doi:10.1017/S1431927621000829
Microsc. Microanal. 27 (Suppl 1), 2021

Open-Source Tools and Containers for the Production of Large-Scale S/TEM Datasets

Alexander M Rakowski[1], Joydeep Munshi[2], Benjamin Savitzky[3], Shreyas Cholia[1], Matthew L Henderson[1], Maria KY Chan[2] and Colin Ophus[3]

[1]LBNL, United States, [2]ANL, United States, [3]Lawrence Berkeley National Laboratory, California, United States

Four-dimensional scanning transmission electron microscopy (4D-STEM) has expanded the types of information which can be extracted from materials. However, 4D-STEM experiments also produce large datasets spanning GB to TB, which are often infeasible to examine manually. Consequently, there have been extensive efforts (e.g., Py4DSTEM [1]) for automating the analysis of these datasets. Currently, these methods rely on classical computer vision techniques (e.g., Fourier filtering and cross-correlation), which typically require significant manual tuning for each experiment. Parameter tuning is a significant roadblock towards the goal of robust and automated analysis of 4D-STEM datasets. Machine learning algorithms, notably deep neural networks (DNNs), have been demonstrated to outperform classical techniques in most computer vision tasks and are well suited for automated analysis [2]. While much of the utility of modern DNNs is attributable to increased compute performance and the evolution of open-source high-level software interfaces, the development of availability of large-high-quality datasets has been of pivotal importance. To train high the performant DNNs requires creating high-quality datasets, analogous to the role of specimen preparation in recording electron microscopy images.

We present an all python, portable, scalable and efficient method for generating large datasets suitable for training DNNs (Figure 1). We show that the process is platform agnostic and scalable by producing examples on a single local machine and a high-performance computing (HPC) cluster with GPU acceleration. We also discuss the importance of metadata labeling and architectural decisions that maximize the data's reusability and the framework's flexibility and portability.

To exemplify the data generation workflow, we produced a dataset suitable for detecting and localizing the Bragg scattering, a notoriously difficult but crucial task in analyzing crystalline materials. The generated dataset comprises convergent beam electron diffraction (CBED) patterns, projected potentials (V_g), and out-of-plane tilt (q_z). The atomic potentials and CBED images were calculated using PyPrismatic, a python wrapper for the Prismatic S/TEM [3,4] image simulation software. Following which, the projected potentials and out-of-plane tilts were generated through bespoke python scripts. Finally, these datasets were densely labeled with metadata and stored in a HDF5 file format. The careful semantic labeling maximizes the utility and reusability of the data for other projects, addressing "dark data" issues [5]. Crucially, while the workflow is python-based, we minimize platform and device architecture considerations by containerizing the process (Figure 2). This approach allows the workflow to be easily shared and any work replicated, with minimal software and hardware compatibility concerns. Furthermore, containers offer many other compelling benefits, including rapid prototyping and the ability to run multiple containers in an independent or linked mode simultaneously.

The described image generation workflow greatly simplifies generating large datasets, which are vital for training DNNs required for the automated and robust analysis of 4D-STEM datasets. Such networks offer great promise and could ultimately, for example, perform strain and orientation mapping of materials with currently inaccessibly large fields of view, classification of unknown polycrystalline structures, and accurate specimen thickness estimations. Furthermore, the work demonstrates the benefit of containerized solutions for distributing software to the scientific community.

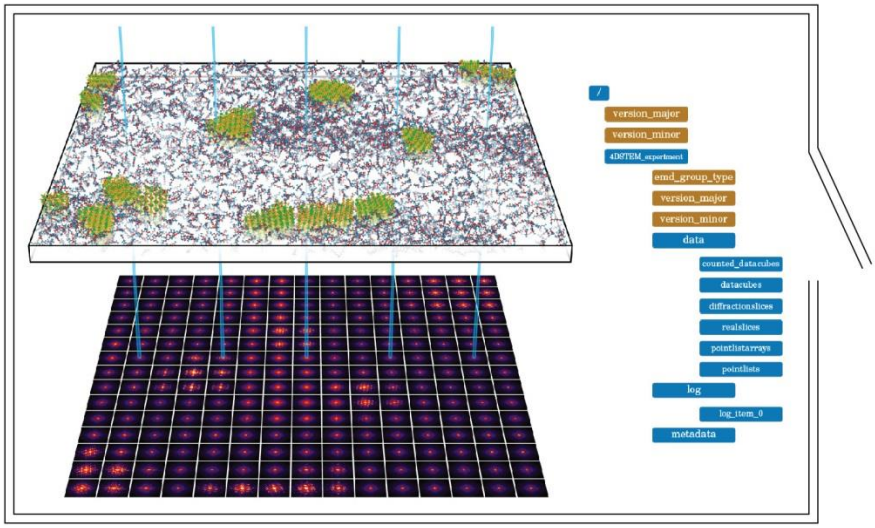

Figure 1. Schematic representation of the containerized 4D-STEM image simulation process, and the densely labeled hdf5 output file.

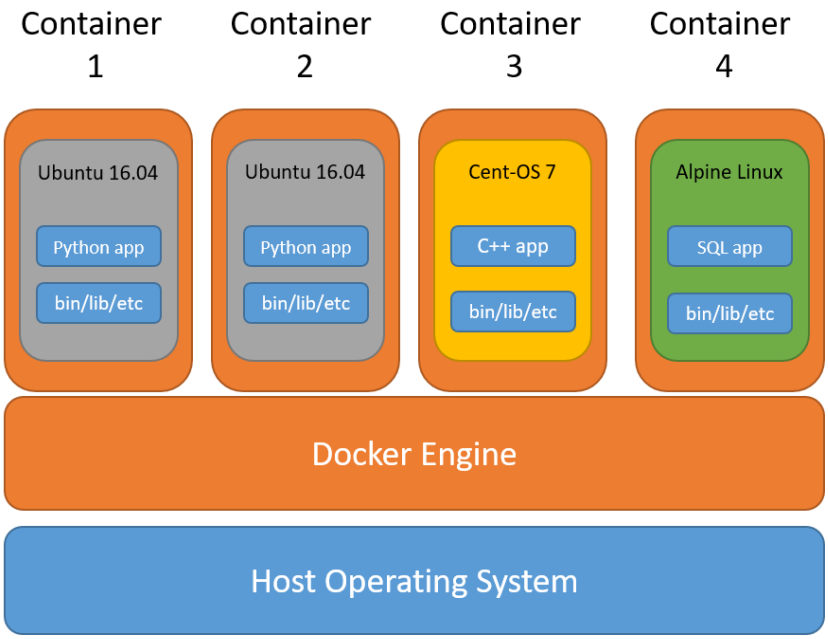

Figure 2. Schematic of the benefits of the containerizing processes. The example shows two instances of a python app (container 1 and 2), and a single instance of a C++ and SQL app while being agnostic to the host system operating system, utilizing the Docker Engine.

References

1. Savitzky, B.H., et al., py4DSTEM: A software package for multimodal analysis of four-dimensional scanning transmission electron microscopy datasets. arXiv preprint arXiv:2003.09523, 2020.
2. Guo, Yanming, et al. "Deep learning for visual understanding: A review." *Neurocomputing* 187 (2016): 27-48.
3. Ophus, C., A fast image simulation algorithm for scanning transmission electron microscopy. Advanced Structural and Chemical Imaging, 2017. **3**(1): p. 13.
4. Pryor, A., C. Ophus, and J. Miao, A streaming multi-GPU implementation of image simulation algorithms for scanning transmission electron microscopy. Advanced Structural and Chemical Imaging, 2017. **3**(1): p. 15.
5. Schembera, B. Like a rainbow in the dark: metadata annotation for HPC applications in the age of dark data. J Supercomput (2021). https://doi.org/10.1007/s11227-020-03602-6

https://doi.org/10.1017/S1431927621000829 Published online by Cambridge University Press