167-PFlops Deep Learning for Electron Microscopy: From Learning Physics to Atomic Manipulation

Robert M. Patton[1], J. Travis Johnston[1], Steven R. Young[1], Catherine D. Schuman[1],
Don D. March[2], Thomas E. Potok[1], Derek C. Rose[3], Seung-Hwan Lim[1], Thomas P. Karnowski[3],
Maxim A. Ziatdinov[4,5], and Sergei V. Kalinin[4,5]
Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831-6085

***Abstract*—An artificial intelligence system called MENNDL, which used 25,200 NVIDIA Volta GPUs on Oak Ridge National Laboratory's Summit machine, automatically designed an optimal deep learning network in order to extract structural information from raw atomic-resolution microscopy data. In a few hours, MENNDL creates and evaluates millions of networks using a scalable, parallel, asynchronous genetic algorithm augmented with a support vector machine to automatically find a superior deep learning network topology and hyper-parameter set than a human expert can find in months. For the application of electron microscopy, the system furthers the goal of improving our understanding of the electron-beam-matter interactions and real-time image-based feedback, which enables a huge step beyond human capacity towards nanofabricating materials automatically. MENNDL has been scaled to the 4,200 available nodes of Summit achieving a measured 152.5 PFlops, with an estimated sustained performance of 167 PFlops when the entire machine is available.**

***Index Terms*—machine learning, evolutionary computation, high performance computing**

I. Justification for ACM Gordon Bell Prize

167-PFlops projected on Summit with 152.5-PFlops measured on available 4,200 nodes to highlight structural defects in raw microscopy data using an asynchronous genetic algorithm to find the optimal deep learning network topology and hyperparameters. First known approach toward automated identification of atomic-scale structural irregularities in dynamic electron microscopy data.

II. Performance Attributes

Performance Attributes	Content
Category of achievement	Peak performance - 152.5-PFlops measured, 167-PFlops projected
Type of method used	N/A (Deep Learning)
Results reported on basis of	Whole application including I/O
Precision reported	Mixed single-half
System scale	Measured on 4,200 available nodes out of 4,600
Measurement mechanisms	FLOP count
Number of networks evaluated	Approximately 525,000 / hour
Training repetitions (forwards and backwards passes in a network)	424 million / second
Scanning Transmission Electron Microscopy (STEM) image analysis	First automated raw image-based atomic defect analysis

III. Overview of the Problem

The ability to control matter at the atomic scale holds great promise for future breakthroughs in virtually every area of our lives [1]. Recently, it has been realized that scanning transmission electron microscopy (STEM) allows not only visualizing materials structure at the atomic scale, but that it also can be used for atomic-level modification of matter [2]–[5]. This breakthrough emerged on top of other STEM-enabled breakthroughs in the last several years, including the capability to map structural distortions underpinning ferroic and quantum physics, map chemical reactions atom by atom, and explore internal electric fields at the atomic level. However, given tens of thousands of STEM platforms worldwide, each producing hundreds of images per microscope per day and often terabytes of ptychographic experimental data, the analysis becomes the issue due to the sheer volume of images. Consequently, real-time data analytics of these images is particularly important in the context of the electron beam manipulation of individual atoms, where real-time image-based feedback is a necessary condition to enable atom by atom fabrication, which is the ultimate goal of nanotechnology [6].

The main bottleneck to achieving this goal is that despite large volumes of images routinely generated in STEM experiments worldwide (datasets in gigabyte to terabyte range per single experiment), presently there are *no* image analytics tools for rapid and automated extraction of structural information from raw experimental images that are characterized by high levels of noise, missing structural elements, etc. Indeed, most of the methods available to date for analysis of atomically resolved data are slow, inefficient, require frequent manual inputs (sometimes for each individual image), and are not well generalizable between different materials and experimental setups. Consequently, less than one percent of data is analyzed, with selection driven by operator choice ("best" image from the stack), which also drives the experimental planning and microscope design. Utilization of deep learning, which has proven to be one of the most successful machine learning

[1]Computational Data Analytics Group
[2]Geographic Information Science and Technology Group
[3]Imaging, Signals, & Machine Learning Group
[4]Institute for Functional Imaging of Materials
[5]Center for Nanophase Materials Sciences

SC18, November 11-16, 2018, Dallas, Texas, USA
978-1-5386-8384-2/18/$31.00

methods for image analysis, offers a path to overcome these limitations. Unlike previous approaches to image processing and computer vision, deep learning does not rely on hand-engineered features; instead, it learns the features needed for the image processing task from the data. The unique aspect of data in high resolution electron microscopy is the presence of well-defined ground truth, i.e., atomic positions (at least, prior to some dramatic structural transformations), and reasonably well understood physics of the imaging process, which allows creation of a training set with problem-specific physical constraints.

Applying deep learning presents a new problem: defining an optimal network topology *and* hyperparameters associated with the corresponding deep learning network, such as the number of layers, the type of each layer, and the corresponding parameters for each layer type as described in [7]. It has been shown that customizing a network's hyperparameters and topology to match the dataset can result in networks with significantly higher accuracy [8], [9]. A typical approach when applying deep learning to a new scientific dataset is to use some other deep learning network "off-the-shelf" and then perhaps hand-tune some of the hyperparameters to be more well-suited to the new dataset while leaving the network topology fixed. For non-deep learning experts, it is often not clear how to update the hyperparameters effectively to improve network performance. Evaluating a particular set of hyperparameters on a dataset is computationally intensive; specifically, it requires training a network with those hyperparameters to see how well it performs the task. Then, this computationally intensive task must be repeated for each hyperparameter set to be evaluated. Hypothetically, consider a simple deep learning network with a fixed topology (i.e., number of layers, layer types, order of layers) consisting of 5 convolution layers, each of which require at least two hyperparameters, for a total of 10 hyperparameter values needed to be defined. If we assume that each hyperparameter could take on a value ranging from 1 to 100, then there are 100^{10} (1E20) candidate networks. To complicate matters, our hypothetical network has several additional hyperparameters that would need to be defined. Furthermore, if the topology (i.e., number of layers, layer types) is to be changed as well, then the search space grows exponentially large. Requiring domain scientists to create an optimal deep learning network for their particular data will most likely produce unsuccessful results, since even deep learning experts rely primarily on intuition and trial-and-error experimentation in determining an appropriate network.

To address this challenge, this work leverages an artificial intelligence system called Multinode Evolutionary Neural Networks for Deep Learning (MENNDL) [10], developed to intelligently optimize both the deep learning network topology *and* the corresponding hyperparameters. It effectively parallelizes and scales the evaluation of millions of networks within hours, utilizing the computational power of a GPU-based HPC system. Consequently, this system creates deep learning networks that are highly tuned to an application space and go beyond what a domain expert would conceive as being optimal. MENNDL reduces human-effort from months to hours in designing a deep learning network, which enables scientists to move more quickly toward finding new discoveries within their data. In this particular application, MENNDL constructs a deep learning network capable of atomic level mapping of chemical transformation pathways in solids. The MENNDL designed network outperforms current state-of-the-art results as well as human expert results. Though all of Summit's 4,600 nodes are not currently available, we demonstrate the scalability of our approach on up to 4,200 nodes and extrapolate out the potential performance on all of Summit. *For 4,200 nodes, we achieve a sustained, measured performance of 152.5 PFlops with a projected sustained 167 PFlops using 4,600 nodes of Summit.*

IV. Current State of the Art

Our research spans several domains (electron microscopy data analysis, deep learning, and hyperparameter optimization) that have not previously been combined into a single task leveraging HPC. Thus, there is no single point of comparison, and as such, we report the state-of-the-art for each area of research.

A. Scanning Transmission Electron Microscopy

Despite large volumes of data generated in dynamic STEM experiments, including solid-state reactions and phase transformations, presently we do not learn much from this data. Practically, via manual visual analysis, we can discern global changes and some qualitative information of atomic dynamics. The vast majority of analysis is manual or qualitative interpretation. The current state-of-the-art method for the analysis of individual STEM images relies on singular value decomposition-based denoising technique and the pattern matching-based techniques for identifying atoms in the images [11]. However, due to computational costs and poor generalizability (it requires a manual input for each separate image) this approach is practically unsuitable for the automated analysis of large volumes of high-resolution STEM data. As for the analysis of individual atomic defects, the current state-of-the-art in locating defects in each frame of dynamic STEM data (on periodic lattice structures) is the Fourier-transform-based image subtraction [12]. Unfortunately, each frame requires manual fine tuning of threshold values, especially for those before and after a beginning of nucleation of a new phase, making it impossible to do real-time data analytics, which is necessary for the automated manufacturing of nanodevices. Finally, when it comes to identification (and interpretation) of subtle atomic defect structures, hand tuning separate frames from dynamic STEM data inevitably introduces a human bias in the initial defect identification. Parenthetically, the microscopy field knows several examples of when hand tuning parameters in Fourier-based methods on complex structures resulted in a "discovery" of phases and structures that were not actually present in the system. Because it requires manual intervention, less than one percent of experimental data is ever analyzed. With rapid detectors and storage capacities

capable of collecting million-frame movies, this problem is particularly acute. We can now collect extreme amounts of data and chemical reactions in solids, induce and control atomic motion atom by atom, and learn fundamental physics - but reliance on human analysis limits progress.

B. Deep Learning on HPC

As deep learning is a computationally complex machine learning approach, its popularity in the HPC community [13] continues to rise. Indeed, it is well suited for GPU-based HPC systems. While considerable work in scaling a single network across many nodes exists either through data parallelism or model parallelism, there are practical and algorithmic limits to what can be achieved using these approaches. While scaling data parallel approaches across many thousands of GPUs in order to achieve a high Flops measurement is a simple task, actually achieving a proportional decrease in time-to-solution over what can be achieved with tens or hundreds of GPUs has not been convincingly shown. This is largely due to the resulting large effective batch sizes (i.e. if the number of nodes is increased 1000x then the batch size is increased 1000x) and the effect it has upon convergence. Current work has only scaled to hundreds of deep learning accelerators [13], [14], and even then this relies on assumptions about input size and the networks used that may not hold true generally. Additionally, many barriers exist in scaling asynchronous gradient descent methods for decreased time-to-solution [15]. The primary limitation is the increase in batch size as nodes are added. The ideal batch size for optimal convergence per training example is somewhere between 32-512 training examples for many problems [16]. If we allow the batch size to grow during training, the batch size can increase to 64,000 for some problems during the final stages of training while still providing a decrease in time-to-solution, as shown in [14]. However, these batch sizes are only realized during the final stage of training. Even assuming an effective batch size of 64,000 is feasible for a particular problem during the final stages of training, that would only represent a batch size of less than 3 per GPU on Summit's 27,600 GPUs. This batch size is too small to maximize usage on each GPU.

C. Hyperparameter Optimization

For many applications that could benefit from leveraging deep learning, determining the network topology and the appropriate hyperparameter values remains a key problem. For example, hyperparameters such as the number of layers and the corresponding hyperparameters for each layer type used must be defined. Given the wide range of potential hyperparameter options, and the nonlinear relationship among them, an enormous search space that requires significant computational power to optimize exists.

Despite the development of advanced deep learning frameworks [17] which make deep learning very accessible, hyperparameter selection remains a significant barrier and typically requires deep learning domain expertise to achieve satisfactory results, especially for datasets that have not been as heavily studied as natural image datasets like ImageNet [18].

Progress has been made toward reducing the barrier of hyperparameter optimization. In the work of [19] it was shown that a simple random search outperformed manual tuning performed by an expert. Then, in [8] [9], it was shown that Tree of Parzen Estimators (TPE) outperformed random search. These works showed that by leveraging a small cluster computer, optimized networks could be achieved that outperform human experts in a fraction of the time. More recently, the work of [20] showed that using a surrogate radial basis function (RBF) outperforms TPE. However, despite the progressive improvements, the significant drawback to all of these approaches is that they do not assist in defining the network topology (i.e., the number of layers and layer types); and, the topology has a significant impact on the overall performance of a deep learning network for an application.

V. Innovations Realized

A. Nanoscience and Nanotechnology Innovations

Using MENNDL, we develop a deep learning network for rapid analysis of the dynamic STEM data from 2-dimensional material under electron beam irradiation. This custom network allows us to create a library of defects, map chemical transformation pathways at the atomic level, including detailed transition probabilities, and explore subtle distortions in local atomic environment around the defects of interest [21]. Employing the custom network, we are able to get an unprecedented insight into the nature and mechanisms of solid-state reactions and electron-beam-matter interactions on the atomic level, which is of crucial importance to controllable nanofabrication as well as to fundamental atomic-scale chemistry. Furthermore, the developed network solves the problem of instructing a computer how to choose automatically the "best region" in a sample to make a measurement or perform atomic manipulations without human supervision. This is a critical step towards a fully-automated ("self-driving") microscope.

B. Deep Learning Innovations

Our deep learning framework is called Multinode Evolutionary Neural Networks for Deep Learning (MENNDL). MENNDL relies on two optimization methods, genetic algorithms [10], [22] and support vector machines [23], [24], to intelligently optimize deep learning network topologies and hyperparameters. MENNDL effectively parallelizes network evaluation and fully utilizes the computational power of Summit. *The resulting software framework facilitates the discovery of an optimal deep learning network for a particular scientific dataset in a quick, efficient, and automated manner using a GPU-based HPC system.*

This framework has two components. The first component utilizes a genetic algorithm to determine the appropriate topology (number and type of layers) and corresponding hyperparameters of a deep learning network. The set of network topologies that achieve the highest accuracy are then used as input to the second component, which utilizes a support

vector machine to further refine hyperparameters within a fixed topology network. Like other approaches for hyperparameter optimization, including random and grid search, MENNDL benefits from additional compute resources in order to evaluate more potential solutions quickly. Unlike random and grid search, however, MENNDL utilizes machine learning and the results from previously evaluated networks in determining new networks to evaluate. As such, there is less wasted compute time on poorer performing networks and more time training and evolving better performing networks.

Like the work of [8] [9], MENNDL can create deep learning networks within hours that will perform as well or better than what a domain expert could create within months. However, unlike [8] [9], which only optimized some of the hyperparameters, the key innovation is that this framework effectively utilizes Summit to optimize the *entire* topology (e.g., number and type of layers) and hyperparameters of a deep learning network for scientific data in general, and in this demonstration, electron microscopy data specifically. Consequently, this framework can create deep learning networks that are more highly tuned to an application space and go beyond what a domain expert would conceive as being optimal. This reduction of months to hours in designing a deep learning network topology enables scientists to move more quickly toward finding new discoveries within their data. In this particular application, we have demonstrated this approach enables atomic level mapping of chemical transformation pathways in solids.

C. Optimization Innovations for Summit

There are several key innovations in the development of MENNDL that enables it to scale to effectively utilize Summit's resources.

1) Compute Utilization Optimizations: The genetic algorithm component of MENNDL is implemented using a global single-population master-slave genetic algorithm (GA) [**?**], in which one node (the master) is used to host the population and evolutionary processes of the algorithm and the remaining nodes are used to evaluate the fitness of networks in the population (Figure 1). Each individual in the population represents a single deep learning network and is shown in Figure 1 as a double-helix. The fitness score for each network in the population corresponds to its accuracy on a validation data set after training for a fixed number of iterations on a training data set using Caffe [26]. Depending on the network topology and hyperparameters, the time required to evaluate a particular network can vary significantly (as shown in the figure by different length double-helixes), and GAs are inherently synchronous. As a result, while the GA is parallelizable, its utilization of the machine decreases with scalability. To overcome this issue and ensure maximum utilization of computational resources as well as maximizing the number of networks that can be evaluated within a given time, MENNDL utilizes an asynchronous genetic algorithm architecture similar to [27]. Thus, each GPU of each node is continuously evaluating a network rather than waiting for all of the other nodes to finish evaluation of their respective individuals.

Once MENNDL establishes a network topology and an initial set of hyperparameters, the support vector machine (SVM) component of MENNDL is then used to tune the hyperparameters on the fixed topology network. In particular, an SVM is used to predict whether certain hyperparameters will perform well for the given network topology. During this component of MENNDL, we first utilize each compute node to evaluate as many random hyperparameter sets as possible in a pre-defined fixed time period. Once this time period is complete, the master node collects all of the results, builds the appropriate SVM from those results, and then the SVM is transmitted to each of the nodes. Then, each node determines a hyperparameter configuration that is predicted to perform well based on the SVM provided by the master node. Once a hyperparameter set that is predicted to perform well has been found, the node trains that network for a pre-defined fixed time period. All of the compute nodes are continuously evaluating/training one or more networks throughout except when the master node is building the appropriate SVM; however, very little time is spent in the building of the SVM on the master node, resulting in high node utilization throughout.

2) I/O Transfer Optimizations: A key bottleneck for training deep learning networks on HPC is data transfer costs. Unlike simulation codes that typically perform a significant number of write operations, machine learning code performs a significant number of read operations. In the case of deep learning, a single network will train by iteratively reading over a dataset thousands of times. In this particular application of hyperparameter optimization utilizing HPC, there are tens of thousands of networks all performing thousands of read operations from the same data set. To minimize data transfer costs and increase the utilization of the GPUs, MENNDL takes advantage of one of Summit's key architectural features where "each node has 1.6TB of non-volatile memory that can be used as a burst buffer" [28]. At the beginning of the job execution, local copies of the data are created on the burst buffer of each node. Then, throughout the network evaluation process, the local copies of the data are utilized rather than requiring continuous data transfers from the parallel file system throughout training.

3) Communication Optimizations: MENNDL includes optimizations that reduces the size of the communicated messages between nodes. Deep learning networks are typically specified in a custom file format based on the framework being used to implement and train the network. As noted above, we use the deep learning framework Caffe for network evaluation. Caffe uses a file format called "prototxt" to specify a network's topology and hyperparameters. For the GA component of MENNDL, we devised a compression technique on the network specification by encoding network topology and hyperparameters in a fixed length genome string. This genome string is then used to communicate networks between the master node coordinating the GA and the compute nodes that will train the networks. Prototxt files are on the order of kilo-

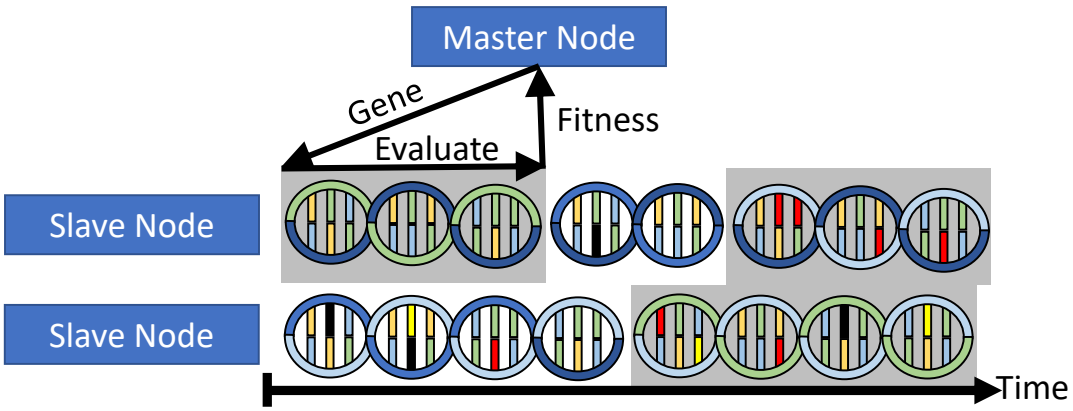

Fig. 1. An illustration of the genetic algorithm component of MENNDL.

bytes, whereas the network genome representation, which is decompressed to form a full prototxt file at the compute node, is on the order of bytes. As such, this results in a thousand-fold lossless compression technique for network representations. Similarly, during the SVM component of MENNDL, we use a fixed length string to communicate between the master and slave nodes. Since a fixed network topology is defined prior to the SVM component execution, the length of this string is determined based on the exact number of hyperparameters to be optimized. Again, rather than communicating the prototxt file (on the order of kilobytes) to define the network, only the array of hyperparameter values and corresponding accuracy value (on the order of bytes) is communicated.

VI. Performance Measurements

A. Application Used to Measure Performance

1) Scanning Transmission Electron Microscopy Data: We analyzed a STEM "movie" showing the phase evolution and atomic defect dynamics in a fixed region (15nm×15nm) of a single layer of molybdenum (Mo)-doped tungsten disulphide (WS_2) under 100 kV electron beam irradiation. We utilized the fact that macroscopically (on the length scale of the image) the defects can be discovered via the Fourier method, providing the ground truth for training. Thus, we utilized a manually-tuned Fourier transform based approach to label defects within the first frame of the video (Frame 0). We divide this image into tiles to produce a training and validation set for hyper-parameter optimization. The resulting best network is then used to produce detection results on subsequent frames not used for training or validation. The initial frame (used for training and validation), as well as two subsequent frames (used to evaluate the effectiveness of deep learning on this problem), are shown in Figure 2. It is clear that the Fourier transform-based method labels tuned for the first frame produce radically different results from the human-expert labels and thus, it is not a practically useful as an automated labeling method. The Fourier transform-based method would need to be re-tuned for each frame, while the MENNDL designed network does not.

2) MENNDL Software Details: The GA component of MENNDL's code is written in C++ and utilizes MPI for node-to-node communication. The master node uses CPU resources only. The SVM component of MENNDL's code is written in Python, and we use the SVM implementation in Python's scikit-learn package using a radial basis function (RBF) kernel. Each slave or worker node that performs network evaluations launches `nvcaffe` (NVIDIA's Caffe fork) to train a particular network configuration for both components of MENNDL.

B. System and Environment

The system used for these measurements is Oak Ridge Leadership Computing Facility (OLCF)'s Summit machine. Upon completion, Summit will have approximately 4,600 compute nodes. Each node will contain two IBM POWER9 CPUs and six NVIDIA Volta GPUs connected together with NVIDIA's NVLink [28]. In this work, we measure our application performance on the available nodes on Summit and project performance for the full 4,600-node system.

Measuring floating point operations (Flops) for our software framework is a non-trivial task. In particular, MENNDL stochastically generates and then evaluates (through the training process) deep learning networks. The specific hyperparameters chosen for any network have an enormous impact on the performance of the network. The hyperparameters determine what is possible for the network to learn and, as a result, the accuracy of the trained network. The hyperparameters also have a direct impact on the number of Flops required to process a batch of training data, the size and shape of that data throughout the network hierarchy, and the efficiency of GPU utilization. In order to measure the Flop performance of either application, it is first necessary to understand the performance of the individual GPUs. Since the network hyperparameters are being randomly generated and those hyperparameters impact the Flop performance, the performance of any individual GPU can be viewed as a random variable $X_1, X_2, ..., X_n \sim X$. The unknown distribution X will be a function of many things including what type(s) of networks MENNDL is generating. While the specific distribution of X may not be known, its

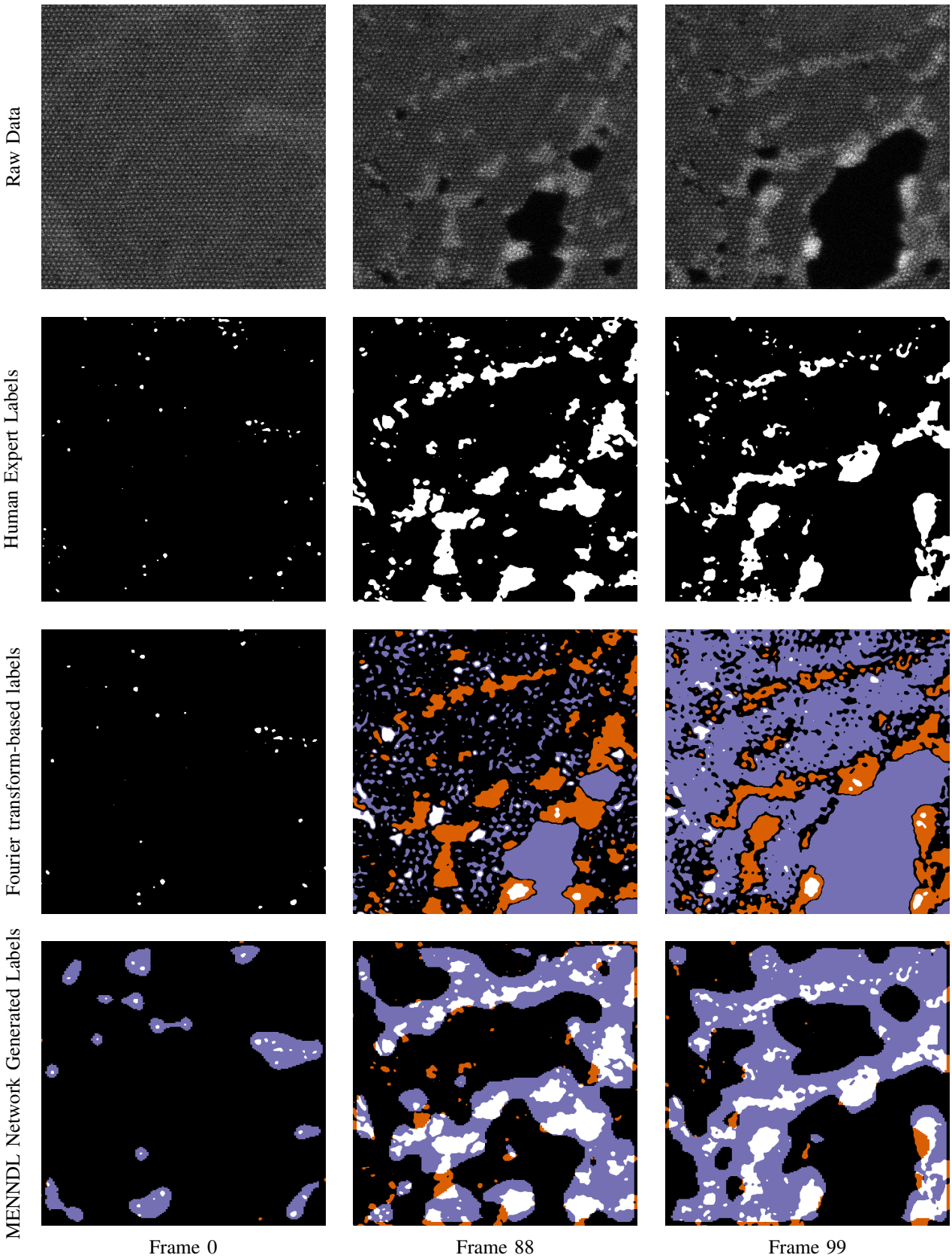

Fig. 2. Identification of structural defects in scanning transmission electron microscopy data. First row: raw experimental data. Second row: result of a human expert (semi-)manually labeling atomic defects. Another human expert may label this slightly differently. Third row: result of a Fourier-transform based method tuned to Frame 0. Fourth row: result produced by convolutional neural network. In the third and fourth rows, white indicates correctly identified defects (good), purple pixels identified as defects that were not defects (bad), and orange indicates defects that were not identified (worse). Black indicates regions that contained no defects and were identified as such.

mean (μ) and variance (σ^2) can be empirically estimated. Regardless of the distribution of X, once μ and σ^2 are known, the (typical) performance of the application as a whole can be accurately determined. As it is simply the sum of i.i.d. random variables, the whole application performance $Z = \sum^n X_i$ is a random variable which is normally distributed: $Z = \sum^n X_i \sim \mathcal{N}(n\mu, n\sigma^2)$.

Deep learning network training is commonly performed using Stochastic Gradient Descent (SGD) [29] [30]. A random subset (a batch) of the data is chosen; it is then fed through the network (forward pass); a loss function is computed; finally, network weights are updated using computed gradients (backward pass). The number of floating point operations required to complete one forward and backward pass (one iteration) on a single batch of data can vary even when all network hyperparameters are held constant. To compute the *typical* number of operations required per batch for a single (fixed) set of hyperparameters, NVIDIA's profiler `nvprof` is used to count the total number of operations performed for N batches for N in $\{4, 6, 8, 10, ..., 18\}$. This gives us 8 data pairs each containing a number of training batches and a total number of floating point operations. A best-fit line for these data points is computed. The y-intercept captures the number of floating point operations required to initialize this particular network while the slope captures the typical number of operations per batch. MENNDL utilizes `nvcaffe`, which records the (average) number of batches per second. From this, the average Flop-performance of this (fixed) network can be computed by multiplying the slope (floating point operations per batch) by the number of batches per second. This provides X_i for a single network. This process is repeated in order to generate as many data points X_i as desired.

Ultimately, in order to estimate accurately μ and σ^2 (the mean and variance of the distribution X), deep learning networks were profiled on a set of the available nodes on Summit (specifically, one network for each GPU on Summit). Once μ and σ are estimated, the expected, overall system performance $Z = \sum^n X_i$ can be determined. Z is normally distributed with mean $n\mu$ and variance $n\sigma^2$, where n is the number of networks evaluated.

VII. Performance Results

A. Sustained and Peak Performance

To measure performance as described in Section VI-B, we first determine the distribution of X (the performance of a single GPU) as well as its mean (μ) and variation (σ^2). To this end, we profiled $4,200 \times 6 = 25,200$ networks (one network on each GPU using 4,200 nodes of Summit). This gives us statistically large sample of X_i values. The distribution is shown in Figure 3.

We measured a mean, GPU-level, performance of 5.039 TFlops (single precision) or 6.053 TFlops (mixed, single-half precision) with standard deviation of 1.725 TFlops and 2.701 TFlops respectively. Though we show the distribution for both single precision only and mixed single-half precision, throughout the rest of this work, we focus only on mixed single-half precision, as that achieves better performance overall.

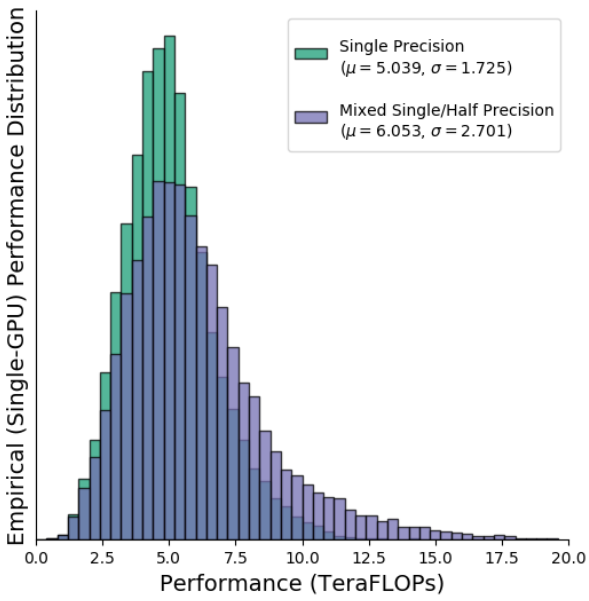

Fig. 3. The observed distribution X from using single precision training (single) and mixed single-half precision training (half).

Given these measurements, we determine the expected, overall system performance $Z = \sum^n X_i$, where Z is normally distributed with mean $n\mu$ and variance $n\sigma^2$. In our case $n = 4,200 \times 6$ (since we were only able to run on up to 4,200 nodes of Summit), but it is reasonable to expect the performance to hold up to the full machine (about 4,600 nodes) since the ratio of communication to computation is so small. In Figure 4 we show the observed and projected performance (in PFlops) for mixed single-half precision training. Note that this figure is in log-log scale (so the deviation appears to shrink as the number of nodes increases). As shown in this figure, the sustained performance for mixed single-half precision is approximately 152.5 PFlops for 4,200 nodes. When Summit's full 4,600 nodes are available, we project that we will achieve approximately 167 PFlops sustained performance for mixed single-half precision. These results were obtained using preproduction system software, not-yet generally available software, and performance results on Summit are expected to improve over coming months.

Our Flops measurement is done just using the GPUs on Summit. The theoretical peak performance of an individual NVIDIA Volta GPU is 15.6 TFlops for single precision performance. Thus, the theoretical peak performance of Summit's full 4,600 node system (with six GPUs per node) is approximately 433 PFlops. Given that, *our projected value of 167 PFlops mixed-precision is achieving approximately **38.6**% of Summit's theoretical peak single-precision performance.* Peak performance on our particular application would require

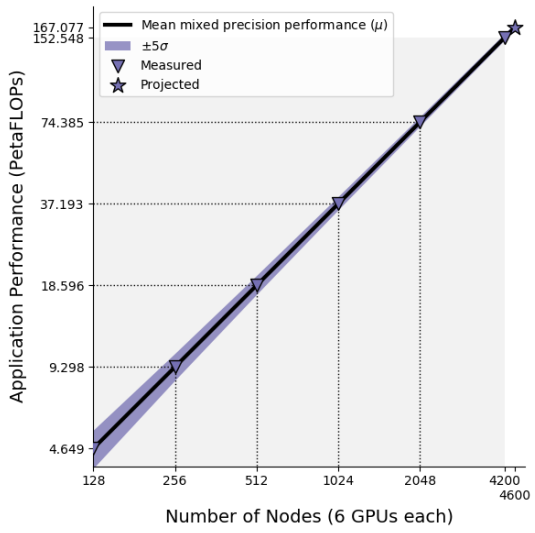

Fig. 4. Mixed single-half precision, whole application performance. The shaded region shows our measured performance.

every GPU running a network that maximizes the performance of that GPU. This is unreasonable to expect, since different network topologies and hyperparameter settings have radically different performance characteristics (as shown in Figure 3), and evaluating different networks is key to the performance of our application. It is worth noting that the best network performance we observed (as shown in Figure 3) achieved approximately 15.6 TFlops in single precision. Thus, if that network was duplicated across all of the GPUs of Summit, our peak application performance would approach Summit's theoretical peak performance of 433 PFlops, however, there is little scientific value in evaluating a single network tens of thousands of times.

B. Weak Scaling

Figure 4 shows our weak scaling results, in which we increase the number of networks to evaluate as we increase the number of nodes. The problem size per node is six networks (as we evaluate one network on each GPU per node). Because the time to evaluate each network can vary as each has a unique set of hyperparameters (as shown in Figure 3), there is some variation expected. Overall, the variance in per network Flops is small and is not dependent on the number of nodes. Thus, our application exhibits linear weak scaling.

C. Strong Scaling

MENNDL performs a hyperparameter optimization, so the more samples that are made of the search space, the more likely it is that an optimal set of hyperparameters will be found. As a result, we scale the size of the problem (the number of networks to evaluate) to match the resources available, so weak scaling results are more illustrative of our expected performance. However, we may also define a strong scaling result by framing our problem as gaining an understanding of the search space landscape in order to drive hyperparameter optimization.

To sufficiently understand the landscape of the search space, a minimum number of hyperparameter configurations should be sampled, which can define our minimum problem size. Since the number of hyperparameters can change based on the network's topology, this is a non-trivial value to define for MENNDL as a whole. To simplify the problem, we identify a lower bound for a minimum problem size for MENNDL by restricting our attention to a minimum problem size for the SVM component of MENNDL. A canonical "small" deep neural network, LeNet [24], has 11 hyperparameters, each of which may take on a range of values. To gain a base understanding of what the hyperparameter search space looks like for a given problem on LeNet, it is reasonable to assume that at least 2 to 3 values will be chosen for each hyperparameter value. Thus, for LeNet, between 2^{11} and 3^{11} (2,048 and 177,147) networks should be evaluated. For our strong scaling results, we selected several different numbers of networks (50,000, 100,000, and 200,000), which represent different numbers of LeNet hyperparameter configurations to evaluate in order to gain a nominal understanding of the hyperparameter search space.

As shown in Figure 5, we have very promising strong scaling results. The key issue that prevents perfect strong scaling for a fixed problem size (a fixed number of networks), is that different networks take different amounts of time to evaluate. Thus, as evaluations are completed, eventually there are some idle nodes for a fixed problem size, because some nodes are still being used to evaluate the networks while others have no new networks to evaluate. In practice, MENNDL will continue to generate new networks to evaluate, so nodes will not be idle because there is not a network to evaluate.

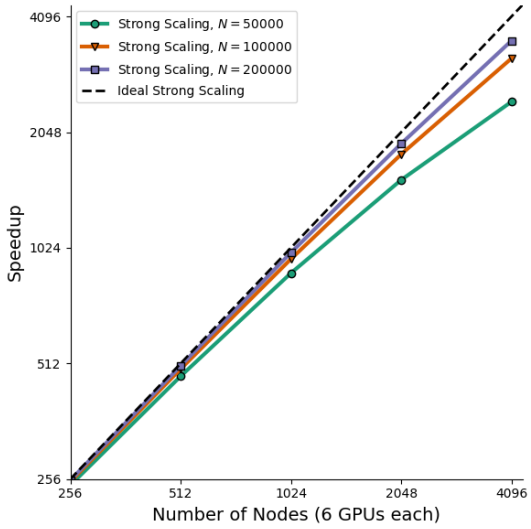

Fig. 5. Strong scaling results for different problem sizes (different number of networks to evaluate.

D. Application Results

To train a deep convolutional neural network (CNN) [31] for atomic defect identification, we label an initial image via the Fourier method, providing the ground truth for training. Unlike the Fourier method, once the CNN is trained, it relies on local edge properties for identification of defects and is thus stable towards rotations and fragmentations of the lattice. This allowed us to train a CNN using only the (properly augmented) first frame of dynamic STEM data (movie) or a single image obtained before recording a movie.

We utilized the human expert ground truth to train and validate CNNs using MENNDL. We trained MENNDL for 65 generations and achieved a validation accuracy of 99.51% using Frame 0 of Figure 2 broken up into small tile images. The best performance seen as a function of the current generation is shown in Figure 6. The performance of the best network at generation 0 in the figure is what one would reasonably expect for the performance of a network that was created by a scientist who had no prior knowledge about what network topology and hyperparameter settings would perform well for their problem. As shown in the figure, MENNDL is able to evolve a network topology and hyperparameter set over time that decreases the validation error, customizing the network to the dataset.

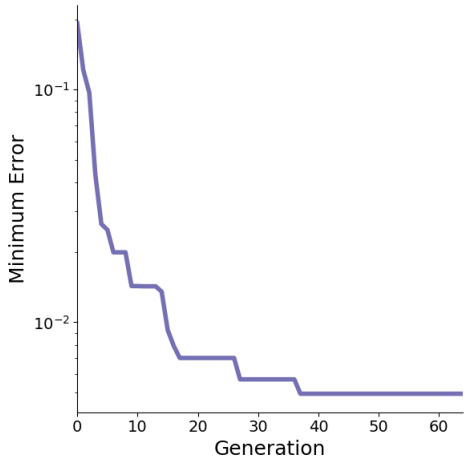

Fig. 6. Best validation error observed vs generation of MENNDL training.

The images in row 4 of Figure 2 show how the best MENNDL trained network performs on the expert-labeled testing images. In contrast to the Fourier transform-method based results shown in row 3 of Figure 2, the CNN-based defect identification shows a stable behavior throughout the entire movie allowing the extraction of most of the atomic defects in a fast and automated fashion. In other words, the CNN generalizes better what was learned from the frame 0 image to apply to frames 88 and 99, thus better capturing where real defects are occurring. The Fourier transform-based method, which is the current state of the art, must be re-tuned by hand for each frame in order to have higher accuracy.

VIII. Implications

The key contributions of this work are:

- The first artificial intelligence system operating at over 100 PFlops.
- An artificial intelligence system, MENNDL, to utilize HPC for automatic, intelligent, and rapid design of deep learning networks for scientific datasets.
- The first known approach to automatically identifying atomic-level structural information in scanning transmission electron microscopy (STEM) data.

It is hard to overestimate the impact of the ability to automatically analyze all of the data available from current STEM experiments. The previous state-of-the-art approaches for image-based feedback in STEM experiments require hand-tooling for each individual image, thus making them useless for real-time application. In this work, we introduce a way to use HPC to enable the automation of designing a deep learning network for automated STEM image analysis. As a result, this MENNDL deep learning approach is the first to overcome the issues associated with the previous state-of-the-art approach. MENNDL is fast, efficient, and capable of generalizing to new images without manual hand-tooling. As opposed to utilizing off-the-shelf networks that were trained on natural images, a deep learning network is tailored specifically for STEM data, which can then be used to produce image-based feedback in real-time. This feedback is required in order to enable atom by atom fabrication, which is the ultimate goal of the field of nanotechonlogy. As the U.S. National Nanotechnology Initiative Strategic Plan [1] states:

> These advances promise to improve human health and quality of life ... nanotechnology has become ubiquitous in our daily lives and can be found in a wide variety of commercial products including healthcare products, cosmetics, consumer electronics, apparel, and automobiles.

Currently, the number of existing commercial STEM platforms worldwide and number of trained operators is skyrocketing, with tools available in most universities and research centers worldwide. The availability of cloud-based computation infrastructure provides the capability to store and manage the data. What is lacking is the high performance computing that enables transforming the data coming from STEM platforms (images, movies capturing dynamics, hyperspectral images, ptychographic datasets) into material-specific descriptors such as atomic coordinates and trajectories, valence states and orbital populations, as well as signatures of quantum phenomena. Such transformation will have to be fast (to enable real-time feedback), unbiased, and quantitative. This last requirement sets physics-based data analysis apart from the largely qualitative and semiquantitative labeling needs that emerge in classical applications of deep learning such as medicine, biology, and image recognition. This requirement in turn necessitates optimization of the deep learning network topology and hyperparameters to those optimal for this specific

domain. Prior to this work, the capability for making these tools was unavailable.

This work produces deep learning networks that are tuned specifically for STEM data. These networks can be made available to the greater electron microscopy community, much like networks that have been tuned manually for natural images have been shared in the computer vision community. As such, scientists can leverage these networks and, with much lower computational costs, tune these networks for their electron microscopy datasets in order to automate their own specific image analysis tasks. *We expect that this will dramatically accelerate the rate of scientific discovery.*

Beyond the impact on the field of nanofabrication, MENNDL also provides the key capability for automatically and quickly customizing the topology and hyperparameters of deep learning networks for many scientific datasets by leveraging the Summit supercomputer. In this work, we describe its application to STEM data. Additionally, it has been used to custom-build deep learning networks that outperform networks designed and hand-tuned by domain experts for a variety of datasets, including datasets from high energy physics, small angle neutron scattering, medical imaging [10], and remote sensing [24]. In these applications, state-of-the-art results have been achieved. Not only do these networks outperform those created by domain experts, they are designed using HPC in a matter of hours, as opposed to the weeks or months it takes for a person to hand-tune a network. Moving forward, MENNDL will enable non-deep learning experts and non-HPC experts to leverage some of the world's fastest computers to build customized deep learning networks for scientific datasets in a matter of hours. MENNDL will allow for more rapid understanding of the types of problems that can be solved using deep learning, as well as faster application of state-of-the-art machine learning techniques to a wider variety of scientific fields.

In summary, we have developed an artificial intelligence system (MENNDL) that is capable of automatically creating deep learning networks customized for particular scientific datasets. We have demonstrated that by running MENNDL on Summit at over 100 PFlops we can produce state-of-the-art results on STEM data image analysis.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Robinson Pino, program manager, under contract number DE-AC05-00OR22725.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

References

[1] T. Kalil. (2016) National nanotechnology initiative strategic plan. [Online]. Available: https://www.nano.gov/sites/default/files/pub_resource/2016-nni-strategic-plan.pdf

[2] J. Lin, O. Cretu, W. Zhou, K. Suenaga, D. Prasai, K. I. Bolotin, N. T. Cuong, M. Otani, S. Okada, A. R. Lupini *et al.*, "Flexible metallic nanowires with self-adaptive contacts to semiconducting transition-metal dichalcogenide monolayers," *Nature Nanotechnology*, vol. 9, no. 6, 2014.

[3] Z. Yang, L. Yin, J. Lee, W. Ren, H.-M. Cheng, H. Ye, S. T. Pantelides, S. J. Pennycook, and M. F. Chisholm, "Direct observation of atomic dynamics and silicon doping at a topological defect in graphene," *Angewandte Chemie*, vol. 126, no. 34, 2014.

[4] S. V. Kalinin, A. Borisevich, and S. Jesse, "Fire up the atom forge," *Nature*, vol. 539, no. 7630, 2016.

[5] N. Jiang, E. Zarkadoula, P. Narang, A. Maksov, I. Kravchenko, A. Borisevich, S. Jesse, and S. V. Kalinin, "Atom-by-atom fabrication by electron beam via induced phase transformations," *MRS Bulletin*, vol. 42, no. 9, 2017.

[6] S. V. Kalinin and S. J. Pennycook, "Single-atom fabrication with electron and ion beams: From surfaces and two-dimensional materials toward three-dimensional atom-by-atom assembly," *MRS Bulletin*, vol. 42, no. 9, 2017.

[7] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures." in *Neural Networks: Tricks of the Trade (2nd ed.)*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Mller, Eds. Springer, 2012, vol. 7700.

[8] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. USA: Curran Associates Inc., 2011.

[9] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. PMLR, 2013.

[10] S. R. Young, D. C. Rose, T. Johnston, W. T. Heller, T. P. Karnowski, T. E. Potok, R. M. Patton, G. Perdue, and J. Miller, "Evolving deep networks using hpc," in *Proceedings of the Machine Learning on HPC Environments*. ACM, 2017.

[11] S. Somnath, C. R. Smith, S. V. Kalinin, M. Chi, A. Borisevich, N. Cross, G. Duscher, and S. Jesse, "Feature extraction via similarity search: application to atom finding and denoising in electron and scanning probe microscopy imaging," *Advanced Structural and Chemical Imaging*, vol. 4, no. 1, 2018.

[12] W. Lin, Q. Li, A. Belianinov, B. C. Sales, A. Sefat, Z. Gai, A. P. Baddorf, M. Pan, S. Jesse, and S. V. Kalinin, "Local crystallography analysis for atomically resolved scanning tunneling microscopy images," *Nanotechnology*, vol. 24, no. 41, 2013.

[13] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *CoRR*, vol. abs/1802.09941, 2018. [Online]. Available: http://arxiv.org/abs/1802.09941

[14] S. L. Smith, P.-J. Kindermans, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *International Conference on Learning Representations*, 2018.

[15] O. Bhardwaj and G. Cong, "Practical efficiency of asynchronous stochastic gradient descent," in *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, ser. MLHPC '16. IEEE Press, 2016.

[16] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2017.

[17] A. Parvat, J. Chavan, S. Kadam, S. Dev, and V. Pathak, "A survey of deep-learning frameworks," in *2017 International Conference on Inventive Systems and Control (ICISC)*, 2017.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, Dec. 2015.

[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 1, Feb. 2012.

[20] I. Ilievski, T. Akhtar, J. Feng, and C. Shoemaker, "Efficient hyperparameter optimization of deep learning algorithms using deterministic rbf surrogates," in *31st AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[21] A. Maksov, O. Dyck, K. Wang, K. Xiao, D. B. Geohegan, B. G. Sumpter, R. K. Vasudevan, S. Jesse, S. V. Kalinin, and M. Ziatdinov, "Deep learning analysis of defect and phase evolution during electron beam induced transformations in ws2," *ArXiv e-prints*, Mar. 2018.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975, second edition, 1992.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, Sep. 1995.

[24] T. Johnston, S. R. Young, D. Hughes, R. M. Patton, and D. White, "Optimizing convolutional neural networks for cloud detection," in *Proceedings of the Machine Learning on HPC Environments*, ser. MLHPC'17. New York, NY, USA: ACM, 2017.

[25] E. Cant-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, 1998.

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[27] M. Golub and L. Budin, "An asynchronous model of global parallel genetic algorithms," in *Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems EIS2000*. Scotland, UK: University of Paisley, 2000.

[28] OLCF. (2018) Summit system overview. [Online]. Available: https://www.olcf.ornl.gov/for-users/system-user-guides/summit/system-overview/

[29] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, 1951.

[30] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Physica-Verlag HD, 2010.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.