Co-design Center for Exascale Machine Learning Technologies (ExaLearn)

Journal Title
XX(X):1–19

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Francis J. Alexander[2], James Ang[4], Jenna A. Bilbrey[4], Jan Balewski[6], Tiernan Casey[7], Ryan Chard[1] Jong Choi[3], Sutanay Choudhury[4], Bert Debusschere[7], Anthony M. DeGennaro[2], Nikoli Dryden[8,11], J. Austin Ellis[7], Ian Foster[1], Cristina Garcia Cardona[3], Sayan Ghosh[4], Peter Harrington[6], Yunzhi Huang[4], Shantenu Jha[2], Travis Johnston[4], Ai Kagawa[2], Ramakrishnan Kannan[4], Neeraj Kumar[4], Zhengchun Liu[1], Naoya Maruyama[8], Satoshi Matsuoka[10,9], Erin McCarthy[8,12] Jamaludin Mohd-Yusof[5], Peter Nugent[6], Yosuke Oyama[8,9], Thomas Proffen[3], David Pugmire[3], Sivasankaran Rajamanickam[7], Vinay Ramakrishniah[5], Malachi Schram[4], Sudip K. Seal[4], Ganesh Sivaraman[1], Christine Sweeney[5], Li Tan[2], Rajeev Thakur[1], Brian Van Essen[8], Logan Ward[1], Paul Welch[5], Michael Wolf[7], Sotiris S. Xantheas[4], Kevin G. Yager[2], Shinjae Yoo[2], Byung-Jun Yoon[2]

Abstract

Rapid growth in data, computational methods, and computing power is driving a remarkable revolution in what variously is termed *machine learning* (ML), *statistical learning*, *computational learning*, and *artificial intelligence*. In addition to highly visible successes in machine-based natural language translation, playing the game Go, and self-driving cars, these new technologies also have profound implications for computational and experimental science and engineering, as well as for the exascale computing systems that the Department of Energy (DOE) is developing to support those disciplines. Not only do these learning technologies open up exciting opportunities for scientific discovery on exascale systems, they also appear poised to have important implications for the design and use of exascale computers themselves, including high-performance computing (HPC) for ML and ML for HPC. The overarching goal of the ExaLearn co-design project is to provide exascale ML software for use by Exascale Computing Project (ECP) applications, other ECP co-design centers, and DOE experimental facilities and leadership class computing facilities.

Keywords

Machine learning, exascale computing, reinforcement learning, active learning, HPC for ML, ML for HPC

Introduction

Working closely with existing Exascale Computing Project (ECP) applications (Alexander et al. 2020), the Co-design Center for Exascale Machine Learning Technologies, known as ExaLearn, has undertaken a focused co-design process targeting learning methods that are common across these applications. These methods include deep neural networks (DNNs) of various types (e.g., recurrent neural networks (RNNs), convolutional neural networks (CNNs), and generative adversarial networks (GANs)), kernel and tensor methods, decision trees, ensemble methods, graphical models, and reinforcement learning methods. To understand and guide trade-offs in the development of exascale systems, applications, and software frameworks, especially given

[1] Argonne National Laboratory, Lemont, IL, USA
[2] Brookhaven National Laboratory, Upton, NY, USA
[5] Los Alamos National Laboratory, Los Alamos, NM, USA
[3] Oak Ridge National Laboratory, Oak Ridge, TN, USA
[4] Pacific Northwest National Laboratory, Richland, WA, USA
[6] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[7] Sandia National Laboratories, Albuquerque, NM, USA
[8] Lawrence Livermore National Laboratories, Livermore, CA, USA
[9] Tokyo Institute of Technology, Tokyo, Japan
[10] RIKEN Center for Computational Science, Kobe, Japan
[11] ETH Zurich, Zurich, Switzerland
[12] University of Oregon, Eugene, OR, USA

Corresponding author:
Francis J. Alexander, Brookhaven National Laboratory, PO Box 5000, Building 725, Upton, NY 11973-5000, USA
Email: falexander@bnl.gov

constraints related to application development costs, application fidelity, performance portability, scalability, and power efficiency, the ExaLearn team has engaged directly with developers of ECP hardware, system software, programming models, learning algorithms, and applications.

The team's deliberate focus on verification and validation and uncertainty quantification with a solid determination of generalization errors is key to success. Using exascale machine learning (ML) to improve the efficiency and effectiveness of Department of Energy (DOE) computing resources and experimental facilities provides a unifying principle.

ExaLearn's goals are fourfold: 1) reduce the development risk of ML software for ECP application teams by investigating crucial performance trade-offs related to implementation and application of learning methods in science and engineering; 2) produce high-performance implementations of learning methods; 3) enable simple and efficient integration of those methods with applications; and 4) contribute to the co-design of effective exascale applications, software, and hardware.

ExaLearn is producing a **Software Toolset** that is applicable to multiple problems within the DOE mission space and has a line-of-sight to exascale computing, i.e., it uses exascale platforms directly or provides essential components to an exascale workflow; does not replicate capabilities easily obtainable from existing, widely available packages; builds in domain knowledge wherever possible; physics-based ML and artificial intelligence (AI) are recurring themes; uncertainty is quantified in a predictive manner; and is interpretable, reproducible, and based on well-grounded mathematical methods.

For its overall focus, ExaLearn has selected four classes of learning problems, namely using ML for development of **surrogate models**, **inverse solvers**, **control policies**, and **design strategies**. Each class (detailed herein) has been demonstrated on a different application area.

Surrogate Models

Problem Definition: Advances in computational algorithms and hardware over the past three decades have afforded scientists the tools to simulate nature over a tremendous range of scales. From computing the history of the cosmos and the explosion of supernovae to the evolution of Earth's climate and the properties of materials and subatomic particles, simulations now play a role in almost all branches of science. Despite having access to tremendous computational resources, scientists still are unable to explore all possible theories or simulate phenomena at the sub-grid scale. ML presents a unique opportunity for bridging this gap in the form of surrogate models. However, much work needs to be done to determine if these surrogates are viable, unbiased replacements for such simulations and, importantly, useful for widespread scientific use.

Answering critical questions in cosmology about the nature of cold dark matter, dark energy, and the inflation of the early universe requires relating observations to simulations of virtual universes with different cosmological parameters to beat down the systematic uncertainties found in the observational data. Observationally, astrophysicists routinely make statistical measurements at the few percent level, and almost all cosmology experiments are dominated by systematic uncertainties. Currently, each virtual universe requires an extremely computationally expensive simulation. The ability to emulate these simulations in high-fidelity and in a fraction of the typical computational time would boost the ability to understand the fundamental nature of the universe. The application of deep learning techniques to generative modeling is renewing interest in using high-dimensional density estimators as computationally inexpensive emulators of fully fledged simulations. These generative surrogate models have the potential to initiate a dramatic shift in the field of scientific simulations, cosmology being an excellent case study. For that shift to happen, we must study the performance of these generators in the precision regime needed for science applications, as well as effectively handle the multiple petabytes (PBs) of training data on future exascale systems.

Training Data: Cosmology data sets are particularly challenging to manage in an ML framework as they are quite large, record multiple gigabytes (GBs) per snapshot in time, and evolve as a function of time over the 13-billion-year history of the universe. Coupled with the fact that it requires thousands of these data sets to perform successful training, it leads to distributed ML training that can take advantage of a large high-performance computing (HPC) system. Our training data are based on more than 10,000 n-body simulations using the codes MUSIC, to initialize the distribution of matter, and pyCOLA, to advance particles throughout the history of the universe (Hahn and Abel 2011; Tassev et al. 2013, 2015). The output of these simulations is then binned into a three-dimensional (3D) histogram of particle counts in a cube of size 512x512x512 at four select redshifts (points in time). Here, four cosmological parameters are varied by 30% of their current, best measured values: Ω_M, H_0, N_{spec}, and σ_8. The data are available at `https://portal.nersc.gov/project/m3363` and

via the ExaLearn data portal at `petreldata.net/exalearn/projects/cosmoflow`. The simulated data, in Figure 1, represent a 3D map of density as a function of time and highlight the filaments and voids in the cosmic web that composes the universe.

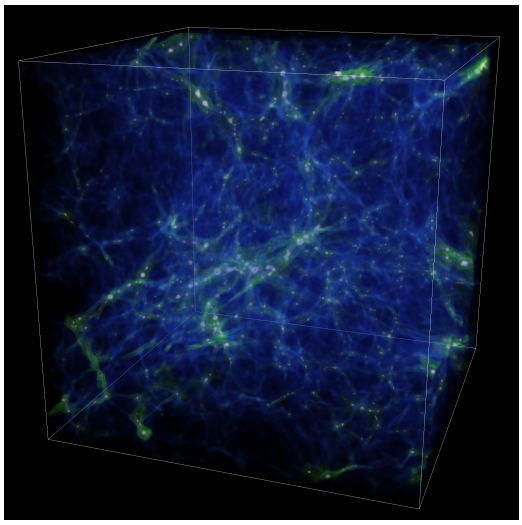

Figure 1. A small snapshot of one cosmology simulation presented in Walther et al. (2019). The density of both dark and baryonic matter in the universe is shown at roughly 2 billion years after the Big Bang. At this time, the universe's structure has clearly formed, showing clusters of galaxies (red), cosmic web (blue), and cosmic voids (black).

Method: The ExaGAN code has implementations in TensorFlow (TF) and Keras for two-dimensional (2D) slices of the aforementioned cubes, while a 3D version is being developed using the Livermore Big Artificial Neural Network (LBANN). Two main summary statistics provide useful metrics regarding the quality of generated samples: the pixel intensity histogram and power spectrum (the relationship between the distance from every pixel to all other pixels, typically employed by calculating a fast Fourier transform (FFT)). The basic model is a simple DCGAN (Goodfellow 2016), essentially identical to that of the original CosmoGAN paper (Mustafa et al. 2019). Early tests have shown that the GAN had trouble capturing the tail of the pixel intensity distribution (i.e., the pixels with very large values), which is heavily squashed by the transformation $s(x)$ used to normalize the data. Pixels with lower values, which constitute the majority of the structures in filaments and voids, were reasonably captured. To ensure accuracy and useful gradients at both regimes of the data domain, we have developed a technique to augment the DCGAN model called *multi-channel-rescaling* (hereafter MCR).

The MCR technique simply concatenates a second image channel to the generator output, where the second channel is a different normalization of the data in the generated sample. The discriminator then is trained with the 2-channel images (the same transformation is applied to the training data). The normalization for the second channel, which appears to work best, simply is a linear scaling of the data, scaled down by some large number (e.g., 1000) and fed through a *tanh* function to improve numerical stability. This method was able to improve the quality and statistical validity of the output samples.

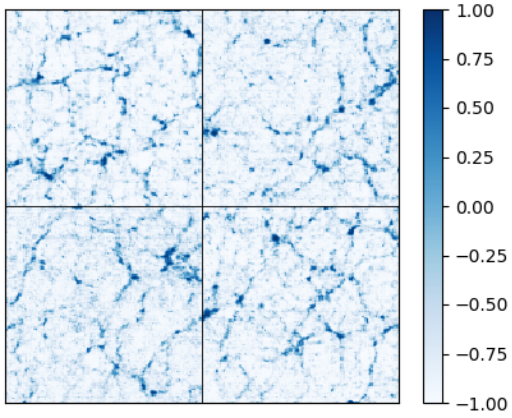

Figure 2. Four GAN-generated 2D slices of cosmological simulations.

While results have been quite good (Figure 2), the following steps have since greatly improved both convergence and the statistics on the resultant maps.

Spectral constraints has been the most promising addition to ExaGAN. The basic idea is to add a constraint to the generator loss function, which penalizes it when the generated samples have a power spectrum that differs from the mean power spectrum of the real data. The key to enforcing this constraint is computing backpropagation-enabled FFTs on the output of the generator. This is a physically motivated constraint rather than a more traditional regularizer found in most ML literature.

Gradient regularization, relatively standard in training GANs, involves penalizing the magnitude of the gradients from the discriminator to remain bounded. Two common variants are *R1 regularization*, which penalizes the discriminator gradients to remain small when processing the real data samples, and *WGAN-GP* (Wasserstein Gradient Penalty Loss), used in conjunction with the WGAN loss function (Arjovsky et al. 2017). Both help to stabilize the training process.

Feature matching involves penalizing the generator to match the statistics of the intermediate feature maps

from the discriminator on the real data, which helps stabilize the training and prevent mode collapse.

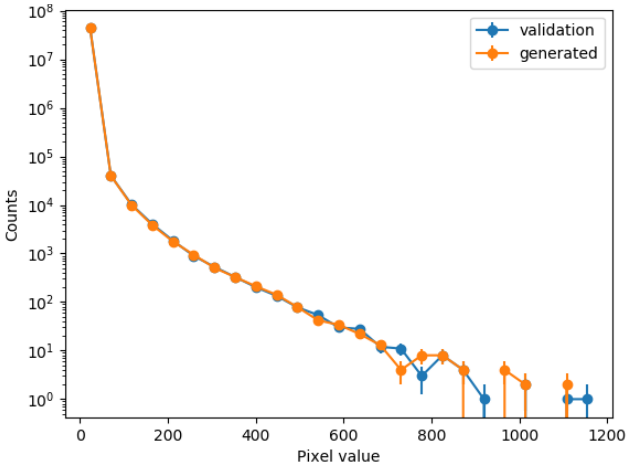

Figure 3. Logarithmic histograms of pixel intensities from the GAN-generated and validation cosmology data sets show an excellent match. Higher-order statistics, such as the power spectrum, also are in agreement.

Remaining Challenges: While 2D maps are quite successful (Figure 3), to push on the inherent statistics and fully measure potential biases in the generated images will require moving to 3D. Because the training on the cosmology data has both high computational and memory costs, we will employ the LBANN code, which already has shown great scaling on 3D cosmology data sets with the CosmoFlow CNN regression-based code. LBANN can spatially partition the training over many graphics processing unit (GPU)-accelerated HPC nodes, enabling the traditional robust scaling that other HPC applications enjoy, i.e., *accelerated time to solution without a compromise in the quality of the learned model* (Van Essen et al. 2015). For the CosmoFlow problem, LBANN is able to achieve an order-of-magnitude improvement in prediction quality using the full 3D data sets in training while significantly reducing training time by exploiting a much larger-scale system (Oyama et al. 2020). The GAN-based surrogate models should be able to take advantage of LBANN to an even greater degree.

Inverse Solvers

Problem Definition: Inverse problems emerge in a variety of application domains of interest within the DOE Office of Science, including neutron scattering data analysis. Here, a data-driven ML approach (Figure 4) is evaluated as a replacement for traditional model-driven, computationally expensive loop refinement methods used to determine material structure from their neutron scattering signatures. Ascertaining the internal structures of target samples requires determination of their crystallographic symmetry classes, as well as structural parameters such as cell lengths and angles. The efficacy of a class-conditional ML pipeline with a shallow classifier to predict the crystallographic symmetry group, followed by a shallow regressor to predict the cell lengths/angles, has been presented in Garcia-Cardona et al. (2019). Here, we illustrate a preliminary evaluation of two class-conditional DNN models in addition to three integrated ML models that predict both the symmetry and cell parameters in a single predictive task. The remaining section discusses training data generation, data preprocessing steps, model descriptions, and preliminary performance results. The methodologies presented can be generalized to X-ray crystallography data analysis with minor modifications.

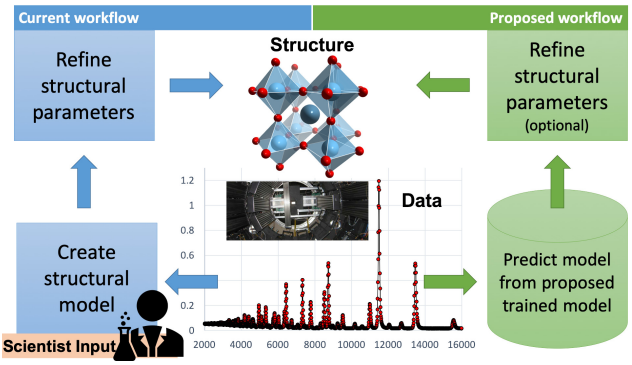

Figure 4. Predictions of material structure by ExaLearn ML models to replace time-consuming, model-driven neutron scattering data analysis workflows.

Training Data: For this evaluation, a perovskite called *barium titanate* ($BaTiO_3$) is used as the target material sample. Without doping, $BaTiO_3$ exists only in three of the 14 possible crystallographic symmetry groups. As such, labeled training data sets only for the tetragonal, trigonal, and cubic crystallographic symmetry classes are created (Table 1). Then, the GSAS-II software tool (Toby and Von Dreele 2013) is used to generate the diffraction pattern, X, for every combination of the lattice parameters, collectively denoted by Y, where Y represents the set $\{a, b, c, \alpha, \beta, \gamma\}$ of unit cell lengths (a, b, c) and angles (α, β, γ) uniformly sampled from the six-dimensional parameter space. Each diffraction pattern X is a set of 2807 2-tuples $(x,\ I(x))$, where x is the time of flight (ToF) sampled in the range [1,360μs, 18,919μs] and $I(x)$ is the corresponding GSAS-generated scattering profile. To maintain consistency with the NOMAD-generated experimental data against which the model

predictions are subsequently validated, a parameter specification file corresponding to the NOMAD instrument is used for the GSAS-II tool.

Table 1. Training Data Set of Labeled Neutron Diffractions

Class	Parameters	Samples (n)	Size
Cubic (predict a)	$a = b = c$	1000	43 MB
Trigonal (predict a, α)	$a = b = c$ $\alpha = \beta = \gamma \neq 90°$	160 400	6.8 GB
Tetragonal (predict a, c)	$a = b \neq c$ $\alpha = \beta = \gamma = 90°$	47 719	2 GB

Note that the spaces of valid cell lengths and angles are determined by physics-driven constraint equations corresponding to each symmetry class. Table 1 lists the relations for the three symmetry classes in this study. For the cubic class, a was sampled in the range [3.5, 4.5]. For the trigonal class, a was sampled in the range [3.8, 4.2] with α in the ranges [60°, 89.8°] and [90.5°, 120°]. For the tetragonal class, a, c were sampled in the range [3.8, 4.2].

Data Preprocessing: Background signals in neutron detectors originate from varied sources (diffuse scattering, air scattering, detector readout noise, etc.) and need to be subtracted to improve the signal-to-noise ratio. A second-order Chebyshev polynomial of the first kind is used to model a NOMAD-specific background signal for each experimentally observed diffraction pattern independently. A signal threshold in the experimental Bragg profile is adjusted so that the area under the profile closely matches (differs by less than 10^{-4}) that under the Chebyshev polynomial. This polynomial then is subtracted from the original experimental signal. Preliminary results indicate this method is more robust to experimental conditions than previously used quantitative measures. In addition to the background corrections, the x-axis (ToF) also needs to be adjusted for better consistency between the simulated and experimentally collected Bragg profiles. For this, each $(x, I(x))$ pair in the experimental diffraction pattern is matched with the closest ToF from the simulated ToF (which is the same for all of the GSAS-II-generated diffraction patterns). Finally, the intensities, $I(x)$, for the experimental and generated Bragg profiles are both normalized to unity.

Models: The models evaluated here belong to two broad categories: class-conditional (**C**) and integrated (**I**) models. In class-conditional models, the overall prediction is carried out in a sequence of two independent learning tasks. In the first task, a classifier predicts the crystallographic symmetry, and, in the second, a regressor predicts the cell lengths/angles. On the other hand, integrated models are designed to predict the symmetry class and cell lengths/angles in a single ML task, often referred to as *multi-task models*. As noted, the number of symmetry classes in this study is restricted to three (cubic, tetragonal and trigonal) out of 14 possible in nature. Accordingly, the integrated models predict four outputs, namely the class label S (0: cubic, 1: tetragonal and 2: trigonal) the lattice parameter a, the lattice parameter c, and angle parameter α. These four predictions form the minimal set of parameters necessary to determine the structures of the three crystal symmetries studied (Table 1).

A one-dimensional (1D) CNN, consisting of two convolutional layers interleaved with two (max) pooling layers and two final fully connected layers (with ReLU and softmax activations, respectively), was designed, hypertuned, and trained on the GSAS-generated training data for the classification task of predicting the symmetry classes. This classifier forms the basis of three models evaluated here: two **C** models that use transfer learning and one **I** multi-task model.

In the first class-conditional transfer learned model, denoted by $\mathbf{C}_1$, the features learned from the classifier were fitted for the parameters a, c, and α by replacing the final softmax layer with a linear layer, the *ANN regressor*, and updating **only** the weights of this ANN regressor during training.

The second transfer learned model, denoted by $\mathbf{C}_2$, has the same architecture as $\mathbf{C}_1$ with the exception that the weights of all the layers are updated during backpropagation. The layers of $\mathbf{C}_2$, except the ANN regressor, are initialized with the corresponding weights from the fine-tuned classifier.

In the third model, an integrated model denoted by $\mathbf{I}_1$, common features are concurrently learned for the two tasks, i.e., classifier for determining symmetries and regression for the parameters a, c and α, in a single learning task. Because $\mathbf{C}_1$ and $\mathbf{C}_2$ are class-conditional models, three models are separately trained for the cubic, trigonal, and tetragonal symmetries with batch sizes of 512 over 500 epochs. For a fair comparison, the model $\mathbf{I}_1$ also is trained with the same batch size.

Another integrated model, denoted by $\mathbf{I}_2$, that does not use the classifier also has been studied. $\mathbf{I}_2$ is a deep learning model based on a 1D convolutional autoencoder (CAE) architecture whose latent representation is used as the input for a regression model with a dense layer configuration. Finally, a shallow integrated model, $\mathbf{I}_3$, that uses random forests also is evaluated.

Results: The background-free training data (Table 1) was partitioned into a training set (80%) and a testing set (20%) with each individual sample normalized so its minimum intensity value was 0

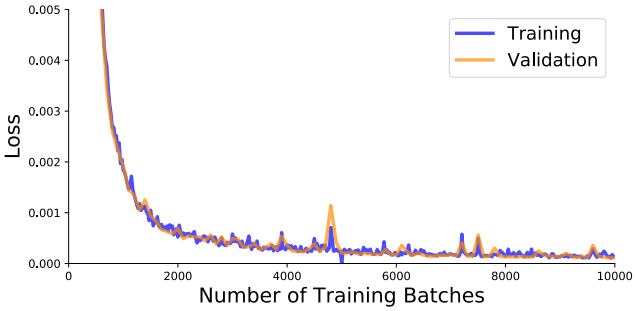

Figure 5. Training and validation loss of the classifier as a function of the number of training batches.

(via translation) and scaled so the maximum value was 1. Combined, there were 800/200 cubic examples (training/testing), 128K/32K tetragonal examples, and 38K/9.5K trigonal examples.

The classifier was trained using a balanced minibatch. For each minibatch, 30 samples were drawn uniformly at random (with replacement) from the training data for each class, yielding a minibatch of 90 examples (30 from each class). Stochastic gradient descent (SGD) with a fixed learning rate 0.001, weight decay of 0.005, and momentum of 0.9 was used for training on a total of 10 000 minibatches (Figure 5).

Table 2 summarizes the performances of the models described earlier. Models $\mathbf{C}_1$ and $\mathbf{C}_2$ report the mean squared error (MSE) over 32 080, 9544, and 200 samples of tetragonal, trigonal, and cubic samples, respectively. In this study, the integrated models, $\mathbf{I}_2$ and $\mathbf{I}_3$, were trained over 27.3K data samples (16% of training data set). The integrated models were found to outperform the class-conditional models. In this initial evaluation, the model $\mathbf{I}_2$ was found to perform the best. This model was used to predict the symmetry class and structural parameters of an experimentally collected Bragg profile from the NOMAD detector in the Spallation Neutron Source (SNS). The predicted structural parameters then were used as inputs to GSAS-II to compute the scattering pattern and compared with the experimental profile (Figure 6).

Table 2. MSE of the different models with scale factors $\dagger : \times 10^{-5}, \ddagger : \times 10^{-2}, * : \times 10^{-4}$.

	Trigonal		Tetragonal		Cubic
Model ↓	$a^\dagger$	$\alpha^\ddagger$	a^*	b^*	a^*
$\mathbf{C}_1$	3.55	1.660	1.950	8.14	4.71
$\mathbf{C}_2$	1.80	0.001	0.270	0.25	0.10
Model →	$\mathbf{I}_1$		$\mathbf{I}_2$		$\mathbf{I}_3$
$(S, a, c, \alpha)^\dagger$	1.90		1.66		12.4

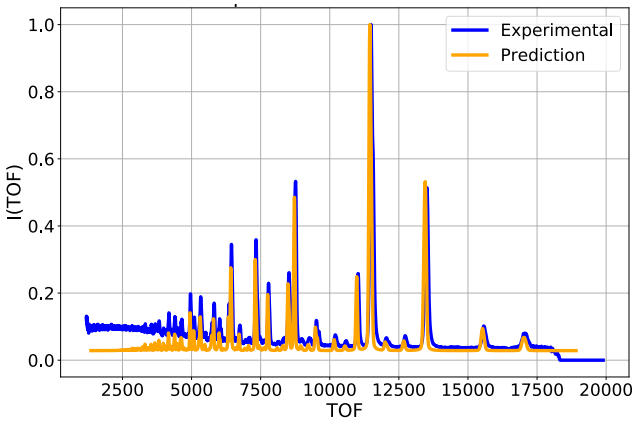

Figure 6. Comparison of prediction from the $\mathbf{I}_3$ model with the experimental profile.

Next steps: In this study, the models were tested on only three symmetry classes over a small sampling of the overall parameter space. Training these models to perform uniformly well across all symmetry classes and larger volumes of the parameter space quickly becomes a computationally challenging enterprise that requires extreme-scale computing resources. Designing unified models capable of accurately predicting the structural parameters belonging to any of the 14 symmetry classes in a single predictive task is the next goal.

Control Policies

Problem Definition: Control problems in everyday life can be found in game playing, as well as autonomous vehicles, robotics, and factory controls. Meanwhile, control problems in science can be found in simulations, experiments, and facility operations. Complex control problems have many different possible paths, numerous intermediate and target states, complicated trade-offs, conditional behavior, complex goal and subgoal relationships, nuances in the order of actions taken, and long-term rewards that may not be immediately obvious. These problems extend beyond straightforward controllers and optimization methods and require non-trivial approaches that can be computationally intensive.

Reinforcement learning (RL) (Sutton and Barto 1998) is a useful technique for solving complex control problems and is quite efficient for many that have been historically intractable. RL can be used on a control problem when mapped to a finite Markov decision process (MDP). Within a finite MDP framework, an agent can take actions in an environment and receive

rewards while attempting to reach a target state. This iterative, online, learning-by-doing approach is accomplished by recording interaction experiences and using them to create a policy. The agent consults the policy when determining the next actions, but action choices also can be random to reach new areas of the search space. Policies can be DNNs, which are useful when the number of possible actions is large and a fast policy is needed.

This section describes the challenges of using RL to solve control problems at scale, describes the scalable RL framework developed by the ExaLearn Control pillar team, and details some applications where the framework is used along with results.

Challenges for Control Problems at Scale: As scientific control problems become more complex, with many more actions and a highly complex policy network, it becomes important to support scalable RL because more training is needed for accurate control. This requires orchestration of many simultaneous tasks: agents choosing actions, learners incorporating experiences into a network, environments (simulations) executing agent actions, communication of experiences from agents to learners, and communication of updated models from learners to agents. All of this may need to take place alongside or within exascale simulations and/or data analytics. Furthermore, scientific applications of RL have environments that often are computationally intensive and require parallelization and acceleration to run quickly during online learning. Multi-rank environments then need to be integrated with existing applications.

Importance Weighted Actor-Learner Architecture (IMPALA) is one state-of-the-art distributed deep RL framework that can be scaled up to thousands of machines with training stability and data efficiency (Espeholt et al. 2018). Scalable, distributed RL infrastructures have been produced, notably the Ray framework (Moritz et al. 2017). Ray uses a distributed task-based parallel programming model and abstracts modular and reusable components of learning algorithms. However, for scientific applications running on exascale machines, we need something easily integrated into existing scalable, optimized application code, as well as the ECP software stack.

EXARL Software: ExaLearn is developing the Easily eXtendable Architecture for Reinforcement Learning (EXARL) framework to provide scientists and non-ML experts with a readily available, easily modifiable set of RL components that can run in a distributed and scalable fashion on exascale hardware. The framework has three core components: *environments*, *agents*, and *learning workflows*.

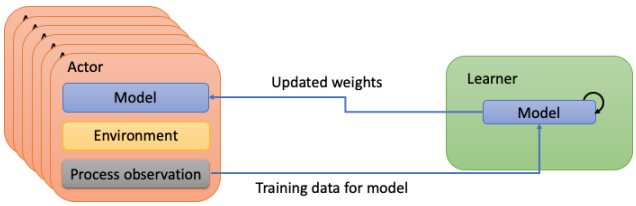

Figure 7. Overview of EXARL architecture. Each actor has a copy of the model, which is updated in every step and used to infer a new action given a state. The actor uses this action to compute the next environment state and runs the Bellman equation to generate updated data. These data are sent back to the learner for training the target model.

The environment component is an extension of the OpenAI gym framework that affords use of scientific environments and can integrate with existing exascale applications (e.g., LAMMPS or NWChem). In addition, it will accommodate traditional OpenAI gym environments to allow for studying the performance of new algorithms using benchmark environments. The agents are collections of RL algorithms with associated neural network architectures (e.g., DQN (deep-Q network) algorithm with multilayer perceptron (MLP) or long short-term memory (LSTM) architecture). The framework has registries for agents and environments that allow users to easily retrieve the desired components. Finally, the learning workflows are implementations of how the agents and environments interact with each other. Each component is easily customized to allow users and developers the ability to focus on their needs. For example, a domain expert can focus on developing an environment, while an algorithm expert can use an existing environment to study various algorithm and learning strategies. Each component has a default configuration setup that is easily steered using the CANDLE Supervisor application.

The architecture of EXARL is separated into learner and actors (shown in Figure 7). A simple round-robin scheduling scheme is used to distribute work from the learner to the actors. The learner consists of a target model that is trained using experiences collected by the actors. Each actor consists of a model replica, which receives the updated weights from the learner. This model is used to infer the next action given a state of the environment. The environment can be rendered/simulated to update the state using this action. In contrast to other architectures such as IMPALA (Espeholt et al. 2018) and SEED (Espeholt et al. 2019), each actor in EXARL independently

stores experiences and runs the Bellman equation to generate training data. These training data are sent back to the leaner (once enough data are collected). By locally running the Bellman equations in each actor in parallel, the load is equally distributed among all actor processes. The learner distributes work by parallelizing across episodes, and actors request work in a round-robin fashion. Each actor runs all of the steps in an episode to completion before requesting more work from the learner. This process is repeated until the learner gathers experiences from all episodes.

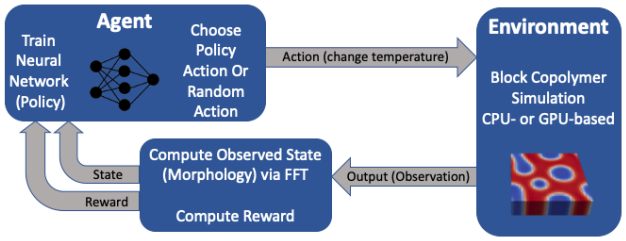

Figure 8. The problem of guiding block copolymer self-annealing is mapped to a finite MDP as an application of scalable RL.

Application-Block Copolymers: Consider, for example, the problem of controlling the self-assembly of block copolymers (Majewski and Yager 2016) via temperature change during experiments at light sources (Noack et al. 2019), (Noack et al. 2020). A block copolymer material typically begins in a disordered state and requires global or local heating to induce ordering. Materials may evolve toward generic equilibrium morphologies or become trapped in metastable state as ordering involves passing through multitudes of intermediate states in a complex, high-dimensional energy landscape (Nowak and Yager 2020). Target states (especially non-equilibrium states) can be quite difficult to reach, often requiring hundreds of experimental trials to get right because of many dead-end paths through the search space. We adopted this application to demonstrate the usefulness of RL and to exercise scalable RL (Figure 8). We mapped this problem to RL and have been able to show learning convergence for guiding the annealing to both equilibrium and non-equilibrium states.

Remaining Challenges and Next Steps: Future possible application areas include epidemiology (controlling policy for pandemic guidelines), combustion, and additive manufacturing. Future directions for the EXARL framework include changes that enable further scaling, such as using multiple learners, and to run multi-process environments.

Design Strategies

Problem Definition: Scientists representing various domains have embraced deep learning as an automated route for experiment and simulation design, often with the goal of producing molecular systems with optimized properties (Figure 9). Using deep learning or statistics for steering experimental campaigns, known as *Optimal Experimental Design* (OED) (Lookman et al. 2018) or *Active Learning*, relies on the combination of AI techniques, from generative modeling to uncertainty quantification, to form a cohesive application. One goal in ExaLearn is to develop illustrative examples of applying OED to steer HPC-based simulation campaigns and develop the knowledge base and tools needed for OED at the exascale.

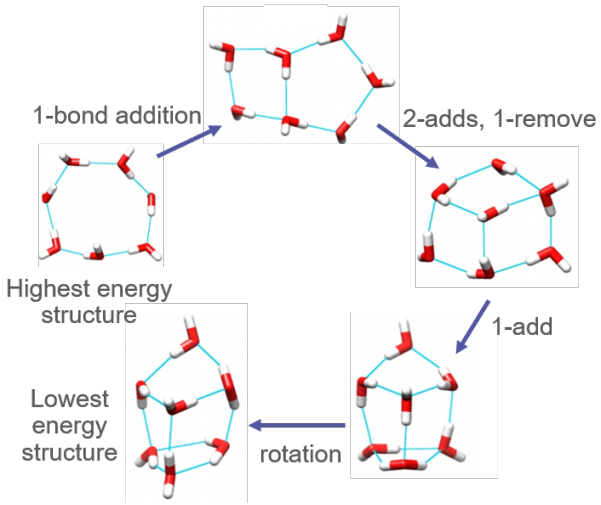

Figure 9. Step-by-step generation of a low-energy water cluster from a higher energy structure. The ExaLearn Design team aims to automate this design process by applying OED to steer HPC-based simulation campaigns that can, for example, produce a structure with optimal properties.

Motivation and Algorithmic Workflow: As illustrated by the CANDLE project's Supervisor application, highly parallel distributed optimization algorithms require near-continuous generation of new tasks to maintain high system utilization (Wozniak et al. 2018). Work by the Rocketsled team at Lawrence Berkeley National Laboratory (LBNL) has illustrated that the time required to generate new tasks for optimization increases steadily for large problems (Dunn et al. 2019), which could become a bottleneck and lead to worker starvation for highly parallel workflows.

The task generation bottleneck becomes an even larger concern when using advanced techniques, such

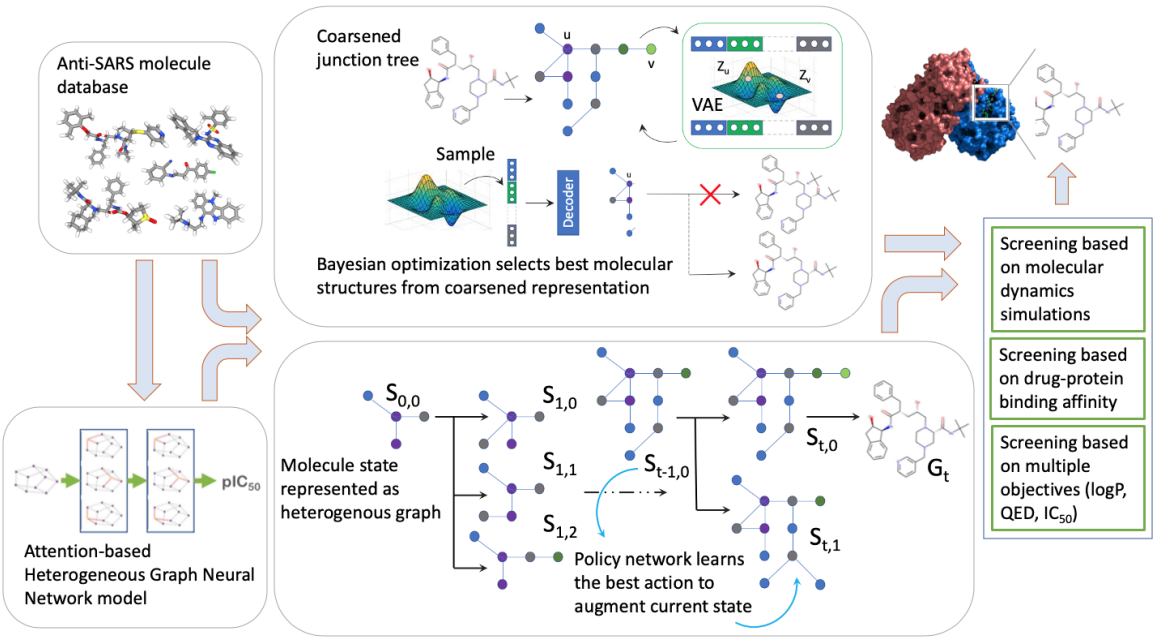

Figure 10. Depiction of a workflow developed for generating anti-SARS-CoV-2 drug candidates. An anti-SARS database was used to train a MPNN to predict pIC_{50} and a JT-VAE model. The trained MPNN was used as the scoring function in both JT-VAE (top) and DQN-based molecular generation (bottom). Candidate molecules were screened by pIC_{50} and validated by a Drug Target Binding Affinity classifier.

as RL for enumerating tasks or OED methods that simulate the effect of performing batch computations in parallel (e.g., the batch active search of Jiang et al. (2018)). As such, we examine several different OED science applications with varying trade-offs between simulation cost and experimental design. Two key goals are *scalability*, being able to handle the combinatorial explosion in the state-search space, and *interpretability*, generating design pathways that domain experts can reason about.

Target Science Applications: All three of the featured science applications are based on the determination of optimal graph structures. The similarity between tasks allows significant re-use of the deep learning techniques between problems, so work can focus on exploring computational aspects of each tasks. Herein, we describe the scientific problems and progress in developing applications for solving each design challenge.

Electrolyte Design. Electrolytes form the barrier between the charged electrodes of a battery and must simultaneously be impermeable to electrons current and allow the free movement of charged ions. The molecules that can provide such a barrier yet not interact with the highly reactive electrode materials, be non-flammable, and degrade slowly in harsh electrochemical conditions are rare. Finding such molecules has been a long-standing challenge problem in DOE (Cheng et al. 2015). ExaLearn is building a suite of ML models (Ward et al. 2019; Dandu et al. 2020) and quantum-chemistry workflow tools (Smith et al. 2020) to quickly estimate the outcome of quantum-chemistry simulations and evaluate the most valuable simulations on HPC.

Drug Molecule Design. The design of new drug compounds with target properties is a key area of research in generative modeling. We applied two generative models to design drug candidates targeting SARS-CoV-2: RL in the form of a DQN (Zhou et al. 2019)) and the junction-tree variational autoencoder (JT-VAE) technique (Jin et al. 2018)). Figure 10 depicts the workflow. The main difference between the two techniques is that the JT-VAE is trained on a database of small molecules with activity for SARS, while the DQN is trained through the RL search. Both techniques rely on a scoring function to steer generation. We have examined using different scoring functions based on molecular properties, such as logP and quantitative estimate of druglikeness (QED).

Notably, DQN was able to produce tens of thousands of optimized candidates in the time it took JT-VAE using Bayesian optimization to produce hundreds of candidates. Moreover, the DQN outperformed JT-VAE in generating higher scoring molecules. We attribute

the difference in optimization performance to JT-VAE implicitly sampling from a distribution of drug-like molecules and DQN having no such constraints. The candidate molecules generated by JT-VAE have consistently better druglikeness and synthesizability scores, even when those properties were not explicitly included in the scoring function.

The RL agent uses no information about the space of experimentally studied drug molecules during its training process and, accordingly, finds molecules far from it. JT-VAE implicitly uses the distribution of molecules in its training set to bias toward realistic molecules at the expense of generating novel candidates. The RL-based approach lacks such constraints and can optimize without even implicitly regarding synthesizability or any other characteristic not explicitly encoded in the scoring function.

Colmena Software: Future applications for OED on HPC systems will require significant concurrency between simulation workflows gathering new data and AI tasks that process the data to select new experiments. **Colmena** is designed to simplify the development of such applications by providing a simple programming interface for writing applications that manages concurrent AI and simulation tasks. Colmena is backed by the Parsl workflow engine (Babuji et al. 2019) that supports running tasks at all major DOE computing centers. Colmena and the applications ExaLearn is using to explore scaling OED to HPC are all open source and available on GitHub (`github.com/exalearn/colmena`).

Software: Molecular Graph Descriptors. We also have examined methods for post hoc interpretation of a neural network aimed at property prediction (Bilbrey et al. 2020). The interpretation relies on the computation of graph-based descriptors of molecular systems. The tools to compute these descriptors have been made open source and available on GitHub (`github.com/exalearn/molecular-graph-descriptors`).

Optimal Experimental Design Library. A central goal of this work is to produce a software library for OED (Pronzato 2008). This library will package a menu of interchangeable OED algorithms, parallelized for HPC environments, which may be chosen by a user prior to runtime. This is needed to handle complex OED problems involving many candidate experiments and independent forward solver evaluations.

For context, OED is a field that combines robust optimization with Bayesian inference. The aim is to design a system that is optimal with respect to a given goal on average across some uncertain parameters. Experiments may be conducted to reveal with greater statistical accuracy/precision what the true values of the uncertain parameters are, but these experiments are costly to perform. Thus, the goal is to select the most highly informative experiments with respect to the engineering goal. Mean Objective Cost of Uncertainty (MOCU) (Dehghannasiri et al. 2017; Yoon et al. 2013) is one algorithm that implements OED. Others include entropy-based exploration strategies, active learning, and knowledge gradient (Frazier et al. 2008; Settles 2009).

Performance and Scaling

To support the Surrogate application pillar, ExaLearn has developed a new capability for spatially partitioning the training of 3D CNNs that was developed by the scalability and performance cross-cut team. This enables training on very large data cubes that would otherwise be infeasible due to memory limitations. Training on full-size data samples enables models to learn longer-range effects than would otherwise be possible. Due to the huge data size, we also have developed a new data ingestion pipeline that leverages parallel input/output (I/O) through HDF5 and MPI-IO, as well as an in-memory distributed data store to reduce I/O overheads.

To demonstrate its capability, we studied the impact on the CosmoFlow 3D CNN and its associated data set as described in the Surrogate Models section. We studied the impact of data size and neural network architecture choices for this problem and present performance and scaling results, including training a single model with up to 512 V100 GPUs. Furthermore, we demonstrate an order-of-magnitude improvement in the prediction quality of the CosmoFlow network while significantly reducing training time. This work demonstrates the benefits of training on large, high-resolution data for surrogate models and techniques for overcoming the associated challenges, which can be applied to many scientific applications.

We use LBANN (Van Essen et al. 2015) to implement our approach as it has already demonstrated good scalability for 2D spatial partitioning (Dryden et al. 2019). We extended LBANN's spatial parallelism to efficiently support 3D data in hybrid-parallel training. Previous CosmoFlow network efforts have been limited by memory to training with windowed data samples of size 128^3. Our work increases this by a factor of 64 to 512^3.

Figure 11 shows the strong scaling performance of the CosmoFlow network with the 512^3 data set. We use global mini-batch sizes (N) of 1, 2, 4, 16 and 64 and split the network in the depth dimension. We run the framework for four epochs with a 128-sample

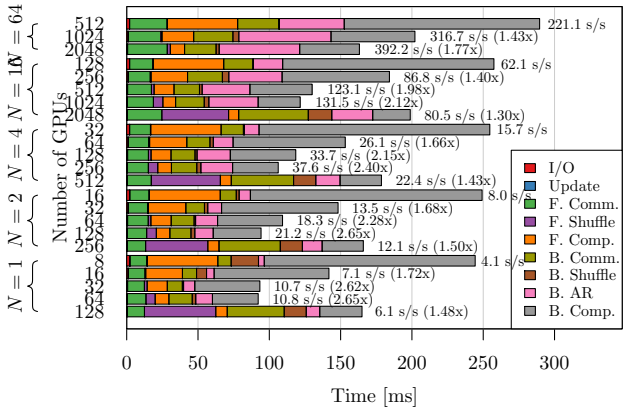

Figure 11. Strong scaling of CosmoFlow. "F" and "B" are forward and backward passes, respectively. N is the mini-batch size. Bars are annotated with throughput (samples/s) and speedup relative to the minimum setting with the same N.

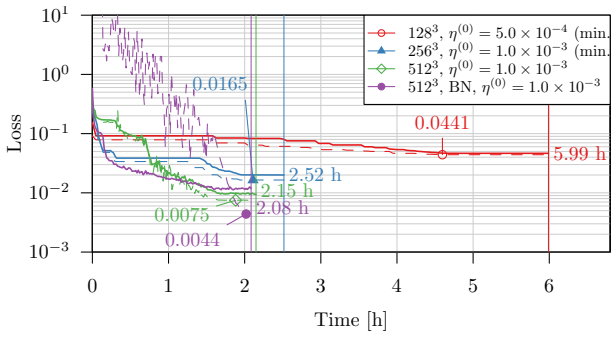

Figure 12. Training/validation losses (solid/dashed lines, respectively) and the smallest validation losses (points) of the CosmoFlow network with four different configurations. For 128^3 and 256^3, we show the minimum loss values at each point in time for visibility.

subset of the data set (if the mini-batch size is smaller than 128) or the full data set and show the median iteration time except for the first epoch. As shown in the figure, when the mini-batch size, N, is 16 and 64, we achieve speedups of 1.98x with 512 GPUs (128 nodes) compared to 128 GPUs (32 nodes) and 1.77x with 2048 GPUs (512 nodes) compared to 512 GPUs (128 nodes), respectively.

Figure 12 shows training results of the CosmoFlow network with the full-resolution data set (512^3) and split versions (128^3 and 256^3). We swept the initial learning rate from 10^{-4} to 10^{-2} logarithmically and show the results with the best. We train for 130 epochs with a mini-batch size of 64 in every configuration and use the 4-way partitioning (256 GPUs in total) for the networks without batch normalization layers, or 8-way (512 GPUs in total) for networks with batch normalization, due to the increased memory requirements. To account for training variance, we show the median result of five trials with different initial random seeds.

We observe that the test loss decreases significantly as we increase the data set size to 0.0169 MSE with 256^3 and 0.00727 MSE with 512^3 data. Adding batch normalization improves this result to 0.00445 MSE, achieving an order-of-magnitude improvement compared to the baseline 128^3 data. At the same time, we get of speedup from 128^3 to 512^3 with the same number of GPUs and same mini-batch size. This result implies that the CNN can be trained with the same computing resources and data set size but with a smaller mini-batch and small overheads. This introduces an opportunity to keep mini-batch sizes fixed and strong-scale onto more GPUs for speedup. Overall, the capabilities developed for scaling model training and improving model quality in ExaLearn will enable new generations of data-driven surrogate models for an expanding number of scientific applications.

Uncertainty Quantification

While the potential for ML technologies to revolutionize computational science is driving modeling innovations across the DOE landscape, the need for robust interrogation of these modeling approaches in terms of errors, biases, and information quality is becoming more urgent to establish required credibility. The translation of parametric and model-form uncertainty quantification techniques, originally designed to address these issues in the classical hypothesis-driven modeling setting, into the data-driven and reduced order modeling paradigm afforded by ML techniques presents a significant challenge. Probabilistic ML methods that seek to elevate the training of neural network models into a setting where uncertainties can be propagated through models, e.g., Bayesian neural networks, go some way in achieving this, albeit with the accompanying penalty of greatly increased training expense.

ExaLearn has explored using approximate inference techniques to learn posterior probabilities on the parameters of ML models in the context of combustion modeling. In particular, submodels that feed into the larger exascale simulations are targets for surrogate modeling to replace repeated evaluations of expensive kernels or queries from precomputed tabulations of submodel outputs.

Figure 13 shows the results of training a neural network using the Bayesian interpretation of

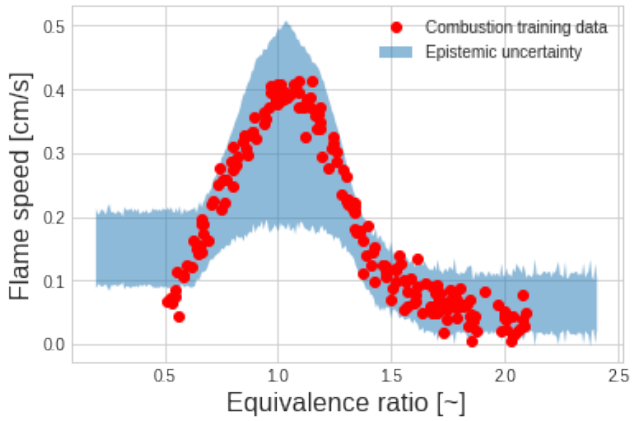

Figure 13. Neural network surrogate model for combustion submodel predictions, trained using dropout to estimate epistemic uncertainty in model outputs (shown here at the two standard deviation level).

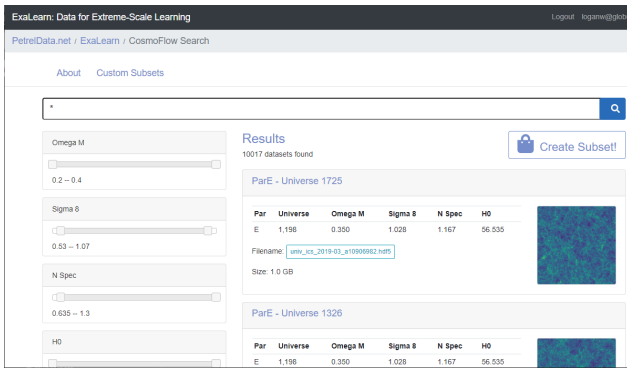

Figure 14. CosmoFlow data accessible via PetrelData.net are indexed by key simulation parameters. All 1+ TB are accessible via Globus (Chard et al. 2016).

dropout (Gal and Ghahramani 2016) with data from combustion flame speed models and outputs polluted by noise as a quantity of interest. The confidence interval on these predictions delivered by the Monte Carlo dropout predictions enables a framework for evaluating different neural network architectures, required volumes of training data, and accuracy under extrapolation with respect to nominal predictions and their variance—as well as the consequences of combining model and experimental data for the purposed of constructing holistic models.

More elaborate variational techniques for approximating the Bayesian posterior probability of neural network parameters also have been considered for these combustion submodels. These techniques have been compared against fully Bayesian training using Hamiltonian Monte Carlo sampling to interrogate the impact of the missing correlation information ignored by the approximate techniques. With this machinery, we are extending the combustion system analysis to the reduced-order modeling setting needed for simulations of detailed chemical processes at exascale by studying the systematics associated with constructing surrogates for reduced representations of detailed quantities of interest using projection-based compression of detailed model state, using principle component analysis (Echekki and Mirgolbabaei 2015) and other manifold learning techniques.

Software, Frameworks, and Infrastructure

Using AI applications on HPC systems provides a set of software challenges as yet unseen with conventional HPC applications. In particular, the need to store and distribute training data for AI+HPC applications and the rapidly evolving nature of "code" and computational environments require a redesign of the infrastructure that supports easily exchanging ML components in an HPC environment. ExaLearn's Software thrust focuses on paving the way for AI on exascale systems by studying and solving these infrastructure challenges.

Infrastructure-Sharing Training Data: Beyond the source code, model architecture, and model weights for an AI application, the training data for an application must be preserved to allow for future improvements. Algorithmic improvements leading to more accurate, faster, or more robust ML models cannot benefit from AI+HPC codes without the ability to retrain the core AI application components. Upgrading to better algorithms will never be as simple as linking to a new training library. We have developed a data repository hosted on Petrel (Allcock et al. 2019), a multi-PB data store at the Argonne Leadership Computing Facility, that forms a living archive of training data for key DOE AI applications. As shown in Figure 14, CosmoFlow (Mathuriya et al. 2018) data are available at `petreldata.net/exalearn/projects/cosmoflow` for ready use by the community.

The ExaLearn data infrastructure builds on the same Globus services that back the Materials Data Facility (MDF) (Blaiszik et al. 2019), DLHub (Chard et al. 2019; Li et al. 2020), and other DOE-affiliated open data projects. The web interface for the data library, PetrelData.net, is backed by a Django web service that provides an easily customizable route for hosting new open data projects with many needed features. PetrelData.net provides a search index for each project and, where needed, controls access to search index metadata or the data itself by using authentication services provided through Globus (Chard et al. 2016). It also provides an application

programming interface (API) for querying data and supports Globus transfers from a Python SDK. In total, the PetrelData.net service allows for data from exascale AI to be made accessible easily to both humans and software and will serve as the foundation for future reproducibility and innovations in AI+HPC research.

Infrastructure-DLHub: The flexible nature at the core of the utility of ML for scientific computing leads to significant challenges in deploying it on HPC systems. The source code of a scientific code's AI components (i.e., the weights) constantly change as the model is (re)trained. Furthermore, the rapid evolution of the libraries used to execute the AI components provides development challenges. DLHub offers a route to manage the unstable development environment associated with AI tools rather than relying on meticulous scientists to properly document versions of weights, library dependencies, and any "glue code" needed to transform data structures into forms compatible with AI frameworks.

DLHub encapsulates AI tools into "servables" that capture the computational environment needed to execute the servable along with a clear definition of the inputs and outputs for the component. Servables are available in a registry as Docker containers that can be shipped as complete units and integrated into scientific codes. ExaLearn is working closely with the development of DLHub and the project building the backend for DLHub, funcX (Chard et al. 2020), to create a development environment for creating and maintaining AI components of scientific codes. Our vision is that DLHub and PetrelData will provide an environment that leads to a new generation of scientific applications that rely on both AI and conventional, physics-based computation to optimize performance on the heterogeneous architecture of exascale HPC.

Input/Output

The advent of exascale systems will provide unprecedented capabilities for performing computations. These systems will be used to run increasingly more complex simulations at larger resolutions, as well as help expand the use of ML techniques for scientific discovery. In addition to the challenges of effectively using an exascale system, the widening gap between computation and I/O rates makes it more difficult to save simulation outputs to disk for offline analysis (Foster et al. 2017). The costs of moving data either on or off of the system are a growing challenge to the community. This imbalance between I/O and compute has resulted in challenges to the simulation sciences community in saving the results of a calculation. For the ML community, these challenges exist where large training data are needed to train a model. Data challenges for ML can result from the size of the individual training data sets, as well as the large number of training data sets. Thousands of modestly sized files can result in I/O issues on a supercomputer, which are optimized for smaller numbers of large files.

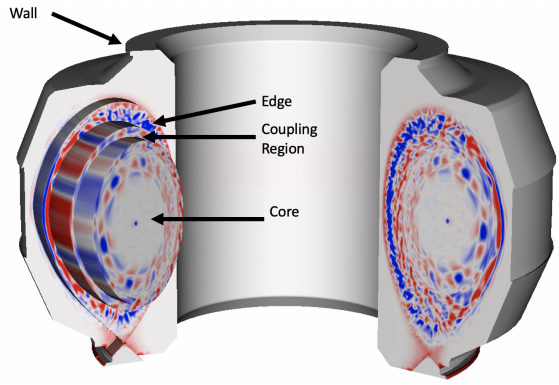

Figure 15. Fusion tokamak showing edge, core, and coupling region. Physics in the core and edge regions are solved with different codes. Spatial coupling between the two codes takes place in the coupling region.

In ExaLearn, we have developed an ML-enabled technique for reducing data movement costs in a coupled plasma physics computation being performed by the WDMApp project. WDMApp aims to deliver a multi-physics simulation that is coupled together at a first-principles level. The project's long-term goal is to provide all physics needed to understand and predict the performance of ITER and future fusion devices. WDMApp's current focus is the spatial coupling of two gyrokinetic codes—one for the interior, or core, of the plasma and the other for the edge region of the plasma (Figure 15). To accomplish this coupling, the particle distribution in each cell must be exchanged between the two codes. For ITER-sized simulations, moving this large amount of data will slow down the simulation. To avoid this, a VAE called *VAPOR* has been developed for the particle distribution data. Instead of transferring large amounts of data between the two codes, we transfer the model parameters, which results in an effective data compression rate of up to 60×. The reconstruction error can vary depending on the warm dense matter (WDM) physics (degree of turbulence, spatio-temporal resolutions, etc.). With a single step and medium-resolution grid data, we achieved an average of 10% root-mean-square training error. Currently, we are focused on developing methods for users to control the error and investigating how reconstruction errors can affect WDM physics. This

work also connects to the ECP Co-design center for Online Data Analysis and Reduction (CODAR) (Foster et al. 2020).

Proxy Applications

The performance of production applications, especially at exascale, relies on a complex combination of hardware architectures, runtime environments, compilers, and algorithmic choices. Proxy applications, small, simplified codes that are representative of larger applications, serve as models for performance-critical computations in larger applications and represent a compromise between the simplicity of kernel benchmarks and the complexity of full applications. Proxy applications serve as important driving forces in the architecture-system-application co-design efforts to ensure good performance for real applications on modern supercomputing systems.

There have been many proxy applications developed to represent scientific computing/computational simulation-type applications, for example, Mantevo (Heroux et al. 2009) and the ECP Proxy Application Suite (Richards et al. 2020). However, few proxy applications represent the workloads of data analysis or ML applications. To address this shortcoming and better serve as a focal point for exascale learning technology interactions with the ECP PathForward vendors, the ExaLearn co-design project has been working on developing proxy applications representing key ML areas.

One such proxy application is miniGAN (Ellis and Rajamanickam 2020), a GAN proxy application that has been developed as part of ExaLearn and released through the ECP Proxy Application Suite. GANs (Goodfellow et al. 2014; Radford et al. 2016; Creswell et al. 2018) are DNNs that simultaneously train two models: a generator **G** and a discriminator **D**. As the network trains, **G** produces increasingly accurate synthetic data, while **D** attempts to distinguish the synthetic data from the original training data. Important for miniGAN's use as a proxy application, GANs test a greater variety of layer types and training conditions than standard convolutional or feedforward neural networks.

Relating to specific ECP applications, miniGAN aims to be a proxy application for related machine applications in cosmology, such as CosmoFlow (Mathuriya et al. 2018) and ExaGAN (Mustafa et al. 2017), and in wind energy, such as ExaWind (Sprague et al. 2020). miniGAN models the performance for training generator and discriminator networks. The GAN's generator and discriminator generate plausible 2D/3D maps and identify fake maps, respectively. miniGAN is built on top of the PyTorch (Paszke et al. 2017) and Horovod (Sergeev and Balso 2018) packages and has been developed so that optimized mathematical kernels (e.g., kernels provided by Kokkos Kernels or vendor libraries) can be plugged into to the proxy application to explore potential performance improvements. miniGAN has been released as open-source software available through the ECP Proxy Application website (`proxyapps.exascaleproject.org/ecp-proxy-apps-suite/`) and GitHub (`github.com/SandiaMLMiniApps/miniGAN`). A generator is provided to generate a data set (series of images) that are inputs to the proxy application.

In addition to miniGAN, the ExaLearn team (Control) has developed a microbenchmarking suite that will be incorporated into the ECP Proxy Application project in the near future. This proxy application uses the simple CartPole ((Brockman et al. 2016)) environment in the EXARL framework and allows scaling of workers and environments. This is useful for checking MPI communications, as well as CPU/GPU utilization as the application scales.

Conclusion

Developments in AI and ML are proceeding at lightning speed. The investment in research, development, and deployment of AI/ML methods across the globe has experienced exponential growth. In response, the ExaLearn team has elected to focus on four ML pillars—surrogates, control, inverse problems, and design—with Exascale DOE applications in mind. While we have, thus far, targeted each pillar on a single representative problem, the end goal is to **demonstrate the integration of all four pillars toward a solution in one application area that requires all four types of learning**. DOE applications that can benefit from such an integration abound—from tokamak fusion to design of combustion engines and wind farms.

Acknowledgments

We thank the ECP leadership and our ECP colleagues from the many other projects with whom we have interacted throughout the ExaLearn project.

Funding

This research is supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of U.S. Department of Energy Office of Science and the National Nuclear Security Administration. A portion of the research was supported by JSPS KAKENHI Grant Number JP18J22858, Japan. The research has used resources

of the Argonne and Oak Ridge Leadership Computing Facilities, Livermore Computing Facitily, and Energy Research Scientific Computing Center (NERSC) DOE Office of Science User Facilities supported under Contracts DE-AC02-06CH11357, DE-AC05-00OR22725 and DE-AC52-07NA27344 (LLNL-JRNL-XXXXXX), DE-AC02-05CH11231, respectively.

References

Alexander F, Almgren A, Bell J, Bhattacharjee A, Chen J, Colella P, Daniel D, DeSlippe J, Diachin L, Draeger E, Dubey A, Dunning T, Evans T, Foster I, Francois M, Germann T, Gordon M, Habib S, Halappanavar M, Hamilton S, Hart W, Huang Z, Hungerford A, Kasen D, Kent PRC, Kolev T, Kothe DB, Kronfeld A, Luo Y, Mackenzie P, McCallen D, Messer B, Mniszewski S, Oehmen C, Perazzo A, Perez D, Richards D, Rider WJ, Rieben R, Roche K, Siegel A, Sprague M, Steefel C, Stevens R, Syamlal M, Taylor M, Turner J, Vay JL, Voter AF, Windus TL and Yelick K (2020) Exascale applications: Skin in the game. *Philosophical Transactions of the Royal Society A* 378(2166): 20190056.

Allcock WE, Allen BS, Ananthakrishnan R, Blaiszik B, Chard K, Chard R, Foster I, Lacinski L, Papka ME and Wagner R (2019) Petrel: A programmatically accessible research data service. In: *Practice and Experience in Advanced Research Computing*. pp. 1–7.

Arjovsky M, Chintala S and Bottou L (2017) Wasserstein GAN. *arXiv e-prints* : arXiv:1701.07875.

Babuji Y, Woodard A, Li Z, Katz DS, Clifford B, Kumar R, Lacinski L, Chard R, Wozniak JM, Foster I, Wilde M and Chard K (2019) Parsl: Pervasive parallel programming in Python. In: *28th International Symposium on High-Performance Parallel and Distributed Computing*. ACM. DOI:10.1145/3307681.3325400.

Bilbrey JA, Heindel JP, Schram M, Bandyopadhyay P, Xantheas SS and Choudhury S (2020) A look inside the black box: Using graph-theoretical descriptors to interpret a continuous-filter convolutional neural network (CF-CNN) trained on the global and local minimum energy structures of neutral water clusters. *The Journal of Chemical Physics* 153(2): 024302. DOI:10.1063/5.0009933.

Blaiszik B, Ward L, Schwarting M, Gaff J, Chard R, Pike D, Chard K and Foster I (2019) A data ecosystem to support machine learning in materials science. *MRS Communications* 9(4): 1125–1133. DOI:10.1557/mrc.2019.118.

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W (2016) Openai gym. *arXiv preprint arXiv:1606.01540* .

Chard K, Tuecke S and Foster I (2016) Globus: Recent enhancements and future plans. In: *XSEDE16 Conference on Diversity, Big Data, and Science at Scale*. pp. 1–8.

Chard R, Babuji Y, Li Z, Skluzacek T, Woodard A, Blaiszik B, Foster I and Chard K (2020) funcX: A federated function serving fabric for science. In: *29th ACM International Symposium on High-Performance Parallel and Distributed Computing*. DOI:10.1145/3369583.3392683.

Chard R, Li Z, Chard K, Ward L, Babuji Y, Woodard A, Tuecke S, Blaiszik B, Franklin M and Foster I (2019) DLHub: Model and data serving for science. In: *International Parallel and Distributed Processing Symposium*. IEEE, pp. 283–292.

Cheng L, Assary RS, Qu X, Jain A, Ong SP, Rajput NN, Persson K and Curtiss LA (2015) Accelerating electrolyte discovery for energy storage with high-throughput screening. *The Journal of Physical Chemistry Letters* 6(2): 283–291. DOI:10.1021/jz502319n.

Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B and Bharath AA (2018) Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine* DOI:10.1109/MSP.2017.2765202.

Dandu N, Ward L, Assary RS, Redfern PC, Narayanan B, Foster IT and Curtiss LA (2020) Quantum-chemically informed machine learning: Prediction of energies of organic molecules with 10 to 14 non-hydrogen atoms. *The Journal of Physical Chemistry A* 124(28): 5804–5811. DOI:10.1021/acs.jpca.0c01777.

Dehghannasiri R, Xue D, Balachandran PV, Yousefi MR, Dalton LA, Lookman T and Dougherty ER (2017) Optimal experimental design for materials discovery. *Computational Materials Science* 129: 311–322.

Dryden N, Maruyama N, Benson T, Moon T, Snir M and Van Essen B (2019) Improving strong-scaling of CNN training by exploiting finer-grained parallelism. In: *International Parallel and Distributed Processing Symposium*.

Dunn A, Brenneck J and Jain A (2019) Rocketsled: A software library for optimizing high-throughput computational searches. *Journal of Physics: Materials* 2(3): 034002. DOI:10.1088/2515-7639/ab0c3d.

Echekki T and Mirgolbabaei H (2015) Principal component transport in turbulent combustion: A posteriori analysis. *Combustion and Flame* 162(5): 1919–1933.

Ellis J and Rajamanickam S (2020) miniGAN. https://proxyapps.exascaleproject.org/app/minigan/.

Espeholt L, Marinier R, Stanczyk P, Wang K and Michalski M (2019) SEED RL: Scalable and efficient deep-RL

with accelerated central inference. *arXiv preprint arXiv:1910.06591* .

Espeholt L, Soyer H, Munos R, Simonyan K, Mnih V, Ward T, Doron Y, Firoiu V, Harley T, Dunning I et al. (2018) Impala: Scalable distributed deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561* .

Foster I, Ainsworth M, Allen B, Bessac J, Cappello F, Choi JY, Constantinescu E, Davis PE, Di S, Di W, Guo H, Klasky S, Kleese Van Dam K, Kurc T, Malik A, Mehta K, Mueller K, Munson T, Ostouchov G, Parashar M, Peterka T, Pouchard L, Tao D, Tugluk O, Wild S, Wolf M, Wozniak J, Xu W, and Yoo S (2017) Computing just what you need: Online data analysis and reduction at extreme scales. In: *European Conference on Parallel Processing*. pp. 3–19.

Foster I, Ainsworth M, Bessac J, Cappello F, Choi J, Di S, Gok AM, Guo H, Huck KA, Kelly C, Klasky S, Kleese van Dam K, Liang X, Mehta K, Parashar M, Peterka T, Pouchard L, Shu T, van Dam H, Wozniak JM, Wolf M, Xu W, Yakushin I, Yoo S and Munson T (2020) Online data analysis and reduction: An important co-design motif for extreme-scale computers. *International Journal of High-Performance Computing Applications* in press.

Frazier PI, Powell WB and Dayanik S (2008) A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization* 47(5): 2410–2439.

Gal Y and Ghahramani Z (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: *International Conference on Machine Learning*. pp. 1050–1059.

Garcia-Cardona C, Kannan R, Johnston T, Proffen T, Page K and Seal SK (2019) Learning to Predict Material Structure from Neutron Scattering Data. In: *IEEE International Conference on Big Data*. pp. 4490–4497. DOI:10.1109/BigData47090.2019.9005968.

Goodfellow I (2016) NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv e-prints* : arXiv:1701.00160.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y (2014) Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND and Weinberger KQ (eds.) *Advances in Neural Information Processing Systems 27*. pp. 2672–2680. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Hahn O and Abel T (2011) Multi-scale initial conditions for cosmological simulations. *Monthly Notices of the Royal Astronomical Society* 415(3): 2101–2121. DOI: 10.1111/j.1365-2966.2011.18820.x.

Heroux MA, Doerfler DW, Crozier PS, Willenbring JM, Edwards HC, Williams A, Rajan M, Keiter ER, Thornquist HK and Numrich RW (2009) Improving Performance via Mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories.

Jiang S, Malkomes G, Abbott M, Moseley B and Garnett R (2018) Efficient nonmyopic batch active search. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N and Garnett R (eds.) *Advances in Neural Information Processing Systems 31*. pp. 1099–1109.

Jin W, Barzilay R and Jaakkola TS (2018) Junction tree variational autoencoder for molecular graph generation. *CoRR* abs/1802.04364. URL `http://arxiv.org/abs/1802.04364`.

Li Z, Chard R, Ward L, Chard K, Skluzacek TJ, Babuji Y, Woodard A, Tuecke S, Blaiszik B, Franklin MJ and Foster I (2020) DLHub: Simplifying publication, discovery, and use of machine learning models in science. *Journal of Parallel and Distributed Processing* in press.

Lookman T, Eidenbenz S, Alexander F and Barnes C (2018) *Materials Discovery and Design: By Means of Data Science and Optimal Learning*, volume 280. Springer.

Majewski PW and Yager KG (2016) Rapid ordering of block copolymer thin films. *Journal of Physics: Condensed Matter* 28(40): 403002. DOI:10.1088/0953-8984/28/40/403002. URL `https://doi.org/10.1088%2F0953-8984%2F28%2F40%2F403002`.

Mathuriya A, Bard D, Mendygral P, Meadows L, Arnemann J, Shao L, He S, Kärnä T, Moise D, Pennycook SJ, Maschhoff K, Sewall J, Kumar N, Ho S, Ringenburg MF, Prabhat and Lee V (2018) CosmoFlow: Using deep learning to learn the universe at scale. In: *SC'18*. IEEE Press.

Moritz P, Nishihara R, Wang S, Tumanov A, Liaw R, Liang E, Elibol M, Yang Z, Paul W, Jordan MI and Stoica I (2017) Ray: A distributed framework for emerging ai applications.

Mustafa M, Bard D, Bhimji W, Lukic Z, Al-Rfou R and Kratochvíl J (2017) CosmoGAN: Creating high-fidelity weak lensing convergence maps using generative adversarial networks. *Computational Astrophysics and Cosmology* 6: 1–13.

Mustafa M, Bard D, Bhimji W, Lukić Z, Al-Rfou R and Kratochvil JM (2019) CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks. *Computational Astrophysics and Cosmology* 6(1): 1. DOI:10.1186/s40668-019-0029-9.

Noack MM, Doerk GS, Li R, Fukuto M and Yager KG (2020) Advances in Kriging-based autonomous x-ray

scattering experiments. *Scientific Reports* 10(1): 1325. DOI:10.1038/s41598-020-57887-x. URL https://doi.org/10.1038/s41598-020-57887-x.

Noack MM, Yager KG, Fukuto M, Doerk GS, Li R and Sethian JA (2019) A Kriging-based approach to autonomous experimentation with applications to x-ray scattering. *Scientific Reports* 9(1): 11809. DOI:10.1038/s41598-019-48114-3. URL https://doi.org/10.1038/s41598-019-48114-3.

Nowak SR and Yager KG (2020) Photothermally directed assembly of block copolymers. *Advanced Materials Interfaces* 7(5): 1901679. DOI:10.1002/admi.201901679. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/admi.201901679.

Oyama Y, Maruyama N, Dryden N, McCarthy E, Harrington P, Balewski J, Matsuoka S, Nugent P and Van Essen B (2020) The Case for Strong Scaling in Deep Learning: Training Large 3D CNNs with Hybrid Parallelism. *arXiv e-prints* : arXiv:2007.12856.

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L and Lerer A (2017) Automatic differentiation in pytorch. In: *NIPS-W*.

Pronzato L (2008) Optimal experimental design and some related control problems. *Automatica* 44(2): 303–325.

Radford A, Metz L and Chintala S (2016) Unsupervised representation learning with deep convolutional generative adversarial networks. In: *ICLR*. pp. 1–16. DOI:10.1007/s11280-018-0565-2.

Richards D, Aaziz O, Cook J, Kuehn J, Moore S, Pruitt D, Vaughan C and Watson G (2020) Quantitative performance assessment of proxy apps and parents. Technical Report Milestone ADCD-504-9.

Sergeev A and Balso MD (2018) Horovod: Fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799* .

Settles B (2009) Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Smith DGA, Altarawy D, Burns LA, Welborn M, Naden LN, Ward L, Ellis S, Pritchard BP and Crawford TD (2020) The MolSSI QCArchive project: An open-source platform to compute, organize, and share quantum chemistry data. *WIREs Computational Molecular Science* DOI:10.1002/wcms.1491.

Sprague MA, Ananthan S, Vijayakumar G and Robinson M (2020) ExaWind: A multifidelity modeling and simulation environment for wind energy. *Journal of Physics: Conference Series* 1452: 012071. DOI:10.1088/1742-6596/1452/1/012071.

Sutton RS and Barto AG (1998) *Reinforcement learning: An introduction*, volume 1. MIT.

Tassev S, Eisenstein DJ, Wand elt BD and Zaldarriaga M (2015) sCOLA: The n-body COLA method extended to the spatial domain. *arXiv e-prints* : arXiv:1502.07751.

Tassev S, Zaldarriaga M and Eisenstein DJ (2013) Solving large scale structure in ten easy steps with COLA. *Journal of Cosmology and Astroparticle Physics* 2013(6): 036. DOI:10.1088/1475-7516/2013/06/036.

Toby BH and Von Dreele RB (2013) GSAS-II: The genesis of a modern open-source all purpose crystallography software package. *Journal of Applied Crystallography* 46(2): 544–549.

Van Essen B, Kim H, Pearce R, Boakye K and Chen B (2015) LBANN: Livermore Big Artificial Neural Network HPC toolkit. In: *Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC '15. New York, NY, USA: Association for Computing Machinery. ISBN 9781450340069. DOI:10.1145/2834892.2834897. URL https://doi.org/10.1145/2834892.2834897.

Walther M, Oñorbe J, Hennawi JF and Lukić Z (2019) New Constraints on IGM Thermal Evolution from the Lyα Forest Power Spectrum. *The Astrophysical Journal* 872(1): 13. DOI:10.3847/1538-4357/aafad1.

Ward L, Blaiszik B, Foster I, Assary RS, Narayanan B and Curtiss L (2019) Machine learning prediction of accurate atomization energies of organic molecules from low-fidelity quantum chemical calculations. *MRS Communications* 9(3): 891–899. DOI:10.1557/mrc.2019.107.

Wozniak JM, Jain R, Balaprakash P, Ozik J, Collier NT, Bauer J, Xia F, Brettin T, Stevens R, Mohd-Yusof J, Cardona CG, Essen BV and Baughman M (2018) CANDLE/supervisor: A workflow framework for machine learning applied to cancer research. *BMC Bioinformatics* 19(S18). DOI:10.1186/s12859-018-2508-4.

Yoon BJ, Qian X and Dougherty ER (2013) Quantifying the objective cost of uncertainty in complex dynamical systems. *IEEE Transactions on Signal Processing* 61(9): 2256–2266.

Zhou Z, Kearnes S, Li L, Zare RN and Riley P (2019) Optimization of molecules via deep reinforcement learning. *Scientific Reports* 9(1): 10752. DOI:10.1038/s41598-019-47148-x.

Author biographies

Francis J. Alexander is Deputy Director of the Computational Science Initiative at Brookhaven National Laboratory.

James Ang is Chief Scientist for computing at Pacific Northwest National Laboratory.

Jan Balewski is a PDSF Consultant in the Data Science Engagement Group at Lawrence Berkeley National Laboratory.

Jenna A. Bilbrey is a Research Scientist in the National Security Directorate at Pacific Northwest National Laboratory.

Tiernan Casey is a Senior Member of Technical Staff in the Extreme-Scale Data Science and Analytics department at Sandia National Laboratories.

Ryan Chard is an Assistant Computer Scientist in the Data Science and Learning Division at Argonne National Laboratory.

Jong Choi is a Scientist in the Scientific Data Group, Computer Science and Mathematics Division, Oak Ridge National Laboratory.

Sutanay Choudhury is a Senior Research Scientist in the Physical and Computational Sciences Directorate at Pacific Northwest National Laboratory.

Bert Debusschere is a Distinguished Member of Technical Staff at Sandia National Laboratories. His research focuses on assessing the confidence in numerical simulations.

Anthony M. DeGennaro is an applied mathematician and computer scientist at the Computational Science Initiative of Brookhaven National Laboratory; his research interests include reduced-order modeling, uncertainty quantification, dynamical systems, and machine learning.

Nikoli Dryden is a post doctoral researcher at ETH Zurich.

J. Austin Ellis is a postdoctoral researcher in the Scalable Algorithms Department in the Center for Computing Research at Sandia National Laboratories.

Ian Foster is Senior Scientist and Distinguished Fellow, and director of the Data Science and Learning Division, at Argonne National Laboratory, and the Arthur Holly Compton Distinguished Service Professor of Computer Science at the University of Chicago.

Cristina Garcia Cardona is a Staff Scientist in the Computer, Computational and Statistical Sciences Division at Los Alamos National Laboratory.

Sayan Ghosh is a Computer Scientist in the Data Sciences group (part of Advanced Computing, Mathematics, and Data Division) at the Pacific Northwest National Laboratory in Richland, WA.

Peter Harrington is a machine learning engineer at Lawrence Berkeley National Laboratory.

Yunzhi Huang is an Electrical Engineering Researcher in Control and Optimization group at Pacific Northwest National Laboratory.

Shantenu Jha is the Chair of the Center for Data Driven Discovery in the Computational Science Initiative at Brookhaven National Laboratory, and Professor in the Computer Engineering Department at Rutgers University.

Travis Johnston is a Research Scientist in the Computer Science and Mathematics Division, Oak Ridge National Laboratory.

Ai Kagawa is an Assistant Scientist working on optimization and machine learning at Brookhaven National Laboratory.

Ramakrishnan Kannan is a Computational Data Scientist in the Computer Science and Mathematics Division, Oak Ridge National Laboratory.

Neeraj Kumar is a Computational Data Scientist in the Earth and Biological Sciences Directorate at Pacific Northwest National Laboratory.

Zhengchun Liu is an Assistant Computer Scientist in the Data Science and Learning Division at Argonne National Laboratory.

Naoya Maruyama was a Computer Scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory and is now a researcher at Nvidia.

Satoshi Matsuoka is the Director of the RIKEN Center for Computational Science (R-CCS) and a Full Professor at the Global Scientific Information and Computing Center (GSIC) at the the Tokyo Institute of Technology.

Erin McCarthy is a Ph.D. Candidate at the University of Oregon and was a summer intern at Lawrence Livermore National Laboratory.

Jamaludin Mohd-Yusof is a Scientist in the Computer, Computational and Statistical Sciences Division at Los Alamos National Laboratory, where he develops novel algorithms and applications for high performance computing and emerging architectures.

Peter Nugent is a Senior Staff Scientist at Lawrence Berkeley National Laboratory where he is the Department Head for Computational Science and Division Deputy for Scientific Engagement in the Computational Research Division.

Yosuke Oyama is a Ph.D. candidate at Tokyo Institute of Technology and summer intern at Lawrence Livermore National Laboratory.

Thomas Proffen is the Director of the High

Performance Computing and Data Analytics Science Initiative of the Neutron Science Directorate at Oak Ridge National Laboratory.

David Pugmire is a Senior Scientist in the Scientific Data Group, Computer Science and Mathematics Division at Oak Ridge National Laboratory.

Sivasankaran Rajamanickam is a principal member of technical staff in the Scalable Algorithms Department in the Center for Computing Research at Sandia National Laboratories.

Vinay Ramakrishniah is a staff scientist with a background in Electrical and Computer Engineering; his research interests include high-performance computing, artificial intelligence, antenna theory, signal processing, and optimizations.

Malachi Schram is a Senior Research Scientist and the leads the Data Science Architecture and A.I. team in the Physical and Computational Sciences Directorate at Pacific Northwest National Laboratory.

Sudip K. Seal is a Senior Researcher in the Computer Science and Mathematics Division at Oak Ridge National Laboratory.

Ganesh Sivaraman is an Assistant Computer Scientist in the Data Science and Learning Division at Argonne National Laboratory.

Christine Sweeney is a Scientist working in the areas of exascale computing, machine learning, and light source experiment data analytics workflows at Los Alamos National Laboratory.

Li Tan is an Assistant Computer Scientist in the Computational Science Initiative at Brookhaven National Laboratory.

Rajeev Thakur is a Senior Scientist and Deputy Director of the Data Science and Learning Division at Argonne National Laboratory.

Brian Van Essen is the Informatics Group Leader in the Center for Applied Scientific Computing at Lawrence Livermore National Lab.

Logan Ward is an Assistant Computer Scientist in the Data Science and Learning Division at Argonne National Laboratory.

Paul Welch is a Scientist in the Theoretical Division at Los Alamos National Laboratory; his background is in the theory and modeling of polymers, with a focus on nanomaterials and non-equilibrium phenomena.

Michael Wolf manages the Scalable Algorithms Department in the Center for Computing Research at Sandia National Laboratories.

Sotiris S. Xantheas is a Laboratory Fellow in the Advanced Computing, Mathematics and Data Division at Pacific Northwest National Laboratory in Richland, WA and an Affiliate Professor, UW-PNNL Distinguished Faculty Fellow in the Department of Chemistry at the University of Washington in Seattle, WA, USA.

Kevin G. Yager is Group Leader for the Electronic Nanomaterials Group in the Center for Functional Nanomaterials at Brookhaven National Laboratory.

Shinjae Yoo is the Group Leader for the Machine Learning Group in the Computational Science Initiative at Brookhaven National Laboratory.

Byung-Jun Yoon is a scientist at Brookhaven National Laboratory and also an Associate Professor in the Electrical and Computer Engineering Department at Texas A&M University