Learning Everywhere: Pervasive Machine Learning for Effective High-Performance Computation: Application Background

Geoffrey Fox[1], James A. Glazier[2], JCS Kadupitiya[3], Vikram Jadhao[4], Minje Kim[5], Judy Qiu[6], James P. Sluka[7], Endre Somogyi[8], Madhav Marathe[9], Abhijin Adiga[10], Jiangzhuo Chen[11], Oliver Beckstein[12], and Shantenu Jha[13]

[1,2,3,4,5,6,7,8]Indiana University, [9,10,11]University of Virginia, [12,13]Arizona State University, [14]Rutgers University

Email: [1]gcf@indiana.edu, [2]jaglazier@gmail.com, [3]vjadhao@indiana.edu, [4]minje@indiana.edu, [5]judy.qiu09@gmail.com, [6]jsluka@indiana.edu, [7]somogyie@indiana.edu, [8]marathe@virginia.edu, [9]adiga@virginia.edu, [10]chenj@virginia.edu, [11]obeckste@asu.edu, [12]shantenu.jha@rutgers.edu

March 2, 2019

1 Introduction

This paper describes opportunities at the interface between large-scale simulations, experiment design and control, machine learning (ML including deep learning DL) and High-Performance Computing. We describe both the current status and possible research issues in allowing machine learning to pervasively enhance computational science. How should one do this and where is it valuable? We focus on research challenges on computing for science and engineering (as opposed to commercial) use cases for both big data and big simulation problems. We summarize this long report at [1].

The convergence of HPC and data-intensive methodologies [2] provide a promising approach to major performance improvements. Traditional HPC simulations are reaching the limits of original progress. The end of Dennard [3] scaling of transistor power usage and the end of Moores Law as originally formulated has yielded fundamentally different processor architectures. The architectures continue to evolve, resulting in highly costly if not damaging churn in scientific codes that need to be finely tuned to extract the last iota of parallelism and performance.

In domain sciences such as biomolecular sciences, advances in statistical algorithms and runtime systems have enabled extreme scale ensemble based applications [4] to overcome limitations of traditional monolithic simulations. However, in spite of several orders of magnitude improvement in efficiency from these adaptive ensemble algorithms, the complexity of phase space and dynamics for modest physical systems, require additional orders of magnitude improvements and performance gains.

In many application domains, integrating traditional HPC approaches with machine learning methods arguably holds the greatest promise towards overcoming these barriers. The need for performance increase underlies the international efforts behind the exascale supercomputing initiatives and we believe that integration of ML into large scale computations (for both simulations and analytics) is a very promising way to get even large performance gains. Further, it can enable

paradigms such as control or steering and provide a fundamental approach to coarse-graining which is a difficult but essential aspect of the many multi-scale application areas. Papers at two recent workshops BDEC2 [5, 6, 7, 8] and NeurIPS [9] confirm our point of view and our approach is synergistic with BDEC2 [10] with its emphasis on new application requirements and their implications for future scientific computing software platforms. We would like to distinguish between traditional performance measured by operations per second or benchmark scores and the effective performance that one gets by combining learning with simulation and gives increased performance as seen by the user without changing the traditional system characteristics.

Our particular interest is the cases where there is a tight coupling between the learning and simulation components, which is seen especially in the categories in MLforHPC in section 1.1.

1.1 Different Interfaces of Machine Learning and HPC

We have identified [11, 12, 10] several important distinctly different links between machine learning (ML) and HPC. We define two broad categories: HPCforML and MLforHPC,

- **HPCforML:** Using HPC to execute and enhance ML performance, or using HPC simulations to train ML algorithms (theory guided machine learning), which are then used to understand experimental data or simulations.
- **MLforHPC:** Using ML to enhance HPC applications and systems

This categorization is related to Jeff Dean's "Machine Learning for Systems and Systems for Machine Learning" [13] and Satoshi Matsuoka's convergence of AI and HPC [14]. We further subdivide **HPCforML** as

- **HPCrunsML:** Using HPC to execute ML with high performance
- **SimulationTrainedML:** Using HPC simulations to train ML algorithms, which are then used to understand experimental data or simulations.

We also subdivide **MLforHPC** as

- **MLautotuning:** Using ML to configure (autotune) ML or HPC simulations.
- **MLafterHPC:** ML analyzing results of HPC as in trajectory analysis and structure identification in biomolecular simulations
- **MLaroundHPC:** Using ML to learn from simulations and produce learned surrogates for the simulations. The same ML wrapper can also learn configurations as well as results. This differs from SimulationTrainedML as the latter is typically using learnt network to redict observation (as in light source data analysis in [8]) whereas in MLaroundHPC we are using the ML to improve the HPC performance.
- **MLControl:** Using simulations (with HPC) in control of experiments and in objective driven computational campaigns [15]. Here the simulation surrogates are very valuable to allow real-time predictions.

All 6 topics above are important and pose many research issues in computer science and cyberinfrastructure, directly in application domains and in the integration of technology with applications. In this report, we focus on topics in MLforHPC, with close coupling between ML, simulations, and

HPC. We involve applications as a driver for the requirements and evaluation of the computer science and infrastructure. In researching **MLaroundHPC** we will consider ML wrappers for either HPC simulations or complex ML algorithms [16] implemented with HPC. Our focus is on how to increase effective performance with the learning everywhere principle and how to build efficient learning everywhere parallel systems.

1.2 Deep Learning and Machine Learning Background

This report does not cover development of new ML algorithms but rather the advancing the understanding of ML and Deep Learning (DL) in support of MLaroundHPC. Of course, the usage experience is likely to suggest new ML approaches of value outside the MLaroundHPC arena.

If one is to use an ML to replace a simulation, then an accuracy estimate is essential and as discussed in section 6 we intend to build on initial work on UQ with ML such as that using dropout regularization to build ensembles for UQ. There are more sophisticated Bayesian methods to investigate. The research must also address ergodicity, viz., have we learned across the full phase space of initial values. Here methods taken from Monte-Carlo arena could be useful as reliable integration over a domain is related to reliable estimates of values defined across a domain. We will need an extensive study of Uncertainty Quantification (UQ) from learning networks and we can expect progress of broad value as this field is still rather immature [17, 18]. Further much of our learning is for analytic functions whereas much of the existing DL experience is for discrete-valued classifiers of commercial importance.

1.3 Simulation Background

One can view the use of ML learned surrogates as a performance boost that can lead to huge speedups as calculation of a prediction from a trained network, can be many orders of magnitude faster than full execution of the simulation as shown in section 5.3. One can reach Exa or even Zetta scale equivalent performance for simulations with existing hardware systems. These high-performance surrogates are valuable in education and control scenarios by just speeding existing simulations. Domain-specific expertise will be needed to understand the needed accuracy and the number of training simulation runs needed.

Development of systematic ML-based coarse-graining techniques is a particular goal of this study that we study in both socio-technical simulations and nano-bio(cell)- tissue layering. Simple examples are the use of a surrogate to represent a chemistry potential or a larger grain size to solve the diffusion equation underlying cellular and tissue level simulations. ML has been shown to be very effective in learning sub-grid processes in climate simulations and gives around a factor of 10 speedup [19]. UberCloud has an interesting example [20] where fluid flow around obstacles is learnt for engineering simulations where a training set of 70,000 gives high fidelity predictions when changing the shape of the obstacles with an inference time that is a 1000 thousand times faster than direct simulation.

There are many groups working in MLaroundHPC but most of the work is just starting and not built around a systematic study of research issues as we propose. There is some deep work in building reduced dimension models to use in control scenarios. [21, 22] In sections 2-4 we look at three distinct important areas; Networked systems with socio-technical simulations, multiscale cell and tissue simulations and at a finer scale biomolecular and nanoscale molecular systems.

Already, autotuning with systems like ATLAS is hugely successful and gives an initial view of MLautotuning. As well as choosing block sizes to improve cache use and vectorization, MLautotuning can also be used for simulation mesh sizes [23] and in big data problems for configuring

databases and complex systems like Hadoop and Spark [24, 25]. In fact there are striking results [13, 26] in using learning as a surrogate for machine learning itself.

We note that biomolecular and biocomplexity areas which represent 40% of the HPC cycles used on NSF computational resources and so this is an area that is particularly ready and valuable. Molecular sciences has had several successful examples of using ML for autotuning and ML for analyzing the output of HPC simulation data. Several fields have made progress in using MLaroundHPC, e.g., Cosmoflow [27] and CosmoGAN [28] are amongst the better known projects; and the Materials community is actively exploring the uptake of MLControl for the design of materials [8].

Section 5 discusses cyberinfrastructure and computer science questions, section 6 covers uncertainty quantification for learnt results while section 7 the infrastructure requirements needed to implement MLforHPC. Section 8 covers new opportunities and research issues.

2 Machine learning for Networked Systems: Inference and Forecasting

2.1 Introduction

Broadly, two mainstream modeling approaches are used to understand complex phenomena: Machine learning and mechanistic models. Across domains, these methods have been compared [29, 30]. Here, we explore ways to fuse these methods and thus overcome the shortcomings of each approach, especially in scenarios where data is sparse and knowledge of underlying mechanism is inadequate. Here, we consider a particular class of mechanistic models - network dynamical systems which have been applied in diverse domains such as epidemiology and computational social science. We focus on prediction and inference problems. But, the methods developed can be applied to other topics as well (such as control). Machine learning models provide predictions on the outcomes of complex mechanisms by utilizing copious amounts of observations for a given problem. Typically, the approach requires choice of an algorithm which does not explicitly account for mechanisms that govern the phenomenon. Mechanistic models (like agent-based models on networks), on the other hand result from a bottom-up approach. They usually involve the mathematical description of the process. Operationally, the model is broken down into subsystems which are simpler to specify and learn through controlled experiments. The emergent behavior of the system is a result of these interacting subsystems. These models capture causality and are explanatory, and their utility extends beyond prediction. However, they typically tend to have too many parameters, are compute intensive, hard to calibrate, and critically depend on how well we understand the subsystems and the interactions between them. Historically, domain expertise of physical principles has been used to inform and improve machine learning outcomes (add citations later). In recent years, there have been several efforts to study physical processes under the umbrella of theory-guided data science (also called Model-based ML or Science-based ML) with focus on ANN as the primary learning tool. [31] provide a survey of these methods and their application to hydrology, climate science, turbulence modeling, etc. where the underlying theory can be used to reduce the variance in model parameters by introducing constraints or priors in the model space. While, we will incorporate these ideas, we intend to explore novel ways of interactions between the network dynamical system and learning algorithms to develop powerful and robust modeling frameworks. To this end, we consider the following directions:

- Learning from observational and simulation data. The goal is to develop robust and efficient models for forecasting. We will augment observations with existing domain knowledge and

behavior encapsulated in the agent-based model to inform the learning algorithm (consistency and generalizability). At the same time, the machine learning surrogate replaces the compute intensive simulation system and therefore enables efficient parameter space exploration.

- Inferring dynamical system behavior. The goal is to infer properties of the network dynamical system from time-series observations and a learning algorithm. In particular, the focus is on inferring local functions and two approaches will be explored: (i) We will exploit learning algorithms capability to infer the functions from observations and network interactions; and (ii) We will apply synthetic data generated from the learner to existing inference algorithms.

The first direction can be thought of as machine learning models being informed by network dynamics while the second direction leverages learning algorithms to better infer the dynamical systems. We will need a formal description for each direction.

2.2 Learning from observational and simulation data

Here, we explore ways to combine these approaches as a potential way to overcome the shortcomings of each approach, especially in scenarios where data is sparse. We consider a particular class of mechanistic models - network dynamical systems which have been applied in diverse domains such as epidemiology, computational social science. A network dynamical system is composed of a network where nodes of the network are agents (representing population, computers, etc.) and the edges capture the interactions between them. Each node is associated with a state (say susceptible 0 or infected 1) and a local function which decides the state of the node a function of the states of its neighbors in the network. The system behavior is represented by the vector of node states, which evolves over time. A popular example of such systems is the SEIR model of disease spread [32]. A networked dynamical system can be thought of as a graphical model that captures dynamics over networks. Recently graphical models of games, inference have also been studied extensively.

Here we will focus on prediction (forecasting) problems. The methods developed can be applied to other topics as well (such as inference or control).

In recent years, there have been several efforts to study physical processes under the umbrella of theory-guided data science (also called Model-based ML or Science-based ML) with focus on ANN as the primary learning tool. [31] provide an excellent survey of these methods and their application to hydrology, climate science, turbulence modeling, etc. where the underlying theory can be used to reduce the variance in model parameters by introducing constraints or priors in the model space. We have independently pursued a similar line of inquiry, but focused on social and health systems instead. The notion of theory in social and biological systems is a bit different than its use in describing physical systems. We will study on theoretical descriptions of contagion like processes on networks in this section.

Challenges. Data sparsity is often a challenge for applying machine learning, especially deep learning methods to forecasting problems in socio-technical systems. One example of such problems is to predict weekly incidence in future weeks as well as onset and peak of the season in an influenza epidemic. An infectious disease, such as influenza, spreads in a population through a social network via person to person interactions.

In such socio-technical systems, it is costly to collect data by surveillance or survey, and it is difficult or impossible to generate data by controlled experiments. Usually we have only limited observational data, e.g. weekly incidence number reported to the Centers for Disease Control and Prevention (CDC). Such data is of low spatial temporal resolution (weekly at state level), not real

time (at least one week delay), incomplete (reported cases are only a small fraction of actual ones), and noisy (adjusted by CDC several times after being published).

The complexity of the dynamics in such a large scale social network, due to individual level heterogeneity and behavior, as well as interactions among individuals, makes it difficult to train a machine learning model that can be generalized to patterns not yet presented in historical data. Completely data driven models cannot discover higher resolution details (e.g. county level incidence) from lower resolution ground truth data (e.g. state level incidence).

Approach. A schematic of the hybrid system is shown in Figure 1. There are three modules: The machine learning algorithm, the network dynamical system and evaluation module, which compares the predictions and of both models and directs the exploration of the parameter space until the quality constraints are satisfied. Initially, the network dynamical system is initialized based on assumptions and understanding of agent properties, network structure, etc. Then, it is calibrated using real-world observations of the phenomenon. The learning algorithm can be initially trained using real-world observations, or these observations can be provided along with simulation data at a later stage. After the initialization, an iterative process of interactions between these two models via the evaluation module ensues. The parameters for the evaluation module are the guarantees for prediction and budget constraints such as the maximum number of times the simulator can be invoked. An essential aspect of the framework is to embed uncertainty quantification for both the learning module [33, 34] as well as the dynamical system [35, 36] into the framework.

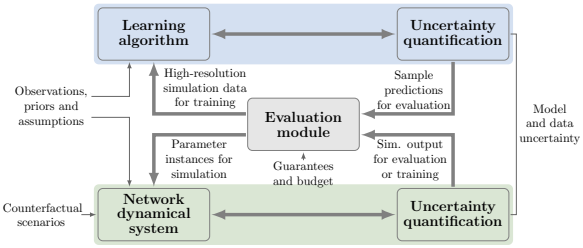

Figure 1: Schematic of the hybrid system of machine learning and agent-based simulation system.

The following are some of the key advantages of using such a hybrid framework as opposed to using only its individual components:

Consistency. The network dynamical system can be used to guide the learning algorithm so that it conforms to the principles and accepted understanding of the phenomenon. At the same time, the learning algorithm will facilitate model selection in a principled manner.

Generalizability. A learning algorithms performance can only be as good as the data provided to it. Here, we will consider ways in which synthetic data goes beyond the observational data and helps voiding overfitting. We consider two directions. Firstly, we consider dynamical systems whose outputs are at a finer resolution than observation data [37]. Secondly, we will simulate counter-factual scenarios for which no observation data is available. Both aspects are covered in the example.

Efficiency. In applications such as epidemic spread, population networks tend to be large, the dynamical process high dimensional and simulations are compute intensive. Therefore, parameter

space exploration and calibration is a major challenge. Recently, machine learning surrogates have been considered to efficiently explore the parameter space and use them to approximate outcomes of the target dynamical system [38]. Some learning algorithms naturally facilitate sensitivity analysis. Algorithms such as decision trees can potentially explain the behavior of the system as well. Much of the work in this regard has been empirical.

2.3 An Illustrative application: Simulation Guided Machine Learning for Epidemic Forecasting

Data sparsity is often a challenge for applying machine learning, especially deep learning methods to forecasting problems in socio-technical systems. One example of such problems is to predict weekly incidence in future weeks as well as onset and peak of the season in an influenza epidemic. An infectious disease, such as influenza, spreads in a population through a social network via person to person interactions.

In such socio-technical systems, it is costly to collect data by surveillance or survey, and it is difficult or impossible to generate data by controlled experiments. Usually we have only limited observational data, e.g. weekly incidence number reported to the Centers for Disease Control and Prevention (CDC). Such data is of low spatial temporal resolution (weekly at state level), not real time (at least one week delay), incomplete (reported cases are only a small fraction of actual ones), and noisy (adjusted by CDC several times after being published).

The complexity of the dynamics in such a large scale social network, due to individual level heterogeneity and behavior, as well as interactions among individuals, makes it difficult to train a machine learning model that can be generalized to patterns not yet presented in historical data. Completely data driven models cannot discover higher resolution details (e.g. county level incidence) from lower resolution ground truth data (e.g. state level incidence).

Statistical methods for epidemic forecasting include generalized linear models (GLM), autoregressive integrated moving average (ARIMA), and generalized autoregressive moving average (GARMA) [39, 40, 41]. [42] proposed a dynamic Poisson autoregressive model with exogenous input variables (DPARX) for flu forecasting. [43] proposed ARGO, an autoregressive-based influenza tracking model for nowcasting that incorporated CDC ILINet data and Google search data.

Deep learning methods based on artificial neural networks (ANN) have recently been applied to epidemic forecasting due to their capability to model non-linear processes. Recurrent Neural Network (RNN) has been demonstrated to be able to capture dynamic temporal behavior of a time sequence. [44] built an LSTM model for short-term ILI forecasting using CDC ILI and twitter data. [45] proposed LSTM based method that integrates the impacts of climatic factors and geographical proximity to achieve better forecasting performance. [46] constructed a deep learning structure combining RNN and convolutional neural network to fuse information from different sources.

Both statistical and deep learning methods cannot model heterogeneity at disaggregate levels based on aggregate data, so they have difficulty producing forecasts of a higher resolution than that of observed data. In addition, deep neural networks need large amount of training data. Thus overfitting often happens due to availability of only small amount of historical surveillance data.

Causal methods based on mechanistic models (e.g. S-E-I-R model for influenza epidemics) are developed for epidemic forecasting. Tuite et al. [47] estimate parameters in a compartmental model in order to predict morbidity in pandemic H1N1. [48] apply various filter methods to model and forecast influenza activity using an SIRS model. [49] propose an optimization approach where parameters of an agent based model are inferred from ground truth data then simulations are used to make epidemic forecasts. [50] and [51] infer the parameters of an agent based model from social media data for ILI forecasting. The benefit of causal methods is that the underlying mechanistic

models allow them to handle new patterns in data; and depending on the model resolution forecasts can be made of various resolutions, even down to individual level in case of a detailed agent based model. The major challenge with these causal methods is the high dimensionality of model parameters. Again, sparse ground truth data is not enough to calibrate many parameters. Furthermore, the agent based epidemiological models are highly compute intensive, making it non-trivial to search through the parameter space. These models usually demand high performance algorithms to simulate.

To address the challenges posed by data sparsity, we adopt the SimulationTrainedML approach. The idea is to use theory to guide machine learning / deep learning model training. Specifically for epidemic forecasting problem, we take an existing epidemiological model, derive a reasonable distribution of parameter values, sample many configurations of the model from the distribution and run HPC based simulations, then use the synthetic data produced by simulations to train a deep neural network, which can be used for forecasting. This approach integrates epidemiology theory, HPC simulations, and deep learning models under the paradigm of Theory Guided Machine Learning (TGML). It improves generalizability of machine learning models, ensures their physical consistency with the epidemiological models, and enables high resolution forecasting. Simulations can generate required synthetic training data for neural network models with minimal costs (mainly HPC costs). The mechanistic model can not only generate data similar to the observed but also produce data that has never occurred in reality, and allows the trained neural networks to model new patterns. The TGML approach can also easily make long term forecasting of epidemic dynamics using simulations. Furthermore, the behavior modeling in agent based models allows counter-factual analysis and what-if forecasts.

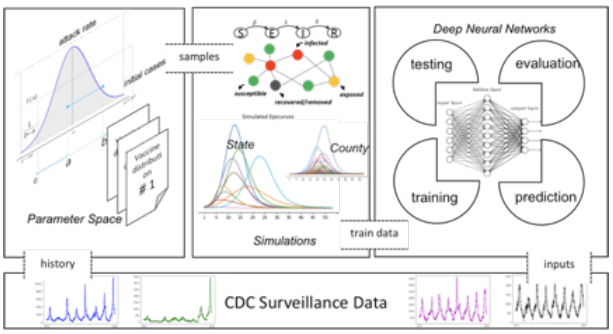

Figure 2: Simulation trained ML

For infectious disease forecasting, we propose DEFSI (Deep Learning Based Epidemic Forecasting with Synthetic Information), a framework that integrates the strengths of artificial neural networks and causal models. It consists of three components. (1) Disease model parameter space construction: Given an existing agent based epidemic model, we estimate a marginal distribution for each model parameter based on coarse surveillance data. (2) Synthetic training data generation: We generate a synthetic training dataset at high-resolution scale by running HPC simulations parameterized from the parameter space. (3) Deep neural network training and forecasting: We train a two-branch deep neural network model on the synthetic training dataset and use coarse surveillance data as inputs to make detailed forecasts.

Why should we build a reduced order model. A natural question is the need for such a neural network model, in other words, why cant a simulation be used directly to produce the needed forecast. There are a number of reasons why this is useful. First and foremost, once the

neural network model is developed real-time forecasting is possible. Second, such a model can also be used when one uses simulations for planning and optimization problems and needs forecasted values during the decision making phase. (Needs a bit more cleaning up)

We run experiments to evaluate DEFSI against several state-of-the-art epidemic forecasting methods. (1) LSTM (single-layer LSTM) [52]; (2) AdapLSTM (CDC + Weather data) [45]; (3) ARIMA (classic ARIMA) [40]; (4) ARGO (CDC + Google data) [43]; and (5) EpiFast (causal forecasting method based on EpiFast simulations) [49]. The last method can make county level forecasts while the others can only make state level forecasts. Experimental results show that DEFSI performs comparably or better than the other methods for state level forecasting. For county level forecasting, the DEFSI method significantly outperforms the EpiFast method.

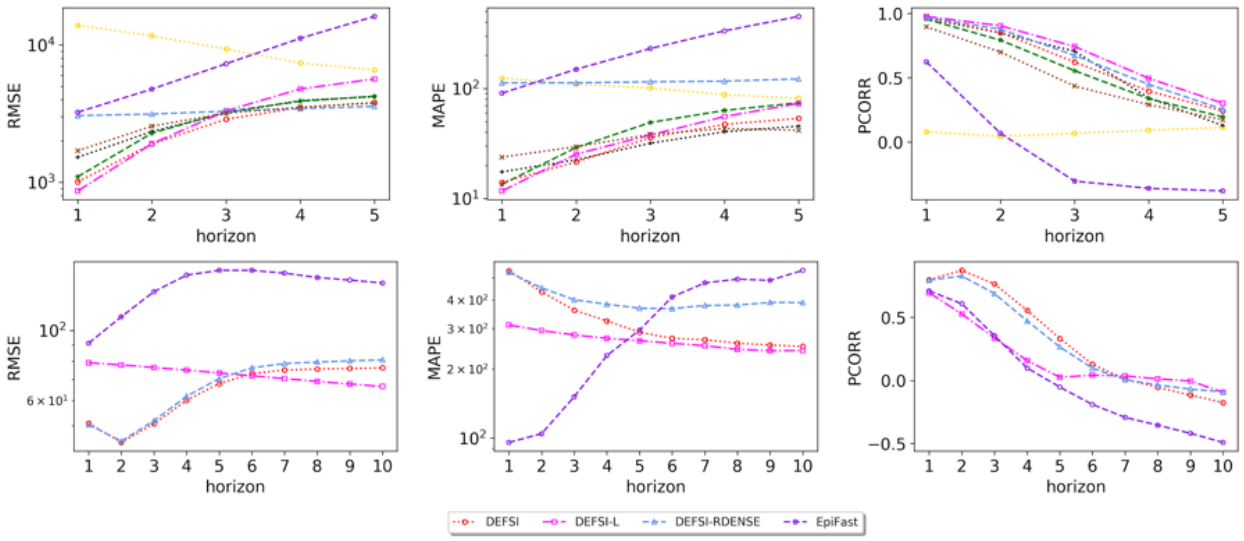

Figure 3: Virginia state level forecasting, 2017-2018 (top) and New Jersey county level forecasting, 2017-2018 (bottom).

3 Machine Learning for Virtual Tissue and Cellular Simulations

3.1 Virtual Tissue Models

Virtual Tissue (VT) simulations [53, 54] are mechanistic multiscale spatial simulations of living tissues which address questions about how tissues develop during embryogenesis, how they maintain themselves in adult life, how they repair themselves after damage, and how they fail due to internal and external perturbations. They also find application in the design of tissues (tissue engineering) and the development of medical therapies, especially personalized therapies. Sample application include personalized models of brain and breast cancers [55], virtual livers [56] for the analysis of toxicant effects on the liver [57, 58], personalized virtual hearts [59] for the design of surgical interventions, virtual lungs, virtual retinas and virtual embryos. While VT simulations are extremely diverse, both in their level of detail, numerical methodologies and range of temporal length scales, they share a number of features that make them appropriate targets for AI assistance. VT simulations are computationally challenging for a number of reasons: 1) VT simulations are agent-based, with the core agent often being biological cells. The number of cells in a real tissue is often of the order of 10^8 or more. Even with coarse-graining, the number of simulated cells is large

$(10^5 - 10^7)$, so calculating the temporal changes of cell shape, position and internal state is time consuming. 2) Agents are often hierarchical, with agents composed of multiple agents at smaller scales, and are highly diverse in type. As a result VT simulations must explicitly represent a very broad range of length and time scales (nanometers to meters and milliseconds to years). 3) Agents within and between scales interact strongly with each other, often over significant ranges [57]. Explicit models of long range force transduction and reorganization of the complex fiber networks [60] of extracellular matrix outside of cells and of the active cytoskeletal fiber networks inside of cells are particularly challenging. Solid mechanics models of bulk material deformations are also challenging. 3) Individual agents typically contain complex network sub models which control their properties and behaviors, e.g., of cell regulation, signaling and metabolism. 4) Materials properties may be complex, like the shear thickening or thinning or swelling or contraction of fiber networks, which may be individually expensive to simulate. 5) Modeling transport is also challenging. Agents exist in a complex environment, which governs the transport of multiple chemical species. In many VT simulations, the solution of the diffusion equation (with complex rules for active and passive boundary crossing) for many species under complex, every-changing conditions is the most computationally expensive part of the simulation. In other cases, e.g. when there is significant flow of blood or interstitial fluid, or bulk material movement, transport requires the solution of complex advection diffusion equations.6) Models are typically stochastic, so predictivity requires execution of many replicas under identical conditions. Importantly, this stochasticity usually does not represent error–different simulation outputs for the same parameters are both valid and the average of two valid solutions is not generally a valid solution. E.g., a vascular growth experiment which can generate two different random spatial vascular structures, or where two qualitatively different results where the interest is the probability of occurrence of an outcome–e.g. Does a cancer metastasize or not and if so, to what organ? 7) Simulations involve uncertainty both in model parameters (there may be hundreds of parameters, many of them poorly quantified) [61] and in model structure (the actual set of interaction rules and types). The former means that parameter optimization and uncertainty quantification are hard, the latter is worse, because the combinatorics involve searches over complex response surface spaces in which gradient methods dont work. 8) Biological and medical time-series data for concentrations or states are often qualitative, semiquantitative or differential, making their use in classical optimization difficult (their use requires the definition of case-specific metrics and their weighted inclusion in the definition of residuals–outcomes are often sensitive to this weighting and we lack formal ways to optimize these weights). 9) In many cases, time series for validation may not be synchronized with model time points (in terms of starting time, ending time or time rates). 10) VT models often produce movies of configurations in 2D x time or 3D x time. In general, we lack appropriate general metrics to compare experimental movies to VT outputs (and we have the additional issue of scaling, region of interest limitation and registration, as well as stochasticity), making feature-extraction and classical optimization problematic. 11) Finally, given the complexity of VT models and their simulation, there is the additional computational challenge of simulating populations of individuals. Simulation of a population can add several additional orders of magnitude to the computational challenge. Unsupervised learning, can, in principle, handle stochasticity, determine appropriate weights, registration and features to allow comparison between experiment and simulation.

In addition, VT simulations are often coupled to models at higher and lower scales, for example to whole body models represented as Physiological Pharmacokinetics (PBPK) models, or to multiple sub-cellular signaling or metabolic models. At each simulation scale there is often a variety of possible representations that vary in their level of detail. As the level of detail at each scale, and the number of couplings between scales increases, the computational complexities outlined in the previous discussion are multiplied. As outlined in Figure 3.1, coupling multiple scales, each with

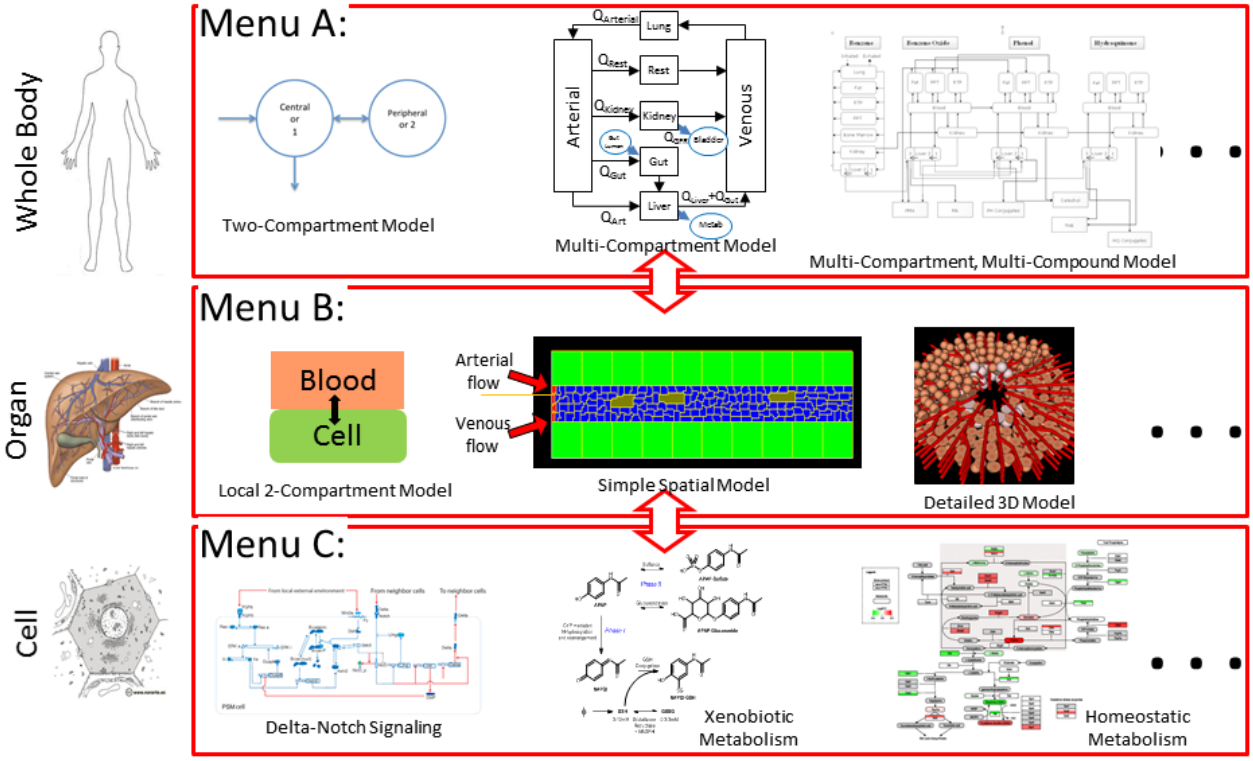

Figure 4: Multiscale VT simulations often include sub-models for particulate scales. For example, the whole body is represented as a PBPK model, coupled to a VT model, with each agent (e.g., cell) in the VT model coupled to a subcellular model. This approach typically, and optimally, uses different modeling modalities at each scale. Constructing a model in this manner allows simulations to include experimental data obtained from different sources. For example, the PBPK scale is suitable for including systemic measurables such as blood concentration versus time following a dose of a xenobiotic. The subcellular scale can often be calibrated using measurements from cell culture assays. There are often multiple possible models at a particular scale that differ in their level of detail.

the possibility of varying levels of details greatly increases the level of complexity. It is possible that ML techniques can be used to short circuit implementations at individual scales.

Perhaps surprisingly, given the above caveats, virtual tissue models can and do generate scientifically and practically meaningful results in a wide variety of cases, and some are even used clinically on a regular basis for the optimization of personalized therapies.

Secondly, a growing number of general modeling frameworks, CompuCell3D, PhysiCell [62], Morpheus, Neuron, VirtualCell, allow the high-level specification of VT simulations of different types. These frameworks typically share an approach to modularization, built around the definition of object classes and their properties, behaviors and interactions. Models are increasingly structured as pluggable components, which can be refined by inclusion of more detailed submodels or replaced by coarse-grained approximations.

3.2 Virtual Tissue Modelling and AI

AI can directly benefit VT applications in a number of ways (and certainly in many more which we have not yet considered):

1. Short-circuiting: The replacement of computationally costly modules with learned analogues. E.g. in the case of diffusive transport, we want to know the flux of a given chemical species at a given location given the global pattern of sources and sinks of the species, without having to solve the diffusion equations. In the case of a model of the heart we want to know the stress distribution in the heart tissue from the unstressed geometry of the heart and its current geometry, without having to solve the nonlinear solids equations which relate stress and strain in this complex material. In both cases, there are proof-of-principle demonstrations that such short-circuiting is possible and can speed up calculations by many orders of magnitude.

2. Parameter optimization in very high dimensional parameter spaces. VT models are typically significantly overfitted (not enough validation data for the hundreds of parameters), but the sets of parameters which lead to equally good solutions are of interest.

3. Model reduction and uncertainty quantification–the analysis of conditions under which parameters or model elements are unnecessary and can be eliminated from the simulation.

4. Encapsulation (coarse graining)–Can you take a bunch of agents, put a box around them and simulate the input-output relations of the box? A classic example of this is the difficulty of deriving meaningful bulk materials parameters from the microstructure of ECM or cytoskeletal components. The key challenge is the lack of homogeneity in biological systems that means that classical approaches to coarse-graining fail.

5. Approaches to treating stochasticity of results as information rather than noise.

6. In many cases the exact quantitative values of the simulation prediction are critical (the contraction efficiency of the heart, a surgical the safety margin around an MRI image of a tumor) and we would be using AI to interpolate between directly simulated results. In other cases, we are primarily interested in the bifurcation structure of the model with respect to one or more parameters (as a concentration of a chemical increases, a circuit goes from stable to oscillatory behavior). In the latter case we want to use AI approaches to predict the range of possible output configurations from input structures (classification) to address problems where model structure is uncertain.

7. The design of maximally discriminatory experiments for hypothesis testing is a key application of VT models–the use of AI to predict the (experimentally realizable) parameter sets for which two models with different hypotheses have the most distinct results would greatly facilitate the application of modeling to experiment.

8. A particular area of interest is the use of AI to run time backwards, to allow the design of initial conditions which lead to desired outputs–such inverse time simulation is especially critical to enable tissue engineering applications of VTs.

9. Another area of interest is the elimination of short time scales–in particular, the rate of blood flow is very fast compared to other tissue-level time scales. If AI can short-circuit the calculations of advection-diffusion transport, it would speed up typical tissue calculations by several orders of magnitude.

10. A final area of AI, which relates to more classical AI-based image analysis is the use of AI to generate initial conditions for VT simulations from experimental images. Of particular interest is the question of whether AI can classify complex stochastic spatial structures (e.g. the branching pattern airways in a human lung or the vascular structure of the liver) to

determine if a VT simulation has generated a tissue which structurally resembles its experimental counterparts. Currently, such classification is not possible using predefined feature extractions methods.

In each of these cases, a VT model has the advantage that it can generate limitless amounts of training data (albeit at high computational cost).

3.3 Multiscale Biological Modelling and MLandHPC

Therefore, there are several areas in multiscale biological modeling that could benefit from advances in MLforHPC. Such as;

1. Generation of additional multiscale time series from limited experimental data (detailed time series data is expensive to obtain). The ability to extract fundamental characteristics of the system, without necessarily developing deep mechanistic understanding of the controlling equations or parameters. Create "synthetic, complex time series data" that is consistent with the limited amount of experimental data.

2. Generation of multiscale time series across species. Leverage much larger data sets in experimental animals to extend the limited data in humans. E.g., map animal to human data including stochasticity and inter-individual variation.

3. Extraction of commonalities across VT simulations. For example, multiple VT models, each for a particular disease, share a common PBPK backbone. Use AI to extract parameters (and parameter ranges) that are VT-model independent such as apparent tissue volumes, blood flow rates etc.

4. Use of AI to score the "biological possibility" of a computationally constructed representation (e.g., a replicates) of a biological system. In particular for 2D and 3D tissue models used as replicates in VT simulations. If an AI classifier can differentiate between real 2D/3D data and synthetic data, then the synthetic data does not adequately represent reality.

5. Prediction of crashes in simulations. When exploring large parameter spaces it is often the case that certain sets of parameters result in either a numerical crash of the simulation or the simulation straying into a non-biologically possible region. MLfor HPC techniques that can monitor a large scale simulation project and develop predictors for when particular parameter sets may give invalid results allowing those simulations to be skipped.

Key concerns about AI replacement of model components include: 1) The accuracy of learned interpolations within the span of the training set, 2) the range of parameters outside of the training set span over which model prediction is qualitatively correct. 3) The efficiency of learning over very sparse training sets. 4) The ability to train when the same input data can lead to very different output data sets (in other words, where the inputs lead not to a single deterministic output, but to a complex range of outputs each of which is valid in its own right). Thus, classical back propagation algorithms may need to be rethought to handle the multivalency of outputs. 5) Model structure inference is challenging, because there is not necessarily a differentiable way to transition between models with different structures for gradient methods (unless we use systems with very high combinatorial complexity–all-to-all couplings).

In many of these cases, in other contexts, AI approaches do better than one might reasonably expect. In biology in particular, feedback systems tend to result in behaviors which are highly

robust to stochasticity and drift of parameters (e.g., changing the concentration of a key input chemical by a factor of ten has no measurable effect on system outputs), but failures tend to be dramatic (e.g. the transition between a regulated stem cell state and a cancer). This robustness makes the predictability of normal situations more credible, but also that divergences between AI predictions and reality may be abrupt and dramatic rather than gradual.

Prior work by Karniadakis [22, 63], Kevrekidis [21] and Nemenman [64] have shown that neural networks can effectively learn to reproduce the temporal behaviors of fairly complex biochemical regulatory and signaling networks over a broad range of physiological conditions. This ios complemented by successful applications in other domains [65]. Liang-liang [66] has shown that networks can also learn to short-circuit nonlinear biomechanics simulations of the aorta–being able to predict the stress and strain distribution in the human aorta from the morphology observable with MRI or CT alone. These successes provide important proof-of-principle for the use of AI to enhance VT simulations. However, the typical VT model is much more complex than these biological models. VTs typically involve the change of position and state of hundreds of thousands or millions of agents, each of which has complex internal control and complex deformable shape. In this sense, the use of AI is more similar to the use of AI to play a very extended version of the game of Go than to learn the behavior of a classical dynamical system. E.g. the dimensionality of the inputs for the spatial components of a VT is very high (the 3D positions, shape and surface and volume properties of every agent in the simulation). Many of these issues represent significant computer-science research problems.

We plan the investigate 3 separate projects which will demonstrate the utility of AI methods in accelerating and improving VT simulations.

1. **Project 1**: Focus–Parameter and model structure estimation in challenging conditions. Test the ability of various back-propagation methods to infer VT model parameters from stochastic simulation outputs for different levels of model complexity. Study the degradation of prediction as the number of parameters increases and the number of simulation replicas decreases to determine a heuristic trade-off for model inferability. Test the ability of AI to identify possible regulatory circuits for control of known biological systems.

2. **Project 2**: Focus–Short-circuiting of high dimensional models for acceleration. Develop a short-circuit diffusion solver which takes spatial source, sink and boundary geometries and conditions and returns the chemical concentration at a desired location and net fluxes over all boundary surfaces. Apply this accelerator to a VT model of a vascular tumor. Study the numerical accuracy of the field predictions and the range of situations in which the AI approximation becomes qualitatively wrong rather than numerically inaccurate. If the base diffusion solver works, extend to include a model of vascular transport via a capillary network.

3. **Project 3**–Focus–Coarse graining. Train a network on a microscopic model of fiber dynamics and reaction-advection-diffusion interactions and their effect on cell movement and polarization under representative conditions. Attempt to predict the dynamics of cell polarization (both mesenchymal to epithelial transition and lamellipodial polarization of fibroblasts) from internal state and external conditions.

Even if the more ambitious aims are not realized, packaging tools that allow the optimization or short-circuiting of well-defined submodels designed to have a clear set of inputs and outputs will allow the acceleration of many classes of VT simulation.

4 Machine Learning for Molecular and Nanoscale Simulations

4.1 MLaroundHPC for nanoscale simulation

4.1.1 Example

We start with a simple example to provide context and illustrate the value of MLaroundHPC in the nanoscale simulation domain. Consider a typical coarse-grained (with implicit solvent) molecular dynamics (MD) simulation of ions confined at the nanoscale: 1 nanosecond of dynamics (1 million computational steps) of a system of 500 ions on one processor takes about 12 hours of runtime, which is prohibitively large to extract converged results within reasonable time frame. Performing the same simulation on a single node with multiple cores using OpenMP shared memory accelerates the runtime by a factor of 10 ($\approx$ 1 hour), enabling simulations of the same system for longer physical times leading to more accurate results within reasonable computing time. The use MPI can dramatically enhance the performance of these simulations: for systems of thousands of ions, speedup of over 100 can be achieved, enabling the generation of accurate data for evaluating ionic distributions. Further, a hybrid OpenMP/MPI approach can provide even higher speedup of over 400 for similar-size systems enabling state-of-the-art research on the ionic structure in confinement via simulations that explore the dynamics of ions with less controlled approximations, accurate potentials, and for tens or hundreds of nanoseconds over a wider range of physical parameters. However, despite the employment of the optimal parallelization model suited for the size and complexity of the system, scientific simulations remain time consuming. In research settings, simulations can take up to several days and it is often desirable to foresee expected overall trends in key quantities; for example, how does the contact density vary as a function of ion concentration in the confinement. Long-time simulations and investigations for larger system sizes become prohibitively expensive. The situation gets increasingly worse as the models become more fine-grained. Explicit solvent leads to a dramatic decrease in computational performance. The above example is illustrative of the general problem in simulation of materials at the nanoscale; self-assembly phenomena associated with capsomeres assembling into a viral capsid or with assembly of functionalized nanoparticles into higher-order structures exhibit similar computational challenges.

4.1.2 Why MLaroundHPC and what does it enable

Given the powerful rise in ML and HPC technologies, it is not the question of if, but when, ML can be integrated with HPC to advance simulation methods and frameworks. It is critical to understand and develop the software frameworks to build ML layers around HPC to 1) enhance simulation performance 2) enable real-time and anytime engagement, and 3) broaden the applicability of simulations for both research and education (in-classroom) usage. Some early work in using ML in material simulations and scientific software applications (see Sec. 4.1.3) has already begun in this direction further inspiring the research objectives. Figure 5 shows a schematic of the MLaroundHPC framework for a generic nanoscale simulation. The goal is typically to predict the structure or correlation functions (outputs) characterizing the nanoscale system over a broad range of experimental control parameters (inputs). As Figure 5 shows, MLaroundHPC can enable the following outcomes:

1. Learn pre-identified critical features associated with the simulation output (e.g. key experimentally-relevant quantities such as contact ionic density in the above example or nanoparticle pair correlation functions in nanoparticle self-assembly simulations).

2. Generate accurate predictions for un-simulated statepoints (by entirely bypassing simulations)

3. Exhibit auto-tunability (with new simulation runs, the ML layer gets better at making predictions etc.)

4. Enable real-time, anytime, and anywhere access to simulation results from trained ML models. (particularly important for education use).

5. No run is wasted. Training needs both successful and unsuccessful runs.

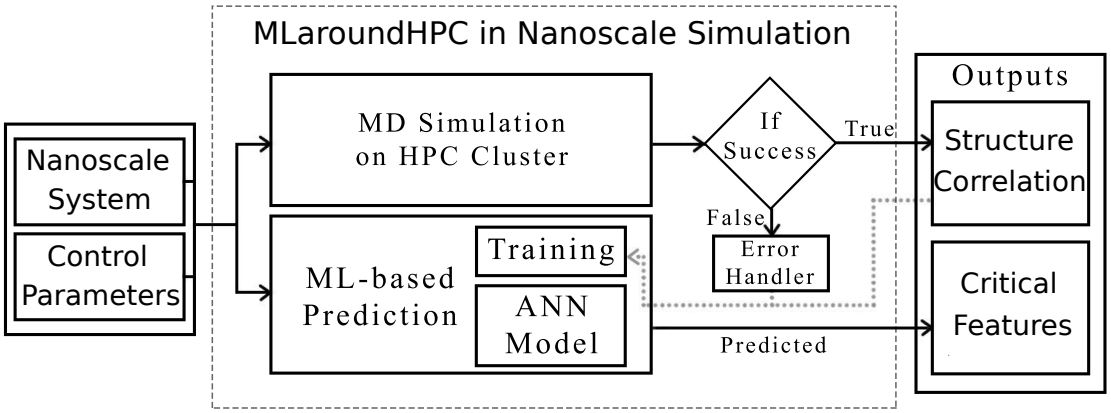

Figure 5: Schematic of the MLaroundHPC framework in nanoscale simulation.

To further illustrate these goals, we discuss an application related to the example simulation presented above in Sec. 4.1.1. The distribution of electrolyte ions near charged or neutral nanomaterials such as nanoparticles, colloids, or biological macromolecules determines their assembly behavior. As a result, the computation of structure of ions confined by surfaces that are nanometers apart has been the focus of several recent experiments and computational studies [67, 68, 69]. Simulations of such coarse-grained models have been employed by many researchers to extract the ionic distributions confined between the interfaces for a wide range of parameters such as electrolyte concentrations, ion valencies, and interfacial separations [70, 71, 68]. Simulations have been performed using customized codes developed in individual research groups [70, 72, 68] or general purpose software packages such as ESPRESSO [73] and LAMMPS [74].

Typically, the entire ionic distribution averaged over sufficient number of independent samples generated during the simulation is a quantity of interest. However, in many important cases, average values of contact density, peak density or density at the center of the slit (mid-point density) directly relate to important physical quantities associated with the effect of ionic self-assembly on the interaction between the surfaces [75, 76]. For example, the effective force between the two interfaces, the osmotic pressure, or the fraction of condensed ions directly depend on the contact density which is the density of ions at the point of closest approach of the ion to either interface [75]. Further, often it is useful to visualize expected trends in the behavior of contact or mid-point density as a function of solution conditions or ionic attributes, before running simulations to explore specific system conditions. In all these cases, it is desirable that a simulation framework provides rapid estimates of these critical output features with high accuracy. MLaroundHPC can enable precisely this. For this example, in Figure 5 our nanoscale system is the system of confined ions, experimental control parameters are electrolyte concentration, solvent permittivity, confinement length, desired output is the density profile (structural information), and critical features are the contact, peak, and mid-point ion densities. We demonstrated in our recent work [77] that an

artificial neural network successfully learned the desired features associated with the output ionic density profiles and rapidly generated predictions that are in excellent agreement with the results from explicit molecular dynamics simulations.

4.1.3 Related work

Traditionally, performance enhancement of scientific applications has largely been based on improving the runtime of the simulations using parallel computing methods, including different acceleration strategies such as OpenMP, MPI, and hybrid programming models. Recent years have seen a surge in the use of ML to accelerate computational techniques aimed at understanding material phenomena. ML has been used to predict parameters, generate configurations in material simulations, and classify material properties [78, 79, 80, 81, 82, 83, 84, 85, 86, 87]. For example, Fu et al. [81] employed ANN to select efficient updates for Monte Carlo simulations of classical Ising spin models that overcome the critical slowing down issues in Monte Carlo simulations. Similarly, Botu et al. [84] employed kernel ridge regression (KRR) to accelerate MD method for nuclei-electron systems by learning the selection of probable configurations in MD simulations, which enabled bypassing explicit simulations for several steps.

4.2 MLaroundHPC for biomolecular simulations

4.2.1 Current work

The use of ML and in particular DL approaches for biomolecular simulations is still in its infancy compared to other areas such as nano-science and materials science. This might be partly due to the large system sizes (100,000 of atoms), the importance of solvent (water, ions, lipids for membrane proteins), the high level of heterogeneity, and possibly the importance of both short range (10) and long range (electrostatic) interactions. Current work in this area originated in materials sciences [88, 89]. One promising approach is based on work by Behler and Parrinello [90] who devised a very efficient and accurate NN-based potential that was trained on quantum mechanical DFT energies; their key insight was to represent the total energy as a sum of atomic contributions and represent the chemical environment around each atom by an identically structured NN, which takes as input appropriate symmetry functions that are rotation and translation invariant as well as invariant to exchange of atoms but correctly reflect the local environment that determines the energy [88]. Based on this work, Gastegger *et al.* [91] used ML to accelerate ab-initio MD (AIMD) to compute accurate IR spectra for organic molecules including the biological Ala_3^+ tripeptide in the gas phase. They employed Behler-Parrinello HDNNP (high-dimensional neural network potentials) and an automated sampling scheme to converge HDNNPs but they did not perform on-the-fly training due to the high cost of DFT energy evaluations. Interestingly, the ML model was able to reproduce anharmonicities and incorporate proton transfer reactions between different Ala_3^+ tautomers without having been explicit trained on such a chemical event, highlighting the promise of such an approach to incorporate a wide range of physically relevant effects with the right training data. The ML model was >1000 faster than the traditional evaluation of the underlying quantum mechanical physical equations.

Roitberg *et al.* [92] trained a NN on QM DFT calculations, based on modified Behler-Parrinello symmetry functions. The resulting ANI-1 model was shown to be chemically accurate, transferrable, and the resulting NN potential could be evaluated at a cost similar to a classical force field so that ab-initio molecular dynamics can be performed to a fraction of the cost of solving the DFT calculations in AIMD. However, the model is built separately on a large database of single point DFT

energy calculations without tight coupling between the training and the simulation steps. Extensions of their work (improving NN generation with an active learning (AL) approach via Query by Committee (QBC)) demonstrated that proteins in an explicit water environment can be simulated with a NN potential at DFT accuracy [93]. The AL approach reduced the amount of required training data to 10% of the original model [93] by iteratively adding training data calculations for regions of chemical space where the current ML model could not make good predictions. Using transfer learning, the ANI-1 potential was extended to predict energies at the highest level of quantum chemistry calculations (coupled cluster CCSD(T)), with speedups in the billions [94].

4.2.2 Outlook

In general the focus has been on achieving DFT-level accuracy. Speeding up classical MD simulations with empirical force fields has not been attempted, likely because the empirical energy functions are typically simpler than the NN potentials and so the computational cost of evaluating a NN potential is higher than or on par with evaluation of a classical empirical potential. However, replacing solvent-solvent and solvent-solute interactions, which typically make up 80%-90% of the computational effort in a classical all-atom, explicit solvent simulation, with a NN potential promises large performance gains at a fraction of the cost of traditional implicit solvent models and with an accuracy comparable to the explicit simulations, as already discussed above in the case of electrolyte solutions. Furthermore, inclusion of polarization, which is expensive (factor 3-10 in current classical polarizable force fields) but of great interest when studying the interaction of multivalent ions with biomolecules might be easily achievable with appropriately trained ML potentials.

An interesting question is if ML potentials can be used to push the typical time step size in the integration of the classical equations of motions to times beyond the 2–5 fs that are currently feasible. Experience from heuristic coarse grained force fields [95, 96] shows that smoothing of the energy landscape by coarse graining allows much larger time steps, e.g. 10 - 20 fs. If ML potentials can be shown to also coarse-grain the energy landscape in a similar fashion then an enhancement of sampling and speedup of factors of 2–5 might be possible, at effectively the same accuracy as traditional classical all atom MD simulations. This assumes that ML potentials somehow capture dynamics on a low-dimensional manifold on which the energies and forces vary more smoothly than on the full $6N$ phase space [97].

5 Cyberinfrastructure and Computer Science Background for the Integration of Machine Learning and HPC

5.1 HPC for Machine Learning

There has been substantial community progress here with the Industry supported MLPerf [98] machine learning benchmark activity and Ubers Horovod Open Source Distributed Deep Learning Framework for TensorFlow [99]. We have studied [100] different parallel patterns (kernels) of machine learning applications, looking in particular at Gibbs Sampling [101], Stochastic Gradient Descent (SGD) [102], Cyclic Coordinate Descent (CCD) [103] and K-means clustering [104, 105, 106]. These algorithms are fundamental for large-scale data analysis and cover several important categories: Markov Chain Monte Carlo (MCMC), Gradient Descent and Expectation and Maximization (EM). We show that parallel iterative algorithms can be categorized into four types of computation models (a) Locking, (b) Rotation, (c) Allreduce, (d) Asynchronous, based on the synchronization patterns and the effectiveness of the model parameter update. A major challenge of scaling is

owing to the fact that computation is irregular and the model size can be huge. At the meantime, parallel workers need to synchronize the model continually. By investigating collective vs. asynchronous methods of the model synchronization mechanisms, we discover that optimized collective communication can improve the model update speed, thus allowing the model to converge faster. The performance improvement derives not only from accelerated communication but also from reduced iteration computation time as the model size may change during the model convergence. To foster faster model convergence, we need to design new collective communication abstractions. We identify all 5 classes of data-intensive computation[2], from pleasingly parallel to machine learning and simulations. To re-design a modular software stack with native kernels to effectively utilize scale-up servers for machine learning and data analytics applications. We are investigating how simulations and Big Data can use common programming environments with a runtime based on a rich set of collectives and libraries for a model-centric approach [107, 108].

5.2 Parallel Computing

We know that heterogeneity can lead to difficulty in parallel computing. This is extreme for MLaroundHPC as the ML learnt result can be huge factors (10^5 in our initial example[77]) faster than simulated answers. Further learning can be dynamic within a job and within different runs of a given job. One can address by load balancing the unlearnt and learnt separately but this can lead to geometric issues as quite likely that ML learning works more efficiently (for more potential simulations) in particular regions of phase space.

5.3 Scaling of Effective Performance

An initial approach to estimate speedup in a hybrid MLaroundHPC situation is given in [77] for a nano simulation. One can estimate the speedup in terms in terms of four times T_{seq} the sequential execution time of simulation; T_{train} the time for the parallel execution of simulation to give training data; T_{learn} is the time per sample to train the learning networkl; and T_{lookup} is the inference time to predict the results of the simulation by using the trained network. In the formula below, N_{lookup} is the number of trained neural net inferences and N_{train} the number of parallel simulations used in training.

$$\text{Effective Speedup } S = \frac{T_{seq}(N_{lookup} + N_{train})}{T_{lookup}N_{lookup} + (T_{train} + T_{learn})N_{train}}$$

This formula reduces to the classic simple $\frac{T_{seq}}{T_{train}}$ when there is no machine learning and in the limit of large $\frac{N_{lookup}}{N_{train}}$ becomes $\frac{T_{seq}}{T_{lookup}}$ which can be huge!

There are many caveats and assumptions here. We are considering a simple case where one runs the N_{train} simulations, followed by the learning and then all the N_{lookup} inferences. Further we assume the training simulations are useful results and not just overhead. We also have not properly considered how to build in the likelihood that training, learning and lookup phases are probably using different hardware configurations with different node counts.

6 Uncertainty Quantification for Deep Learning

An important aspect of the use of a learning ANN is that one must learn not just the result of a simulation but also the uncertainty e.g. is the learned result at all valid and is accuracy sufficient to be used. Here there are clear strategies such as if calculating X-Y and X similar in size to Y, then

can only learn X-Y NOT X and Y separately. Further, use of reduced precision in DL networks needs to be re-examined for our application.

The bias-variance trade-off is a well-studied concept in statistical learning that can provide a principled way to design a machine learning model concerning its complexity given the optimization problem and the dataset [109]. It is based on the decomposition of the expected error that a model can make into two parts: variance and bias. The variance part explains the uncertainty of a model due to the randomness in the training algorithms (e.g. random initialization of parameters or a different choice of hyperparameters) or the lack of representativeness of the subsampled training set used in training. One can reduce the variance of the expected error by introducing a regularization scheme during the optimization process so that the model complexity is in control and can result in a *smoother* model. However, the regularization approach comes at the cost of an increased amount bias, which is another term in the expected error decomposition that explains the fitness of the model to the currently available dataset. By reducing the flexibility of the model using a regularization scheme, the training algorithm can only do a limited effort to minimize the training error. On the contrary, an unregularized model with a higher model complexity than necessary can also result in a minimal training error in that training session, while it suffers from high variance or uncertainty in its prediction.

Ideally, the bias-variance trade-off can be resolved to some degree by averaging the trained instances of a complex model. As the variance term explains the statistical deviations of the different model predictions from their expected value, one can reduce it by training many model instances and average out their variations. Once these model instances are complex enough to fit the training dataset, we can use the averaged predictions as the output of the model. However, averaging many different model instances implies a practical difficulty that one has to conduct multiple optimization tasks to come up with a statistically meaningful sample distribution of the predictions. Given the assumption that the model might as well be a complex one to minimize the bias component (e.g. a deep neural network), model averaging is computationally challenging. Also, for a real-world problem that needs as much labeled dataset as possible, subsampling might not be the best way to make use of the data.

Dropout has been extensively used in deep learning as a regularization technique [110, 111, 112, 113], but recent researches revisit it as an uncertainty quantification (UQ) tool as well [114, 33]. The use of dropout as a tool for acquiring the sample distribution of predictions can be justified by the fact that the dropout procedure on a deep neural network (DNN) can be seen as an efficient way to maintain a pool of multiple network instances for the same optimization task. The efficiency of the procedure comes from the fact that the dropout technique applies a Bernoulli mask to the layer-wise feature vector, a stochastic structure alteration process, to each of the feedforward routines under the stochastic gradient descent training framework. Thus, the training process has the same effect as if it is exposed to many network instances although training is on only one instance. Dropout has shown promising performances in many deep learning benchmarks especially thanks to its ability to prevent overfitting. Although the original usage of dropout has been limited to the regularization during training, recent researches suggest its use during the test time as well. As the network trained with the dropout technique is robust to the structural variation, during the test time one can use the same masking strategy to come up with a set of differently thinned versions of the network, which can form a sample distribution of predictions. Hence, the access to the distribution of the predictions enables one to use it as a UQ metric of the trained deep learning system.

The dropout-based UQ scheme can provide an opportunity for the MLaroundHPC simulation experiments. As deep learning is to train a model in a data-driven fashion, it is obvious that a better ML surrogate can be found once the training routine sees more examples. Usually, the craving for

more data in large-scale machine learning tasks causes a lack of data, as the ML applications that try to catch up human perception in classification tasks, e.g. visual object recognition, require the data examples to be labeled by humans. Crowdsourcing can reduce the cost of generating such a labeled dataset [115]. However, as for modeling the large-scale simulation results, the ML model has to see more examples generated from the HPC simulation, which is a conflicting requirement as the purpose of training ML is to avoid such computation. The UQ scheme can play a role here to provide the training routine with a way to quantify the uncertainty in the prediction—once it is low enough, the training routine might less likely need more data.

7 Machine Learning Requirements for MLforHPC

Here we review the nature of the Machine Learning needed for MLforHPC in different application domains. The Machine Learning (ML) load depends on 1) Time interval between its invocations, which will translate into the number of training samples S and 2) size D of data set specifying each sample. This size could be as large as the number of degrees of freedom in simulation or could be (much) smaller if just a few parameters are needed to define simulation. We note two general issues

- There can very important data transfer and storage issues in linking the Simulations and Machine Learning parts of system. This could need carefully designed architectures for both hardware and software.
- The Simulations and Machine Learning subsystems are likely to require different node optimizations as in different types and uses of accelerators.

7.1 Nanosimulations

In each of two cases below, one is using scikit-learn, Tensorflow and the Keras wrapper for Tensorflow, as the ML subsystem. The papers [77, 116] are using ML to learn results (ionic density at a given location) of a complete simulation

- D=5 with the five specifying features as connement length h, positive valency z_p, negative valency z_n, salt concentration c, and the diameter of the ions d.
- S= 4805 which 70% of total 6864 runs with 30% of the total runs used for testing.

In [23], one is not asking ML to predict a result as in [77],but rather training an Artificial Neural Net (ANN) to ensure that the simulation runs at its optimal speed (using for example, the lowest allowable timestep dt and "good" simulation control parameters for high efficiency) while retaining the accuracy of the final result (e.g. density profile of ions). For this particular application, we could get away by dividing a 10 million time-step run (10 nanoseconds that is a typical timescale to reach equilibrium and get data in such systems) into 10 separate runs.

- Input data size D= 6 (1 input uses 64 bits floats and 5 inputs use 32 bits integers - total 224 bits)
- Input number of samples (S) = 15640 (70% training 30% test)
- Hidden layer 1 = 30
- Hidden layer 1 = 48
- Output variables = 3

Creation of the training dataset took = 64 cores * 80 hrs * 5400 simulation runs = 28160000 or 28 million CPU hours on Indiana University's BigRed2 GPU compute nodes. Each run is 10

million steps long, and you use/learn/train ML every 1 million steps (so block size is a million), yielding 10 times more samples than runs.

Generalizing this, the hardware needs will depend on how often you block, to stop and train the network, and then either on-the-fly or post-simulation, use that training to accelerate simulation or evaluate structure respectively. Blocking every timestep will not improve the training as typically, it won't produce a statistically independent data point to evaluate any structure you desire. So you want to block at a timescale that is at least greater than the autocorrelation time dc; this is, of course, dependent on example you are looking at – and so your blocking and learning will depend on the application. In [77], it is small and dc is 3-5 dt; in glasses, it can be huge as the viscosity is high; and in biomolecular simulations, it will also depend on the level of coarse-graining and will be different in fully atomistic or very coarse-grained systems.

The training effort will also depend on the input data size D, and the complexity of the relationship you are trying to learn which change the number of hidden layers and nodes per layer. For example, suppose you are tracking a particle (a side atom on a molecule in a typical nanoscale simulation), in order to come up with a metric (e.g. distance between two side atoms on different molecules) to track the diversity of clusters of particles during the self-assembly process. This comes from expectation that correlations between side atoms may be critical to a macroscopic property (such as formation of these particles into a FCC crystal). In this case your D is huge, and your ML objectives may be looking for a deep relationship, and you may have to invoke an ensemble of ANN's and this will change hardware needs.

8 New Opportunities and Research Issues

8.1 Broken Abstractions, New Abstractions:

In traditional HPC the prevailing orthodoxy is Faster is Better which has driven the quest for abstractions of hierarchical parallelism to speeding up single units of works. However the new paradigm in HPC — Learning Everywhere, implies new performance, scaling and execution approaches. In this new paradigm, multiple, concurrent heterogeneous units of work replace single large units of works, which thus require both hierarchical (vertical) parallelism as well horizontal parallelism Relinquishing the orthodoxy based upon hierarchical (vertical) parallelism as the only route to performance is necessary while supporting the scaling requirements of "Learning Everywhere". Our distinction between traditional performance and the effective performance achieved by using learnt results is important here.

8.2 Research Issues in MLforHPC

Research issues reflect the multiple interdisciplinary activities linked in our study of MLforHPC.

- MLforHPC Cyberinfrastructure
- ML and Deep Learning
- Uncertainty Quantification
- Multiple application domains described in sections 2, 3, 4.1 and 4.2
- Coarse graining studied in our case for network science and nano-bio areas

We have identified the following research areas.

1. Where can application domains use MLaroundHPC and MLautotuning effectively and what science is enabled by this

2. Which ML and DL approaches are most relevant and how can they be set up to enable broad user-friendly MLaroundHPC and MLautotuning in domain science

3. How can Uncertainty Quantification be enabled and separately study ergodicity (bias) and accuracy issues?

4. Is there new area of algorithmic research focusing on finding algorithms that can be most effectively learnt?

5. Is there a general multiscale approach using MLaroundHPC.

6. What are appropriate systems frameworks for MLaroundHPC and MLautotuning. For example, should we wrap microservices invoked by a Function as a Service environment? Where and how should we enable learning systems? Is Dataflow useful?

7. The different characters of surrogate and real executions produce system challenges as surrogate execution is much faster and invokes distinct software and hardware. This heterogeneity gives challenges for parallel computing, workload management and resource scheduling (heterogeneous and dynamic workflows). The implication for performance is briefly discussed in sections 5.2 and 5.3.

8. Scaling applications that are composed of multiple heterogeneous computational (execution) units, and have distinct forms of parallelism that need balanced performance. Consider a workload comprised of N_L learning units, N_S simulations units. The relative number of learning units to simulation units will vary with application and problem type. The relative values will even vary over execution time of the application, as the amount of data generated as a ratio of training data will vary. This requires runtime systems that are capable of real-time performance tuning and adaptive execution for workloads comprised of multiple heterogeneous tasks.

9. The application of these ideas to statistical physics problems may need different techniques than those used in deterministic time evolutions.

10. The existing UQ frameworks based on the dropout technique can provide the level of certainty as a probabilistic distribution in the prediction space. However, it does not always mean that the quality of the distribution is dependent on the quality/quantity of data. For example, two models with different dropout rates can produce different UQ results. If the goal of UQ in MLaroundHPC context is to supply only an adequate amount of data, we need a more reliable UQ method tailored for this purpose rather than the dropout technique that tends to manipulate the architecture of the model.

11. Application agnostic description and defintion of effective performance enhancement.

12. Sean Blanchard [117] has emphasized the potential value of learnt information in fault tolerance where for example learnt results could be used to restart a crashed compute node.

The above Research Areas can be categorized into Algorithms and Methods (1-5), Applied Math (10), Software Systems (6,7), Performance Measurment and Engineering (8,11),

8.3 Possible Dataflow MLforHPC System Architecture

We have advocated in [118] a prototype system approach to MLforHPC that is based on the dataflow representation of the simulation or ML problem where the idea is to support ML wrapping of any dataflow node. This dataflow representation is seen in Big Data systems such as Storm, Heron, Spark, and Flink. We will allow coarse-grain dataflow with dataflow nodes at the job level; this is familiar from workflow systems such as Kepler, Pegasus, and NiFi and illustrated by Persona's [119] integration of Tensorflow into bioinformatics pipelines. We will also allow finer grain size dataflow as seen in modern big data systems and Function as a Service architectures and provide a cleaner architecture with a single approach to ML integration at all granularity's. In this discussion, dataflow can be also be considered as a task graph. The chemistry potential example shows a case where the surrogate replaces a fine grain node. The Indiana University Twister2 system [120, 121, 122] supports such a hybrid coarse/fine grain dataflow graph specifying a problem. We intend the same architecture to support both MLaroundHPC and MLautotuning. As emphasized by Sharma [123], one should explore a range of synchronization and data consistency techniques [124] that have been proposed for distributed ML training. These techniques differ from BSP and strong-consistency, and offer a more relaxed consistency model by exploiting and tolerating staleness. We can explore the appropriate consistency model for integrating ML with conventional HPC applications. Examples have been shown [125, 126, 127] of the successful use of relaxed consistency for certain HPC applications including our Harp project [107, 108] where the trade-off of the staleness of the model versus synchronization cost was explored.

Acknowledgments

This work was partially supported by NSF CIF21 DIBBS 1443054 and nanoBIO 1720625; the Indiana University Precision Health initiative and Intel through the Parallel Computing Center at Indiana University. JPS and JAG were partially supported by NSF 1720625, NIH U01 GM111243 and NIH GM122424. SJ was partially supported by ExaLearn – a DOE Exascale Computing project.

References

[1] Geoffrey Fox, James A. Glazier, JCS Kadupitiya, Vikram Jadhao, Minje Kim, Judy Qiu, James P. Sluka, Endre Somogyi, Madhav Marathe, Abhijin Adiga, Jiangzhuo Chen, Oliver Beckstein, and Shantenu Jha. Learning Everywhere: Pervasive machine learning for effective High-Performance computation. Technical report, Indiana University, February 2019. `https://arxiv.org/abs/1902.10810,http://dsc.soic.indiana.edu/publications/Learning_Everywhere_Summary.pdf`.

[2] Geoffrey Fox, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve. Big data, simulations and HPC convergence. In *Springer Lecture Notes in Computer Science LNCS 10044*, 2016.

[3] Dennard scaling. `https://en.wikipedia.org/wiki/Dennard\_scaling`.

[4] Peter M Kasson and Shantenu Jha. Adaptive ensemble simulations of biomolecules. *Current Opinion in Structural Biology*, 52:87 – 94, 2018. Cryo electron microscopy: the impact of the cryo-EM revolution in biology Biophysical and computational methods - Part A.

[5] Oliver Beckstein, Geoffrey Fox, Shantenu Jha. Convergence of data generation and analysis in the biomolecular simulation community. In Terry Moore, Geoffrey Fox, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[6] Bill Tang. New models for integrated inquiry: Fusion energy exemplar. In Geoffrey Fox Terry Moore, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[7] Logan Ward. Deep learning, HPC, and data for materials design. In Terry Moore, Geoffrey Fox, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[8] Kevin Yager. Autonomous experimentation as a paradigm for materials discovery. In Geoffrey Fox Terry Moore, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[9] José Miguel Hernández-Lobato, Klaus-Robert Müller, Brooks Paige, Matt J. Kusner, Stefan Chmiela, Kristof T. Schütt. Machine learning for molecules and materials. In *Proceedings of the NeurIPS 2018 Workshop*, December 2018.

[10] NSF1849625 workshop series BDEC2: Toward a common digital continuum platform for big data and extreme-scale computing (BDEC2). `https://www.exascale.org/bdec/`.

[11] Oliver Beckstein,Geoffrey Fox, Judy Qiu, David Crandall, Gregor von Laszewski, John Paden, Shantenu Jha, Fusheng Wang, Madhav Marathe, Anil Vullikanti, Thomas Cheatham. Contributions to High-Performance big data computing. Technical report, Digital Science Center, September 2018.

[12] Geoffrey Fox, David Crandall, Judy Qiu, Gregor Von Laszewski, Shantenu Jha, John Paden, Oliver Beckstein, Tom Cheatham, Madhav Marathe, and Fusheng Wang. NSF 1443054: CIF21 DIBBs: Middleware and high performance analytics libraries for scalable data science, poster. In *DIBBS PI meeting*, June 2018.

[13] Jeff Dean. Machine learning for systems and systems for machine learning. In *Presentation at 2017 Conference on Neural Information Processing Systems*, 2017.

[14] Satoshi Matsuoka. Post-K: A game changing supercomputer for convergence of HPC and big data / AI. Multicore 2019, February 2019.

[15] Francis J. Alexander, Shantenu Jha. Objective driven computational experiment design: An ExaLearn perspective. In Terry Moore, Geoffrey Fox, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[16] Anuj Karpatne, Gowtham Atluri, James Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Trans. Knowl. Data Eng.*, 29(10):2318–2331, October 2017.

[17] Shing Chan and Ahmed H Elsheikh. A machine learning approach for efficient uncertainty quantification using multiscale methods. *J. Comput. Phys.*, 354:493–511, February 2018.

[18] Rohit K Tripathy and Ilias Bilionis. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *J. Comput. Phys.*, 375:565–588, December 2018.

[19] Stephan Rasp, Michael S Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proc. Natl. Acad. Sci. U. S. A.*, 115(39):9684, September 2018.

[20] Wolfgang Gentzsch. Deep learning for fluid flow prediction in the cloud. `https://www.linkedin.com/pulse/deep-learning-fluid-flow-prediction-cloud-wolfgang-gentzsch/`, December 2018. Accessed: 2019-3-1.

[21] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[22] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv*, November 2017.

[23] JCS Kadupitiya, Geoffrey C. Fox, Vikram Jadhao. Machine learning for parameter auto-tuning in molecular dynamics simulations: Efficient dynamics of ions near polarizable nanoparticles. Technical report, Indiana University, November 2018.

[24] Microsoft Research. AI for database and data analytic systems at microsoft faculty summit. `https://youtu.be/Tkl6ERLWAbA`, 2018. Accessed: 2019-1-29.

[25] Microsoft Research. AI for AI systems at microsoft faculty summit. `https://youtu.be/MqBOuoLflpU`, 2018. Accessed: 2019-1-29.

[26] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 489–504, New York, NY, USA, 2018. ACM.

[27] Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Kärnä, Diana Moise, Simon J Pennycook, Kristyn Maschhoff, Jason Sewall, Nalini Kumar, Shirley Ho, Michael F Ringenburg, Prabhat, and Victor Lee. CosmoFlow: Using deep learning to learn the universe at scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis SC18*, SC '18, pages 65:1–65:11, Piscataway, NJ, USA, 2018. IEEE Press.

[28] Mustafa Mustafa, Deborah Bard, Wahid Bhimji, Rami Al-Rfou, and Zarija Luki. Creating virtual universes using generative adversarial networks. Technical report, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, 06 2017.

[29] A Townsend Peterson, Monica Pape\cs, and Jorge Soberón. Mechanistic and correlative models of ecological niches. *European Journal of Ecology*, 1(2):28–38, 2015.

[30] Ruth E Baker, Jose-Maria Peña, Jayaratnam Jayamohan, and Antoine Jérusalem. Mechanistic models versus machine learning, a fight worth fighting for the biological community? *Biology letters*, 14(5):20170660, 2018.

[31] Anuj Karpatne, Gowtham Atluri, James H. Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

[32] Mark EJ Newman. Spread of epidemic disease on networks. *Physical review E*, 66(1):016128, 2002.

[33] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, Cambridge University, 2017.

[34] Balaji Lakshminarayanan, Alexander Pritzel, Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 6405–6416, 2017.

[35] Abhijin Adiga, Chris J Kuhlman, Henning S Mortveit, and Anil Kumar S Vullikanti. Sensitivity of diffusion dynamics to network uncertainty. *Journal of Artificial Intelligence Research*, 51:207–226, 2014.

[36] Mayank Lahiri, Arun S Maiya, Rajmonda Sulo, Tanya Y Berger Wolf, et al. The impact of structural changes on predictions of diffusion in networks. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 939–948. IEEE, 2008.

[37] Lijing Wang, J Chen, and Madhav Marathe. DEFSI : Deep Learning Based Epidemic Forecasting with Synthetic Information. Technical Report 1, University of Virginia, 2018.

[38] Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-Based Model Calibration using Machine Learning Surrogates. *Paper-Progress*, pages 1–32, 2017.

[39] Batuhan Bardak and Mehmet Tan. Prediction of influenza outbreaks by integrating wikipedia article access logs and google flu trend data. In *Bioinformatics and Bioengineering (BIBE), 2015 IEEE 15th International Conference on*, pages 1–6. IEEE, 2015.

[40] Michael A Benjamin, Robert A Rigby, and D Mikis Stasinopoulos. Generalized autoregressive moving average models. *Journal of the American Statistical association*, 98(461):214–223, 2003.

[41] Andrea Freyer Dugas, Mehdi Jalalpour, Yulia Gel, Scott Levin, Fred Torcaso, Takeru Igusa, and Richard E Rothman. Influenza forecasting with google flu trends. *PloS one*, 8(2):e56176, 2013.

[42] Zheng Wang, Prithwish Chakraborty, Sumiko R Mekaru, John S Brownstein, Jieping Ye, and Naren Ramakrishnan. Dynamic poisson autoregression for influenza-like-illness case count prediction. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294. ACM, 2015.

[43] Shihao Yang, Mauricio Santillana, and Samuel C Kou. Accurate estimation of influenza epidemics using google search data via argo. *Proceedings of the National Academy of Sciences*, 112(47):14473–14478, 2015.

[44] Svitlana Volkova, Ellyn Ayton, Katherine Porterfield, and Courtney D Corley. Forecasting influenza-like illness dynamics for military populations using neural networks and social media. *PloS one*, 12(12):e0188941, 2017.

[45] Siva R Venna, Amirhossein Tavanaei, Raju N Gottumukkala, Vijay V Raghavan, Anthony S Maida, and Stephen Nichols. A novel data-driven model for real-time influenza forecasting. *IEEE Access*, 2018.

[46] Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. Deep learning for epidemiological predictions. In *SIGIR*, pages 1085–1088, 2018.

[47] Ashleigh R Tuite, Amy L Greer, Michael Whelan, Anne-Luise Winter, Brenda Lee, Ping Yan, Jianhong Wu, Seyed Moghadas, David Buckeridge, Babak Pourbohloul, et al. Estimated epidemiologic parameters and morbidity associated with pandemic h1n1 influenza. *Canadian Medical Association Journal*, 182(2):131–136, 2010.

[48] Wan Yang, Alicia Karspeck, and Jeffrey Shaman. Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics. *PLoS computational biology*, 10(4):e1003583, 2014.

[49] Elaine O Nsoesie, Richard J Beckman, Sara Shashaani, Kalyani S Nagaraj, and Madhav V Marathe. A simulation optimization approach to epidemic forecasting. *PloS one*, 8(6):e67164, 2013.

[50] Ting Hua, Chandan K Reddy, Lei Zhang, Lijing Wang, Liang Zhao, Chang-Tien Lu, Naren Ramakrishnan, and Virginia Tech. Social media based simulation models for understanding disease dynamics. In *IJCAI*, pages 3797–3804, 2018.

[51] Liang Zhao, Jiangzhuo Chen, Feng Chen, Wei Wang, Chang-Tien Lu, and Naren Ramakrishnan. Simnest: Social media nested epidemic simulation via online semi-supervised deep learning. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 639–648. IEEE, 2015.

[52] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[53] G Wayne Brodland. How computational models can help unlock biological systems. *Semin. Cell Dev. Biol.*, 47-48:62–73, December 2015.

[54] James M Osborne, Alexander G Fletcher, Joe M Pitt-Francis, Philip K Maini, and David J Gavaghan. Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLoS Comput. Biol.*, 13(2):e1005387, February 2017.

[55] Paul Macklin, Mary E Edgerton, Alastair M Thompson, and Vittorio Cristini. Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): from microscopic measurements to macroscopic predictions of clinical progression. *J. Theor. Biol.*, 301:122–140, May 2012.

[56] Hermann-Georg Holzhütter, Dirk Drasdo, Tobias Preusser, Jörg Lippert, and Adriano M Henney. The virtual liver: a multidisciplinary, multilevel challenge for systems biology. *Wiley Interdiscip. Rev. Syst. Biol. Med.*, 4(3):221–235, May 2012.

[57] James P. Sluka, Xiao Fu, Maciej Swat, Julio M. Belmonte, Alin Cosmanescu, Sherry G. Clendenon, John F. Wambaugh, and James A. Glazier. A liver-centric multiscale modeling framework for xenobiotics. *PLoS ONE*, 11(9), 2016.

[58] Xiao Fu, James P Sluka, Sherry G Clendenon, Kenneth W Dunn, Zemin Wang, James E Klaunig, and James A Glazier. Modeling of xenobiotic transport and metabolism in virtual hepatic lobule models. *PLoS ONE*, 13(9):e0198060, September 2018.

[59] Natalia A Trayanova. Whole-heart modeling: applications to cardiac electrophysiology and electromechanics. *Circ. Res.*, 108(1):113–128, January 2011.

[60] Francois Nedelec and Dietrich Foethke. Collective langevin dynamics of flexible cytoskeletal fibers. *New J. Phys.*, 9(11):427, November 2007.

[61] Michael Pargett and David M Umulis. Quantitative model analysis with diverse biological data: applications in developmental pattern formation. *Methods*, 62(1):56–67, July 2013.

[62] Ahmadreza Ghaffarizadeh, Randy Heiland, Samuel H Friedman, Shannon M Mumenthaler, and Paul Macklin. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. *PLoS Comput. Biol.*, 14(2):e1005991, February 2018.

[63] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations. *arXiv*, November 2017.

[64] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1:S7, March 2006.

[65] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, December 2018.

[66] Liang Liang, Minliang Liu, Caitlin Martin, John A Elefteriades, and Wei Sun. A machine learning approach to investigate the relationship between shape features and numerically predicted risk of ascending aortic aneurysm. *Biomech. Model. Mechanobiol.*, 16(5):1519–1533, October 2017.

[67] Guangming Luo, Sarka Malkova, Jaesung Yoon, David G. Schultz, Binhua Lin, Mati Meron, Ilan Benjamin, Petr Vansek, and Mark L. Schlossman. Ion distributions near a liquid-liquid interface. *Science*, 311(5758):216–218, 2006.

[68] Yufei Jing, Vikram Jadhao, Jos W Zwanikken, and Monica Olvera de la Cruz. Ionic structure in liquids confined by dielectric interfaces. *The Journal of chemical physics*, 143(19):194508, 2015.

[69] Alexander M Smith, Alpha A Lee, and Susan Perkin. The electrostatic screening length in concentrated electrolytes increases with concentration. *The journal of physical chemistry letters*, 7(12):2157–2163, 2016.

[70] Dezsö Boda, Dirk Gillespie, Wolfgang Nonner, Douglas Henderson, and Bob Eisenberg. Computing induced charges in inhomogeneous dielectric media: Application in a monte carlo simulation of complex ionic systems. *Phys. Rev. E*, 69(4):046702, Apr 2004.

[71] Kipton Barros, Daniel Sinkovits, and Erik Luijten. Efficient and accurate simulation of dynamic dielectric objects. *The Journal of Chemical Physics*, 140(6):064903, 2014.

[72] Sandeep Tyagi, Mehmet Suzen, Marcello Sega, Marcia Barbosa, Sofia S. Kantorovich, and Christian Holm. An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries. *The Journal of Chemical Physics*, 132(15):154112, 2010.

[73] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm. ESPResSo – an extensible simulation package for research on soft matter systems. *Comp. Phys. Comm.*, 174(9):704–727, May 2006.

[74] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995.

[75] Jos W. Zwanikken and Monica Olvera de la Cruz. Tunable soft structure in charged fluids confined by dielectric interfaces. *Proceedings of the National Academy of Sciences*, 110(14):5301–5308, 2013.

[76] Alexandre P dos Santos and Roland R Netz. Dielectric boundary effects on the interaction between planar charged surfaces with counterions only. *The Journal of chemical physics*, 148(16):164103, 2018.

[77] JCS Kadupitiya , Geoffrey C. Fox , and Vikram Jadhao. Machine learning for performance enhancement of molecular dynamics simulations. Technical report, Indiana University, December 2018.

[78] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, July 2018.

[79] Matthew Spellings and Sharon C Glotzer. Machine learning for crystal identification and discovery. *AIChE Journal*, 64(6):2198–2206, 2018.

[80] Samuel S. Schoenholz. Combining machine learning and physics to understand glassy systems. *Journal of Physics: Conference Series*, 1036(1):012021, 2018.

[81] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning monte carlo method. *Phys. Rev. B*, 95:041101, Jan 2017.

[82] Alan Morningstar and Roger G Melko. Deep learning the ising model near criticality. *arXiv preprint arXiv:1708.04622*, 2017.

[83] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami. Machine learning phases of strongly correlated fermions. *Phys. Rev. X*, 7:031038, Aug 2017.

[84] Venkatesh Botu and Rampi Ramprasad. Adaptive machine learning framework to accelerate ab initio molecular dynamics. *International Journal of Quantum Chemistry*, 115(16):1074–1083, 2015.

[85] Andrew L Ferguson. Machine learning and data science in soft materials engineering. *J. Phys. Condens. Matter*, 30(4):043002, January 2018.

[86] Ashley Z Guo, Emre Sevgen, Hythem Sidky, Jonathan K Whitmer, Jeffrey A Hubbell, and Juan J de Pablo. Adaptive enhanced sampling by force-biasing using neural networks. *The Journal of chemical physics*, 148(13):134108, 2018.

[87] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2:16028, 2016.

[88] Jrg Behler. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems. *Angewandte Chemie International Edition*, 56(42):12828–12840, 2017.

[89] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547, July 2018.

[90] Jrg Behler and Michele Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Physical Review Letters*, 98(14):146401, April 2007.

[91] Michael Gastegger, Jrg Behler, and Philipp Marquetand. Machine learning molecular dynamics for the simulation of infrared spectra. *Chemical Science*, 8(10):6924–6935, 2017.

[92] J. S.Smith, O. Isayev, and A. E.Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science*, 8(4):3192–3203, 2017.

[93] Justin S. Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E. Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24):241733, May 2018.

[94] Justin S Smith, Benjamin T. Nebgen, Roman Zubatyuk, Nicholas Lubbers, Christian Devereux, Kipton Barros, Sergei Tretiak, Olexandr Isayev, and Adrian Roitberg. Outsmarting Quantum Chemistry Through Transfer Learning. *chemrxiv*, doi: 10.26434/chemrxiv.6744440.v1, July 2018.

[95] Siewert J Marrink and D Peter Tieleman. Perspective on the Martini model. *Chem Soc Rev*, 42(16):6801–22, Aug 2013.

[96] W G Noid. Perspective: Coarse-grained models for biomolecular systems. *J Chem Phys*, 139(9):090901, Sep 2013.

[97] Eliodoro Chiavazzo, Roberto Covino, Ronald R. Coifman, C. William Gear, Anastasia S. Georgiou, Gerhard Hummer, and Ioannis G. Kevrekidis. Intrinsic map dynamics exploration for uncharted effective free-energy landscapes. *Proceedings of the National Academy of Sciences*, 114(28):E5494–E5503, 2017.

[98] MLPERF benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms. `https://mlperf.org/`. Accessed: 2019-2-8.

[99] Uber Engineering. Horovod: Uber's open source distributed deep learning framework for TensorFlow. `https://eng.uber.com/horovod/`. Accessed: 2019-2-8.

[100] Intel® parallel computing center at indiana university led by Judy Qiu. `http://ipcc.soic.iu.edu/`. Accessed: 2018-9-30.

[101] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proc. Natl. Acad. Sci. U. S. A.*, 101 Suppl 1:5228–5235, April 2004.

[102] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 69–77, New York, NY, USA, 2011. ACM.

[103] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 765–774, Washington, DC, USA, 2012. IEEE Computer Society.

[104] Bo Peng, Bingjing Zhang, Langshi Chen, Mihai Avram, Robert Henschel, Craig Stewart, Shaojuan Zhu, Emily Mccallum, Lisa Smith, Tom Zahniser, Jon Omer, and Judy Qiu. HarpLDA+: Optimizing latent dirichlet allocation for parallel efficiency. In *10th IEEE International Conference on Cloud Computing (IEEE Cloud 2017), June 25-30, 2017 (IEEE Big Data 2017).*, August 2017.

[105] Langshi Chen, Bo Peng, Bingjing Zhang, Tony Liu, Yiming Zou, Lei Jiang, Robert Henschel, Craig Stewart, Zhang Zhang, Emily Mccallum, Zahniser Tom, Omer Jon, and Judy Qiu. Benchmarking Harp-DAAL: High performance hadoop on KNL clusters. In *IEEE Cloud 2017 Conference*. IEEE, 2017.

[106] Langshi Chen, Bo Peng, Sabra Ossen, Anil Vullikanti, Madhav Marathe, Lei Jiang, and Judy Qiu. High-Performance massive subgraph counting using pipelined Adaptive-Group communication. `http://arxiv.org/abs/1804.09764`, April 2018.

[107] Bingjing Zhang, Bo Peng, and Judy Qiu. Parallelizing big data machine learning applications with model rotation. In Fox, G., Getov, V., Grandinetti, L., Joubert, G.R., Sterling, T., editor, *Advances in Parallel Computing: New Frontiers in High Performance Computing and Big Data*. IOS Press, 2017.

[108] Intel parallel universe issue 32 page 31: Judy Qiu, Harp-DAAL for High-Performance big data computing. `https://software.intel.com/sites/default/files/parallel-universe-issue-32.pdf,http://dsc.soic.indiana.edu/publications/Intel-Magazine-HarpDAAL10.pdf`. Accessed: 2018-9-30.

[109] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[110] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[111] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

[112] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[113] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[114] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[115] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[116] JCS Kadupitiya , Geoffrey C. Fox , and Vikram Jadhao. Machine learning for auto-tuning of simulation parameters in Car-Parrinello molecular dynamics. In *Bulletin of the American Physical Society for March meeting, 2019*, 2019.

[117] Sean Blanchard. personal communication, February 2019.

[118] Geoffrey Fox. Computational and data intelligence linked in the intelligent aether for applications. In Terry Moore, Geoffrey Fox, editor, *Online Resource for Big Data and Extreme-Scale Computing Workshop*, November 2018.

[119] Stuart Byma, Sam Whitlock, Laura Flueratoru, Ethan Tseng, Christos Kozyrakis, Edouard Bugnion, and James Larus. Persona: A High-Performance bioinformatics framework. In *2017 {USENIX} Annual Technical Conference ({USENIX} {ATC} 17)*, pages 153–165, 2017.

[120] Geoffrey Fox. Twister2 tutorial at BigDat2019 cambridge UK january 8-11 2019. `https://twister2.gitbook.io/twister2/tutorial`. Accessed: 2019-1-30.

[121] Supun Kamburugamuve, Kannan Govindarajan, Pulasthi Wickramasinghe, Vibhatha Abeykoon, and Geoffrey Fox. Twister2: Design of a big data toolkit. In *EXAMPI 2017 workshop at SC17 conference*, 2017.

[122] Supun Kamburugamuve, Pulasthi Wickramasinghe, Kannan Govindarajan, Ahmet Uyar, Gurhan Gunduz, Vibhatha Abeykoon, Geoffrey Fox. Twister:Net - communication library for big data processing in HPC and cloud environments. In *Proceedings of Cloud 2018 Conference*. IEEE, 2018.

[123] Prateek Sharma. personal communication, February 2019.

[124] G Zhao, L Gao, and D Irwin. Sync-on-the-fly: A parallel framework for gradient descent algorithms on transient resources. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 392–397, December 2018.

[125] Zhigang Wang, Lixin Gao, Yu Gu, Yubin Bao, and Ge Yu. FSP: Towards flexible synchronous parallel framework for expectation-maximization based algorithms on cloud. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, pages 1–14, New York, NY, USA, 2017. ACM.

[126] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *NSDI*, pages 629–647, 2017.

[127] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, pages 1223–1231, 2013.