge versus using a standard serial-merge of the sketches across varying number of cores. As we can see in figure 2, there is a roughly linear correlation between the number of cores and the runtime of the tree-merge variant. This result makes sense due to the fact that we perform very few (in fact, a logarithmic number of) MPI communications and SVD calculations (at the very least, 10 times fewer SVD calculations in the 128 core experiment). On the other hand, the serial-merge variant begins to plateau at just 16 cores. From these figures, it is clear that merging in a branching fashion is crucial for the scalability of our sketching system. More concretely, the LCLS-II upgrade is expected to increase the rate of data generation from 120 pulses per second to over 1 million pulses per second. In order to keep up with this rate of production, such a factor of reduction is required for online image analysis. Furthermore, we notice in Fig.3 that the error of the tree-merge variant closely tracks the error of the serial-merge variant, confirming our expectation that the theoretical error and space guarantees are maintained in the parallel implementation. This suggests that as we increase the number of cores to handle increasingly large volumes of data, we would not expect our error rates to significantly increase.

VI. Image Monitoring

We now present the full methodology for image monitoring, and more generally, a general purpose LCLS data exploration pipeline illustrated in Fig.4. For beam profile analysis, we performed a variety of image processing techniques such as thresholding by intensity, intensity normalization, and centering to ensure that the primary shape of the beam profile and its distribution of intensity were the focus of the analysis. We then produced a matrix sketch of the processed beam profiles using the previously described accelerated rank adaptive matrix sketching algorithm and projected the data onto this lower dimensional latent space. Likewise, we applied the accelerated rank adaptive matrix sketching algorithm to calibrated large area detector images, namely 2 megapixel detector images at LCLS, followed by latent space projection.

Once the data has been projected, we may apply the popular Uniform Manifold Approximation and Projection (UMAP) algorithm [12] to further reduce the dimension of each data point to 2-d for visualization purposes. Although UMAP is well known for its scalability, it is not suitable for directly analyzing extremely high-dimensional data due to the curse of dimensionality. Furthermore, UMAP would be far too slow to process such high dimensional vectors in real-time, and thus would not be suitable for real-time beam profile monitoring. On the other hand, PCA is a simple linear dimension reduction method, and cannot capture the intricacies of complex data sources effectively. Thus, both stages of the procedure are necessary steps for efficiency and accuracy.

In 2-dimensional space, we may further analyze the data by clustering this latent embedding or by identifying outliers using anomaly detection techniques. This can be achieved for example via the Ordering Points to Identify the Clustering Structure (OPTICS) algorithm in the former case [1], or fast Angle-Based-Outlier-Detection methods in the latter case.

A. Experimental Results

1) Beam Profile Data: In figure 5, we provide the results of applying our technique to beam profile data. Our interpretation is that the data is separated on the Y-axis by where the center of mass is: points close to $y = 0$ correspond to beam profiles which are symmetric along the y-axis, and points to the right of

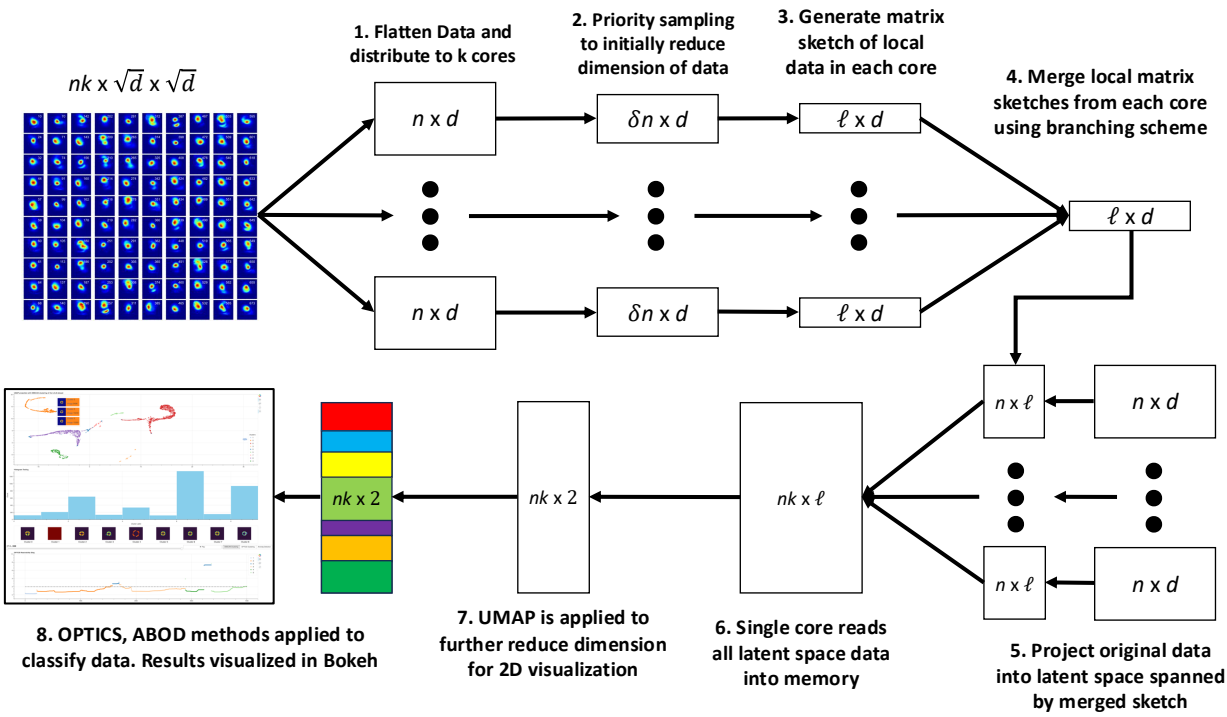

Fig. 4. Schematic depicting the full data processing pipeline. The runs initially arrive as large batches of images. We produce a matrix sketch from this data in each processing core, and merge these sketches to produce a summary sketch. We then apply principal component analysis to project the original data to a low dimensional space, followed by UMAP and clustering/anomaly detection methods for visualization purposes.

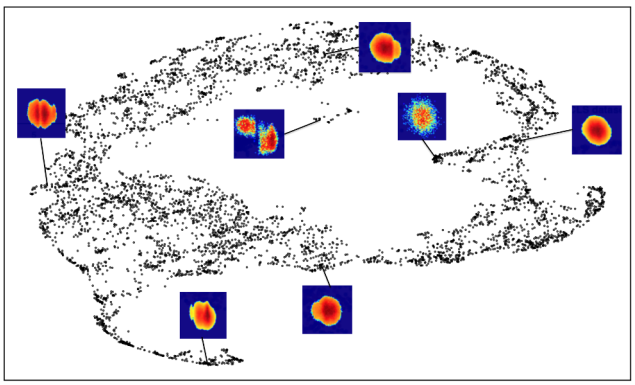

Fig. 5. Results of the latent space embedding on beam profile data. In this run, the unsupervised algorithm places beam profiles based on weight to the left or right on the X-axis and circularity on the Y-axis.

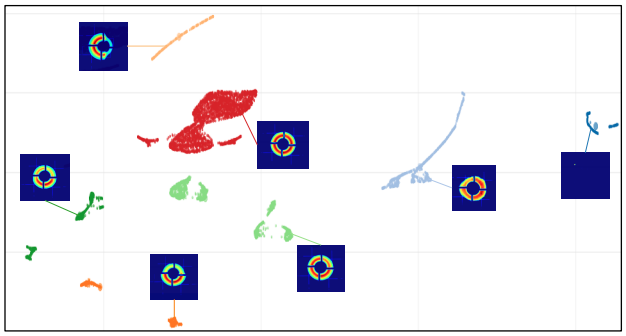

Fig. 6. Results of the latent space embedding on diffraction data. In this run, the unsupervised algorithm separates the data into clusters based on their shape. Roughly speaking, the clusters differ from one another based on the weight in each quadrant of the ring.

$y = 0$ correspond to beam profiles which have higher weight on the left side of the speckle pattern (similarly for the left region of the embedding having higher weight on the right side). On the X-axis, the data is distributed by density of mass: points that are close to $x = 0$ are highly circular, while points that are far from $x = 0$ are elongated or have multiple lobes. We would like to stress that as an unsupervised algorithm, we did not select for these features in the data, but rather that the data naturally separated itself in this way. Additionally, exotic beam profiles which deviate heavily from zero-order mode beam profiles separate themselves readily in such analyses, as these outlier shapes do not match primary features of the other beam profiles. Combined with the corresponding electron beam parameters for each beam profile, such information can provide highly valuable feedback for beam operators to modify the current parameter settings as an experiment proceeds.

2) Diffraction Data: In figure 6, we provide the results of applying our technique to diffraction data, which is a

significant departure from the relatively simple Gaussian-like beam profiles. Applying our technique, the data separates into clear clusters, and we can analyze the distribution data as the experiment is being conducted. These results demonstrate our method generalizes to a broader class of data rather than being restricted to simply beam profile data. Notably, our technique does not use any prior knowledge of the experiment beforehand and is unsupervised.

B. Runtime

To demonstrate the effectiveness of this framework, we ran our framework on a full run of an LCLS XPCS experiment, comprising of $12,0000$ 2-megapixel images. After cropping, we processed the full data set at 136Hz using 64 cores, and the UMAP/OPTICS visualization was produced in less than a minute. This method of beam profile monitoring exceeded the current rate of data generation and gives practitioners the ability to process beam profile and experimental data in real time which is crucial for the usability of this analysis.

VII. Conclusion

In conclusion, we propose a new method for monitoring beam profile and general data exploration for X-ray image data. In this method, we propose a new matrix sketching algorithm and parallelization scheme with strong scaling properties and attractive accuracy-speed trade-off. We demonstrate the effectiveness of this method by evaluating the beam profile of a recent LCLS experiment in real-time speed and find that our technique generalizes to experimental data beyond beam profiles, such as X-ray diffraction data.

Appendix

A. Proof of Correctness of Parallelization Scheme

A sketch is considered a mergeable summary if given data set A_1, A_2 with corresponding summaries B_1, B_2, one can create a single summary B of $[A_1; A_2]$ which achieves the same formal space/error tradeoff as each of B_1, B_2 to A_1, A_2, respectively. In [9], it is proved that the Frequent Directions algorithm produces mergeable summaries. In our merging strategy, we apply the same technique as discussed in [9], though in a branching fashion, thereby achieving the same formal space/error trade-off, since at each step in the merging tree, the space/error trade-off is maintained.

We will prove this formally by induction. Assume without loss of generality that we have $2^n - 1$ cores, and in each core we have data A_k with corresponding sketch B_k, summarizing the data. Our induction hypothesis will be: For $A_0, ..., A_n$ data sets with summaries $B_0, ..., B_n$ where $n \in \{a^k, k = 0, 1, ...\}$ where $a \in \mathbb{Z}$, the branching merging process generates a merged summary B which maintains the same theoretical error/space guarantees. As discussed earlier, the base case of $k = 1$ is covered in [9]. Proceeding with the induction, we would like to show that for for datasets $A_0, ..., A_{a^k}$ and summaries $B_0, ..., B_{a^k}$, we may form a summary B with the same error/space guarantees. Take any arbitrary partition of the $B_0, ..., B_{a^k}$ into a sets. Each partition is a set of a^{k-1} summaries. By the induction hypothesis for $k - 1$ to each of the a partitions, we may form a new summaries (call B'_j the summary of partition j), each maintaining the same error/space guarantees of their predecessors. With these new summaries $B'_0, ..., B'_a$, we may apply base case of the induction hypothesis once more to make the induction conclusion for k.

Acknowledgment

We thank Jana Thayer for building the LCLS data systems and creating the environment for this project. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Award Number FWP-101101. 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.

References

[1] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, page 49–60, New York, NY, USA, 1999. Association for Computing Machinery.

[2] Zvonimir Bujanovic and Daniel Kressner. Norm and trace estimation with random rank-one vectors. *SIAM Journal on Matrix Analysis and Applications*, 42(1):202–223, 2021.

[3] Stanford Linear Accelerator Center. LCLS overview, 2023.

[4] Henry N Chapman, Anton Barty, Michael J Bogan, Sébastien Boutet, Matthias Frank, Stefan P Hau-Riege, et al. Femtosecond diffractive imaging with a soft-x-ray free-electron laser. *Nature Physics*, 2(12):839–843, 2006.

[5] Amey Desai, Mina Ghashami, and Jeff M. Phillips. Improved practical matrix sketching with guarantees. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1678–1690, 2016.

[6] Nick Duffield, Carsten Lund, and Mikkel Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54(6):32–es, 2007.

[7] Paul Emma, R Akre, J Arthur, R Bionta, C Bostedt, J Bozek, A Brachmann, P Bucksbaum, Ryan Coffee, F-J Decker, et al. First lasing and operation of an Ångstrom-wavelength free-electron laser. *Nature Photonics*, 4(9):641–647, 2010.

[8] Alan Genz. Methods for generating random orthogonal matrices. In Harald Niederreiter and Jerome Spanier, editors, *Monte-Carlo and Quasi-Monte Carlo Methods 1998*, pages 199–213, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[9] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016.

[10] Serge Gratton and David Titley-Peloquin. Improved bounds for small-sample estimation. *SIAM Journal on Matrix Analysis and Applications*, 39(2):922–931, 2018.

[11] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 581–588, New York, NY, USA, 2013. Association for Computing Machinery.

[12] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[13] Wojciech Roseker, SO Hruszkewycz, F Lehmkühler, M Walther, H Schulte-Schrepping, S Lee, T Osaka, L Strüder, R Hartmann, M Sikorski, et al. Towards ultrafast dynamics with split-pulse x-ray photon correlation spectroscopy at free electron laser sources. *Nature communications*, 9(1):1704, 2018.

[14] Zhibin Sun, Jiadong Fan, Haoyuan Li, and Huaidong Jiang. Current status of single particle imaging with x-ray lasers. *Applied Sciences*, 8(1):132, 2018.

Appendix: Artifact Description Artifact Evaluation

RTIF CT IDENTIFIC TION

Our contributions are three-fold. In contribution C1, we describe a two- stage rank-adaptive online matrix sketching algorithm to monitor LCLS imaging datasets, combining the priority sampling and frequent directions matrix sketching algorithms to balance run-time with error. This algorithm finds a low-rank approximation of data. To demonstrate the effectiveness of our algorithm, we generated Figure 1 in our paper depicting the error/runtime tradeoff of our algorithm using artifact A1. Artifact A1 is the testMe.py program which contains the modified frequent directions algorithm (as described in our paper) and was run using Artifact A2, one core of the SLAC S3DF Milano computing cluster. A2 runs on an AMD EPYC Milan 7713 , with 120 cores and 480GB per node. The linux distribution is given by Linux sdfiana004 4.18.0-372.32.1.el8_6.x86_64 #1 SMP Fri Oct 7 12:35:10 EDT 2022 x86_64 x86_64 x86_64 GNU/Linux + lscpu and the architecture is x86_64. We use a synthetic dataset generated by Artifact A3, the genData.py file, which produces random matrices with specific singular values as described in our paper and seen in Figure 1. We used the computing environment artifact A4, johnw-ana-4.0.48-py3 : a conda environment containing Matplotlib, UMAP, HDBSCAN, Numpy, Scipy, Sklearn, Bokeh, Seaborn, and HeapQ packages. The batch job was submitted to the S3DF computing cluster A2 using artifact A5: the slurm script script.sh . Once artifact A1 (the testMe.py program) is completed, it produces numpy arrays containing the error and times for each of the parameters varied in the test. We run artifact A6, the plotMe.py file, to generate the actual plots of Figure 1 using matplotlib.

Artifact A12, freqDir.py houses all the code necessary for the parallel and real-data analysis (C2 and C3). It houses the modified frequent directions algorithm, data grabbing and pre-processing step, and the code for visualization. We provide a standalone version of this code, but the original version requires access to the data and a SLAC account, due to the use of the BTX library and specialized SLAC libraries. Therefore, it is not possible to run this code in the exact same way as the authors without being a SLAC user.

Second, in contribution C2, we present a scheme for parallelization of this algorithm and provide strong scaling studies for this scheme. To our knowledge, our proposed modifications of the frequent directions algorithm in acceleration via priority sampling, applying rank adaptation, and parallelization in this branching fashion have not been studied. To this end, we submit a batch job via artifact A7, the script parallelscript.sh , which runs artifact A8, the file parallelrun.py . Artifact A8 simply runs the matrix sketching algorithm across multiple cores and saves the error, runtime, and number of cores to a csv, to which we run artifact A9, visMe.py to generate the plots, figures 2 and 3, which are the main empirical results of this scheme. We use the same code from artifact A3 again to generate the synthetic dataset and run this test on artifact A2.

Third, in contribution C3, using this low rank approximation to project the original data into latent space, we propose a technique for monitoring imaging datasets in a scalable and efficient manner by visualizing and clustering the images using UMAP and OPTICS clustering. We demonstrate the effectiveness of this technique in monitoring X-ray beam profiles corresponding to a Boron Mirror diffraction sample as well as X-ray diffraction images from large area detectors. Artifacts A10 and A11 correspond to the Boron Mirror diffraction sample and the X-ray diffraction images from large area detectors. These datasets are not publicly available, since they are private datasets attained via experimentalist collaborators with the LCLS experiment at SLAC. They could potentially be made available upon request. This full data analysis pipeline is run using artifacts A13, the run.py script, which contains the script necessary for using the code in artifact A12 such as setting up the python FrequentDirections sketching object, and the slurm script fullrunscript.sh to submit the batch-job. This code generates the embedding and visualizations seen in figures 5 and 6, such as a Bokeh HTML file. We run this code on artifact A2, the S3DF Milano compute cluster.

The files and script artifacts described above are available in the following public github repo: https://github.com/johnwinnicki/ARAMS_Submission

REPRODUCIBILITY OF EXPERIMENTS

Other than when noted, the code is hard-coded in the github repo to have the correct parameter settings to reproduce these results, and so no inputs nor input files are required beyond simply running the python files or submitting the slurm scripts.

There are three experiments: one produces figure 1 towards C1, one produces figures 2 and 3 towards C2, one produces figures 5 and 6 towards C3.

In order to produce figure 1, one activates the conda environment A4 and generates a 15000 x 1000 matrix using artifact A3. This should take less than 5 minutes. Using the outputted matrix, one submits batch job launching A1 with varying parameters, using slurm script A5. This should take roughly 2-3 hours, since it re-runs the same analysis 10 times over multiple parameter combination. The output is plotted using A6. All of these scripts should be in the same directory and don't need a special command or input to run (it suffices to simply submit the batch job or run the python script). This should take roughly 10 minutes (mostly due to the computation of the singular values). The expected results and evaluation of them is figure 1 as seen in the paper, with the same observations as discussed in the paper.

In order to produce figures 2 and 3, one simply submits a batch job A8 to run file A7, which should be in the same directory as A8. It calls python file A12. This generates a CSV file, which can be plotted using artifact A9. This should take roughly 2-3 hours to run to completion, but can be made to run faster using settings different than what have been pre-set. This should produce figures 2 and 3 as seen in the paper, with the same observations as discussed in the paper.

To produce figures 5 and 6, one would need access to artifacts A10 and A11. Assuming one has access to those, via a SLAC account, one could call artifact A13 with the parameters changed in

the python file: exp = 'xppc00121', run = 510, det_type = 'XppEndstation.0:Alvium.1' in order to generate figure 5. To generate figure 6, one would change in the Article A13 the parameters to be: exp = 'xpplx9221', run = 244, det_type = 'XppEndstation.0:Alvium.1'. Using 64 cores, this should be submitted to the compute cluster using artifact A13. The output should be html files re-creating the figures 5 and 6 with similar properties as discussed in the paper (for beam profiles one expects a homogeneous embedding with circularity along one axis; for diffraction data one expects clear clustering). The html files should be interactive with hover tooltip functionality. The estimated time of execution is less than 5 minutes.