



ACM HPDC 2019, Phoenix, Arizona, USA

Perspectives on High-Performance Computing in a Big Data World - PartC

The 28th International Symposium on High-Performance Parallel and Distributed Computing

Geoffrey Fox | June 27, 2019

gcf@indiana.edu, <http://www.dsc.soic.indiana.edu/>, <http://spidal.org/>



ML around HPDC/HPC


ML Autotuning

MLforHPDC/HPC (ML for Systems) in detail

- **MLforHPDC/HPC** can be further subdivided into several categories:
 - ~~**ML after HPC**~~: ML analyzing results of HPC as in trajectory analysis and structure identification in biomolecular simulations. Well established and successful
 - ~~**ML Control**~~: Using simulations (with HPC) and ML in control of experiments and in objective driven computational campaigns. Here simulation surrogates are very valuable to allow real-time predictions.
 - **ML Autotuning**: Using ML to configure (autotune) ML or HPC simulations.
 - **ML around HPC**: Using ML to learn from simulations and produce learned surrogates for the simulations or parts of simulations. The same ML wrapper can also learn configurations as well as results. **Most Important.**

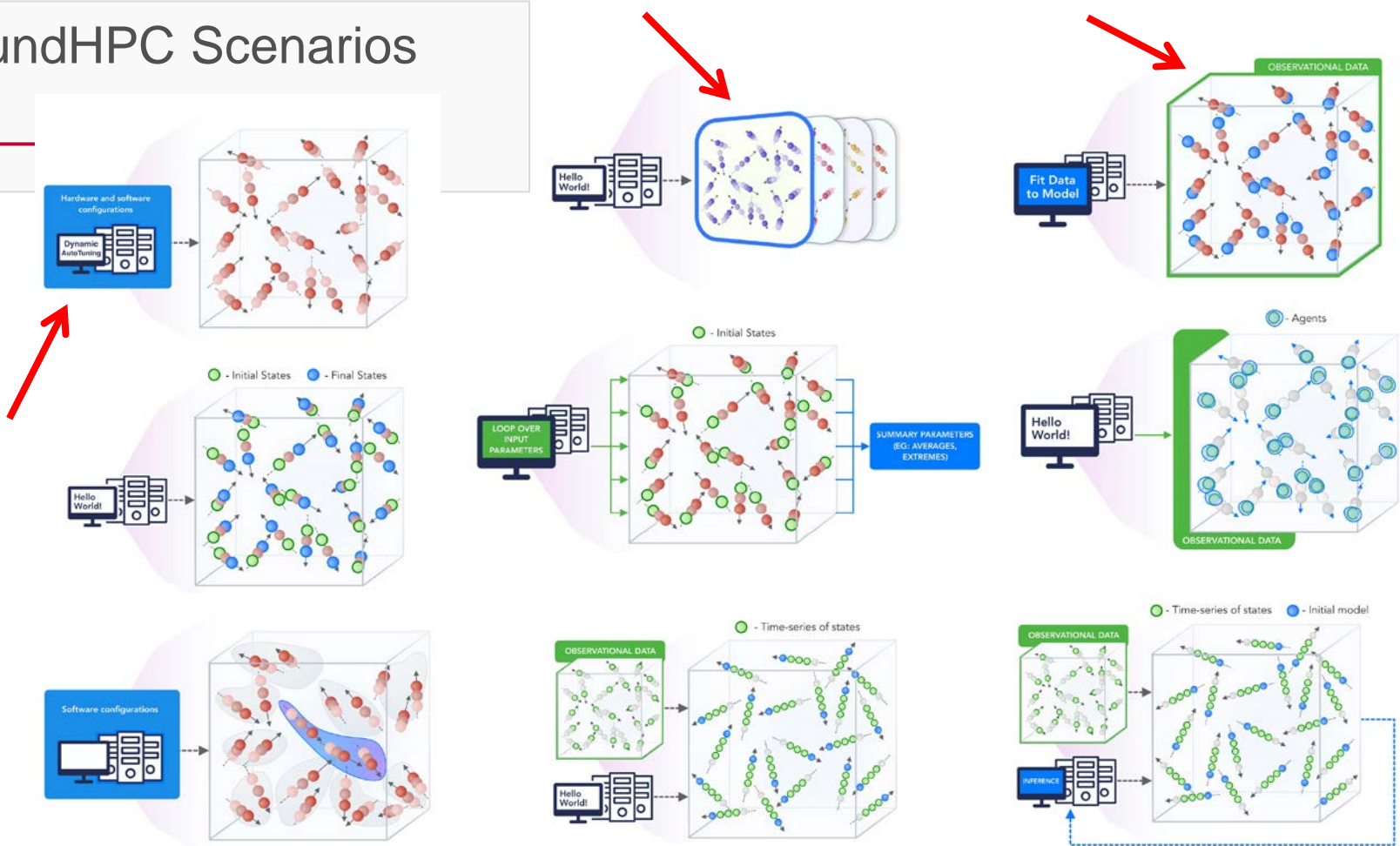
Status of MLforHPC Research

ML for HPDC or Systems

- <http://dsc.soic.indiana.edu/publications/Learning%20EverywhereResource.pdf> 
- 111 Citations (mainly 2017 or later) with very short comments
- MLaroundHPC/MLAutotuning: not much on **Computer Science** or **Partial Differential equations**
- **Particle Dynamics**: largest component with, smart sampling, effective potentials, “Computation Results from Computation defining Parameters” with “simple” deep learning replacing sophisticated dimension reduction; material science properties very active
 - Give examples from nanoparticle simulations
- **Agent-based Simulations** in networked systems or virtual tissues. Perhaps most promising as inevitably data driven as no fundamental equations for cells, cars, people, bacteria
 - Review plan to develop new approach to computational systems biology -- simulate organisms based on models for cell

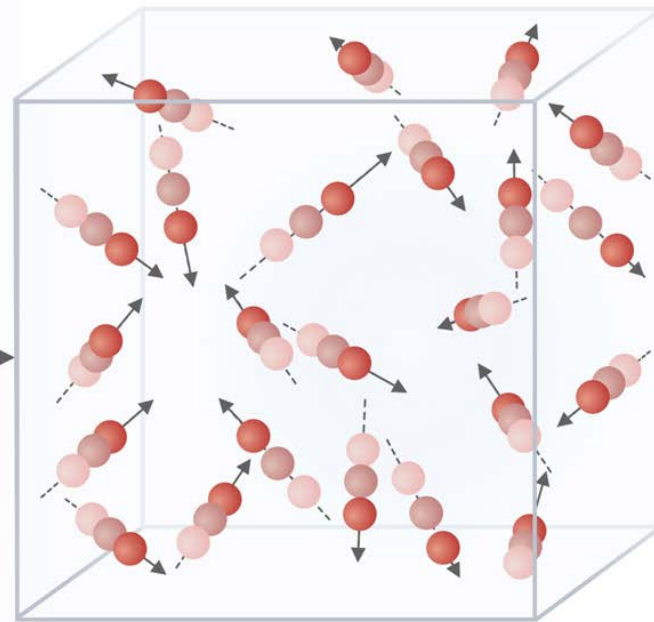
9 MLaroundHPC Scenarios

- INPUT
- OUTPUT



MLAutoTuningHPC: Learning Configurations

- This is classic Autotuning and one optimizes some mix of performance and quality of results with the learning network inputting the configuration parameters of the computation.
- This includes ~~initial values~~ and also ~~dynamic choices~~ such as block sizes for cache use, variable step sizes in space and time.
- It can also include discrete choices as to the type of solver to be used.

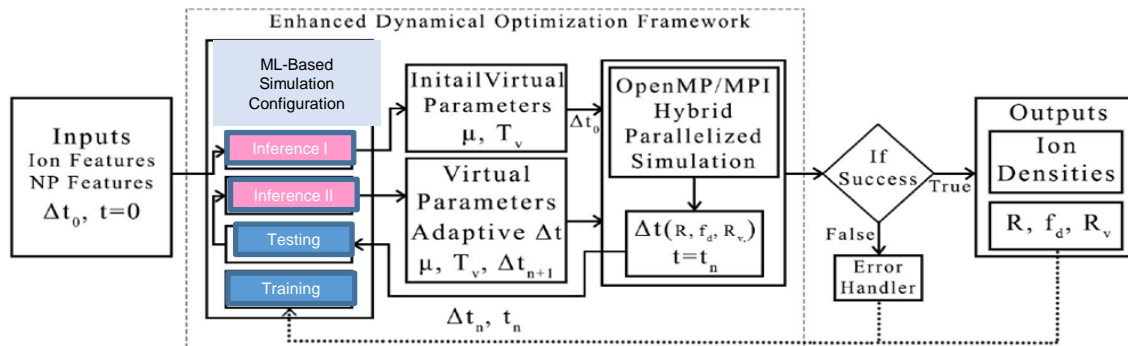
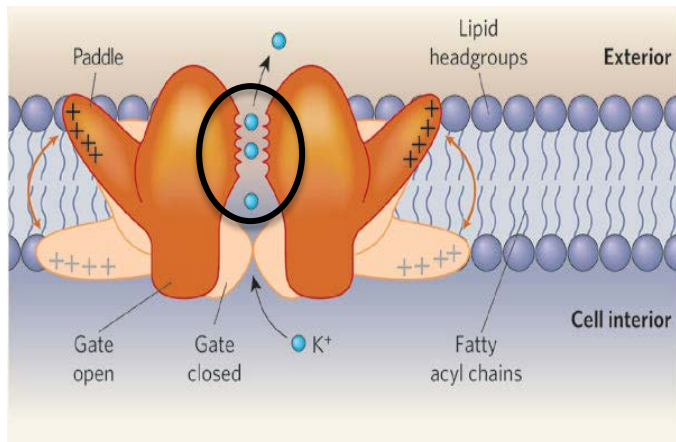


MLAutoTuningHPC: Learning Configurations

MLAutotunedHPC. Machine Learning for Parameter Auto-tuning in Molecular Dynamics Simulations: Efficient Dynamics of Ions near Polarizable Nanoparticles (NPs)

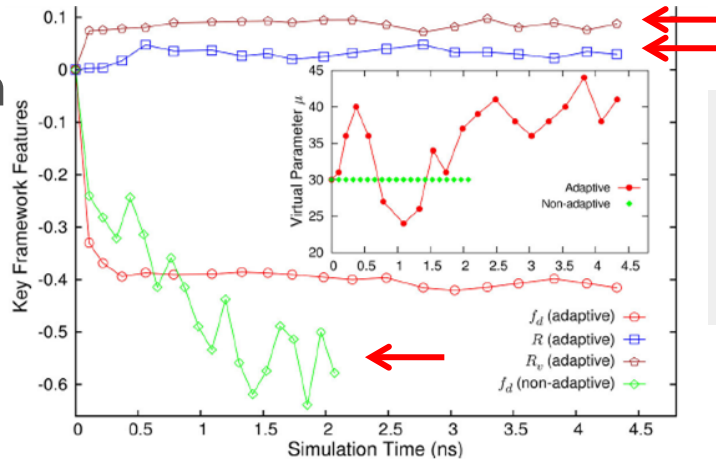
- Integration of machine learning (ML) methods for parameter prediction for MD simulations by demonstrating how they were realized in MD simulations of ions near **polarizable NPs**.
- Note ML used at start and end of simulation blocks

JCS Kadupitiya,
Geoffrey Fox,
Vikram Jadhao

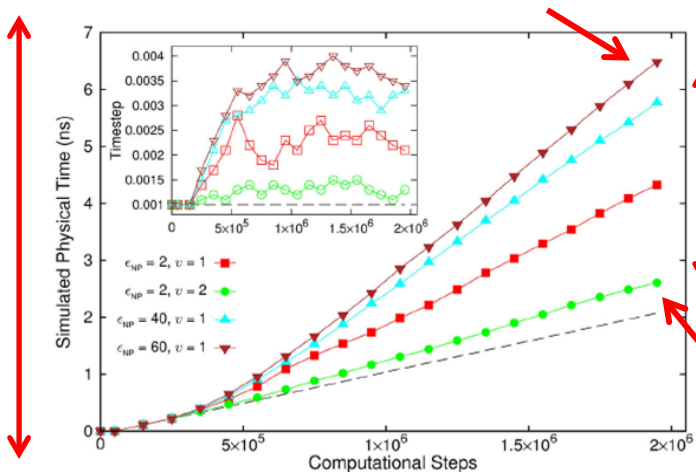


Results for Nanosimulation MLAutotuning

- Auto-tuning of parameters generated accurate dynamics of ions for 10 million steps while improving the stability.
- Integrated with ML-enhanced framework with hybrid OpenMP/MPI
- Maximum speedup of 3 from MLAutoTuning and a maximum speedup of 600 from the combination of ML and parallel computing.



Key characteristics of simulated system showing greater stability for ML enabled adaptive approach.

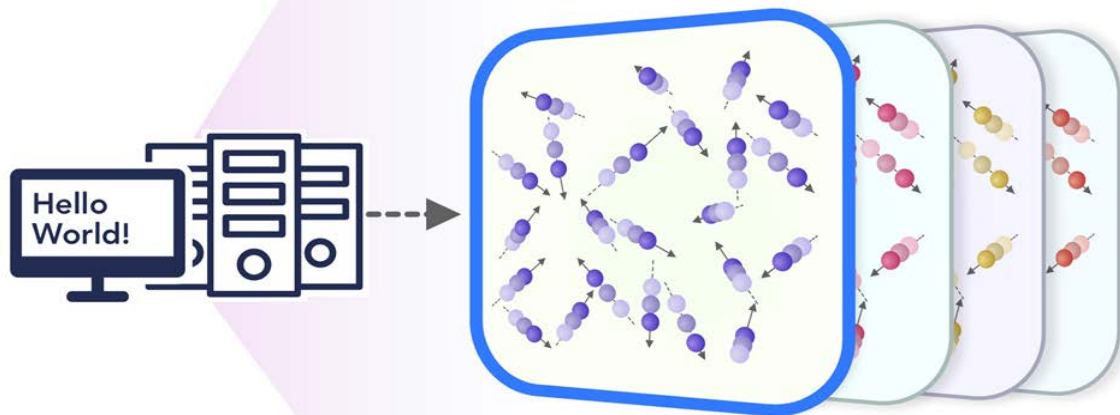


Quality of simulation measured by time simulated per step with increasing use of ML enhancements. (Larger is better).

Inset is timestep used

MLAutoTuningHPC: Smart Ensembles

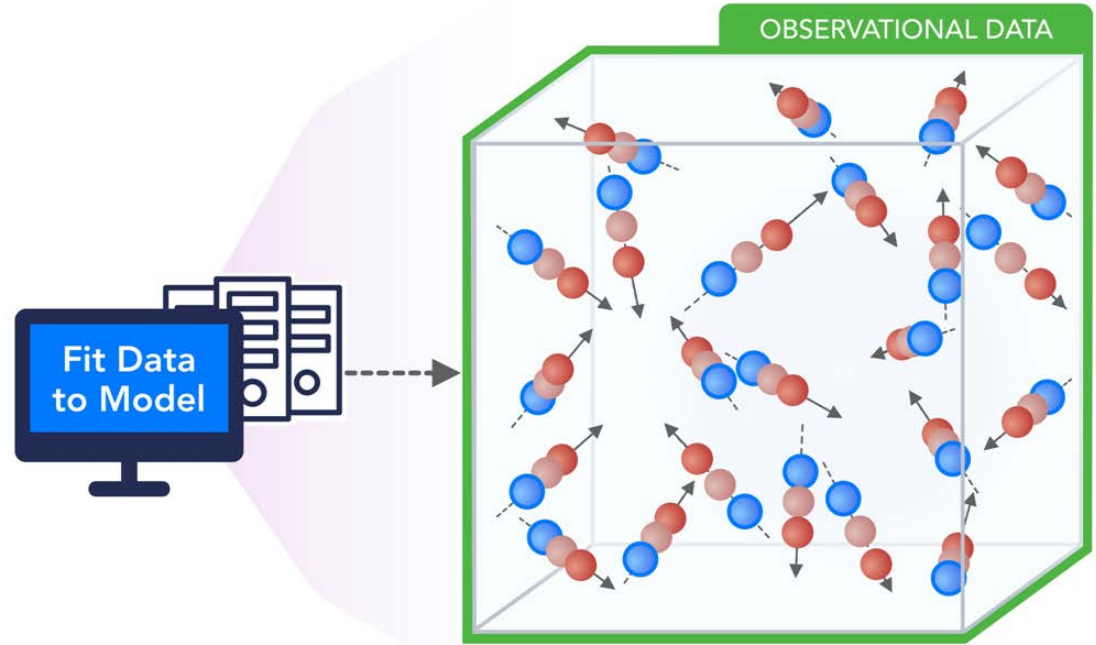
- Here we choose the best set of parameters to achieve some computation goal
- Such as providing the most efficient training set with defining parameters spread well over the relevant phase space.



Smart Ensembles

MLAutoTuningHPC: Learning Model Setups from Observational Data

- Seen when simulation set up as a set of agents.
- Tuning agent (model) parameters to optimize agent outputs to available empirical data presents one of the greatest challenges in model construction.



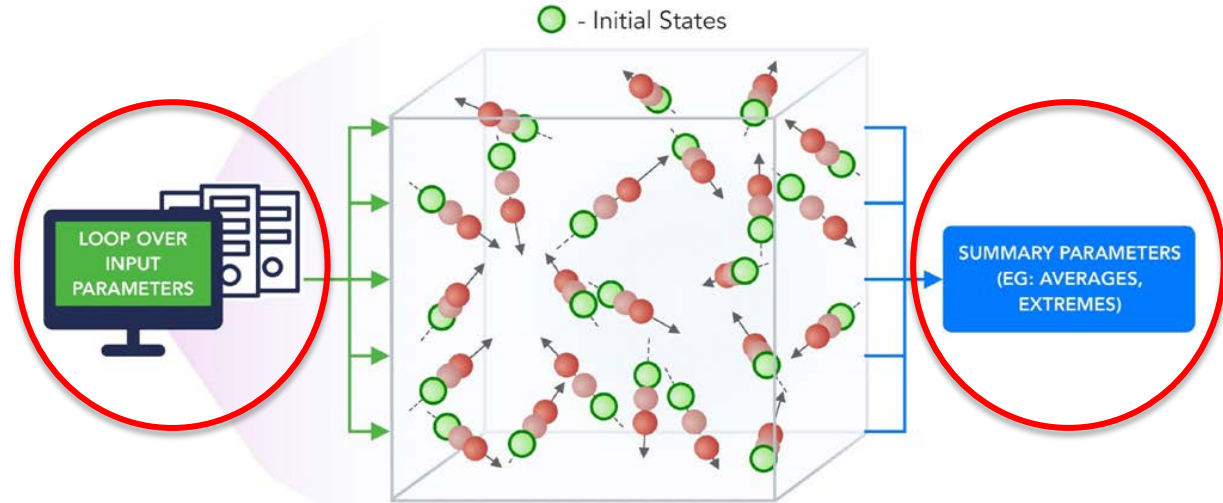
Learning Model Setups from Observational Data

MLforHPC Simulation Surrogates

MLaroundHPC: Learning Outputs from Inputs:

a) Computation Results from Computation defining Parameters

- Here one just feeds in a modest number of meta-parameters that the define the problem and learn a modest number of calculated answers.
- This presumably requires fewer training samples than “fields from fields” and is main use so far

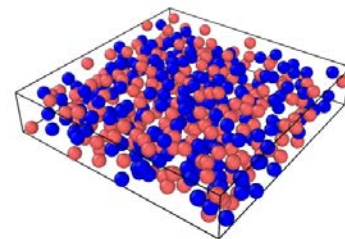
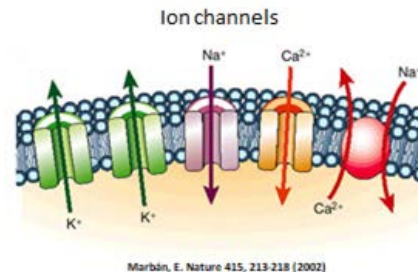
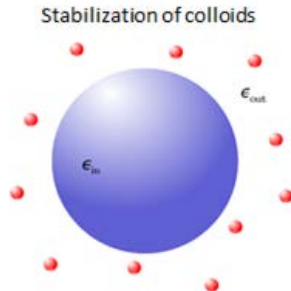
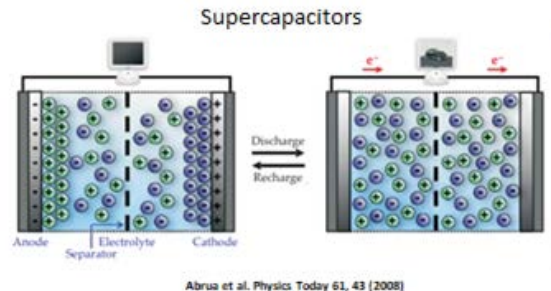


Learning Outputs from Inputs: Computation Results from Computation defining Parameters

Operationally same as **SimulationTrainedML** but with a different goal: In **SimulationTrainedML** the simulations are performed to directly train an AI system rather than the AI system being added to learn a simulation.

MLaroundHPC: ML for High Performance Surrogates of nanosimulations

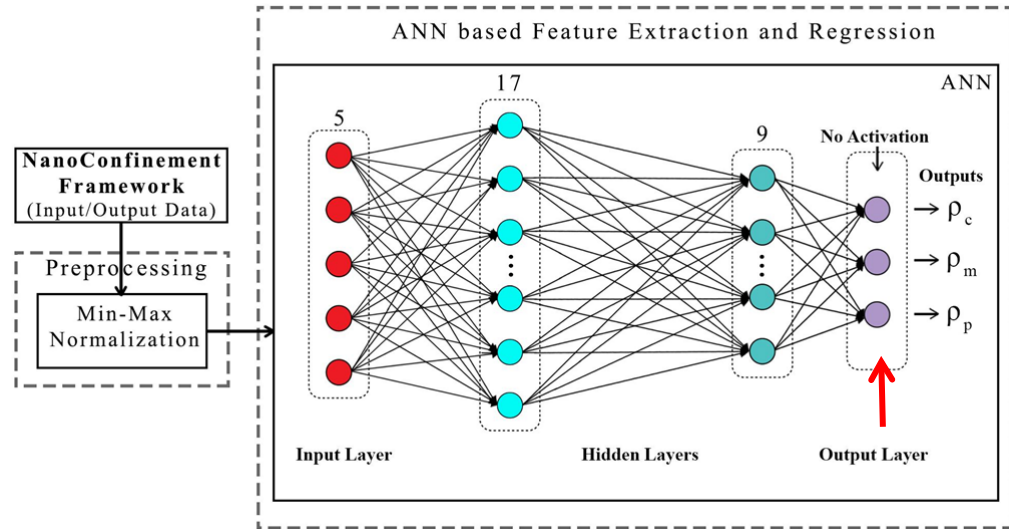
- An example of Learning Outputs from Inputs: Computation Results from Computation defining Parameters
- Employed to extract the ionic structure in electrolyte solutions confined by planar and spherical surfaces.
- Written with C++ and accelerated with hybrid MPI-OpenMP.
- MLaroundHPC successfully learns desired features associated with the output ionic density that are in excellent agreement with the results from explicit molecular dynamics simulations.
- Will be deployed on nanoHUB for education (an attractive use of surrogates)



ANN for Regression


- ANN was trained to predict three continuous variables; Contact density ρ_c , mid-point (center of the slit) density ρ_m , and peak density ρ_p
- TensorFlow, Keras and Sklearn libraries were used in the implementation
- Adam optimizer, xavier normal distribution, mean square loss function, dropout regularization.

- Dataset having 6,864 simulation configurations was created for training and testing (0.7:0.3) the ML model.
- Note learning network quite small



Parameter Prediction in Nanosimulation

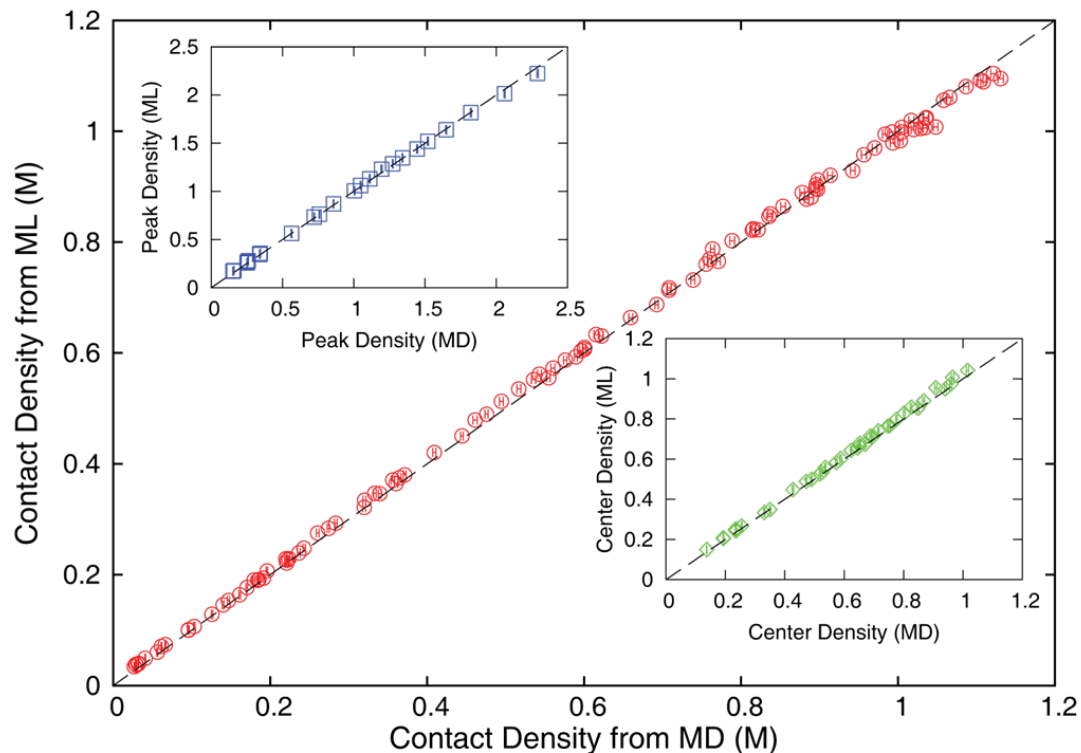
- **ANN based regression** model predicted Contact density ρ_c , mid-point (center of the slit) density ρ_m , and peak density ρ_p accurately with a success rate of 95:52% (MSE \sim 0:0000718), 92:07% (MSE \sim 0:0002293), and 94:78% (MSE \sim 0:0002306) respectively, easily outperforming other non-linear regression models
- Success means within error bars (2 sigma) of Molecular Dynamics Simulations



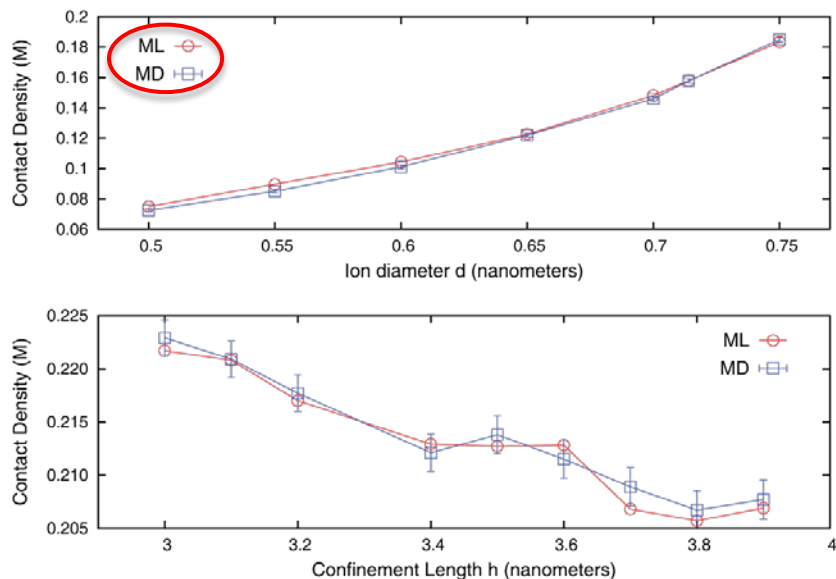
Model	Contact Density		Midpoint Density		Peak Density	
	<i>Success %</i>	<i>MSE</i>	<i>Success %</i>	<i>MSE</i>	<i>Success %</i>	<i>MSE</i>
Polynomial	61.04	0.0129300	60.84	0.0187700	61.87	0.0100400
Kernel-Ridge	78.86	0.0030900	76.57	0.0041200	75.93	0.0049800
Support Vector	80.11	0.0012700	79.55	0.0024900	81.98	0.0010600
Decision Tree	68.44	0.0084600	64.54	0.0094900	62.47	0.0110700
Random Forest	74.15	0.0045700	70.85	0.0078900	75.09	0.0040800
ANN based	<u>95.52</u>	<u>0.0000718</u>	92.07	<u>0.0002293</u>	94.78	0.0002306

Accuracy comparison between ML predictions and MD simulation results

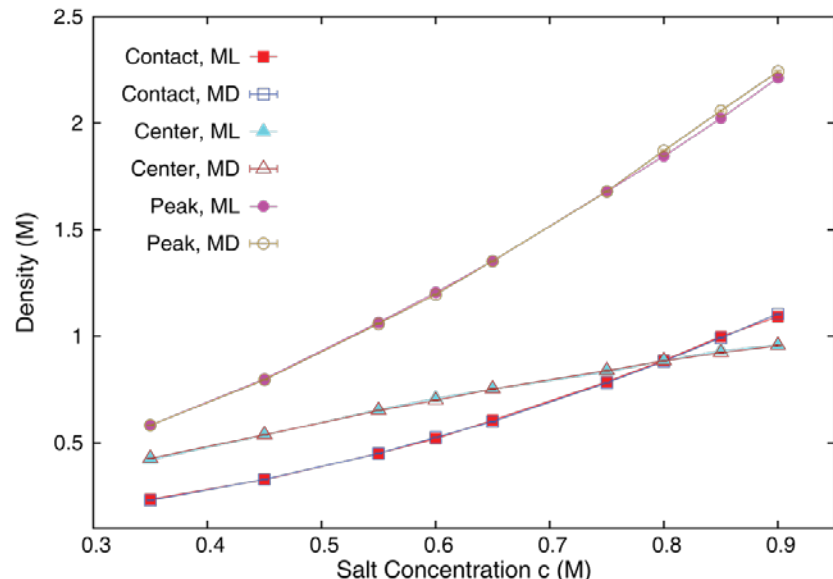
ρ_c , ρ_m and ρ_p predicted by the ML model were found to be in excellent agreement with those calculated using the MD method; data from either approach fall on the dashed lines which indicate perfect correlation.



Rapid access to trendlines using ML Surrogates



(Top) Trendlines for contact density vs. ion diameter
(Bottom) Trendlines for contact density vs. confinement length



Contact, peak, and center-of-the-slit (mid-point) density vs. salt concentration.

ML predictions are within the error bars generated via MD simulations (1%).

Speedup of ML around HPC

- T_{seq} is sequential time
- T_{train} time for a (parallel) simulation used in training ML
- T_{learn} is time per point to run machine learning
- T_{lookup} is time to run inference per instance
- N_{train} number of training samples
- N_{lookup} number of results looked up

N_{train} is 7K to 16K in our work

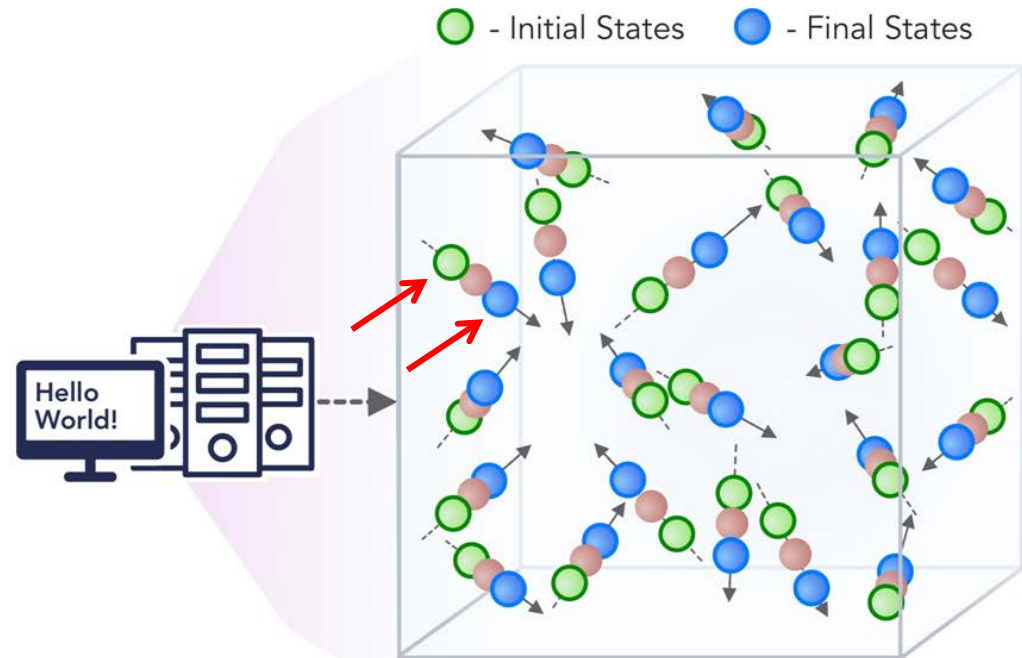
$$\text{Effective Speedup } S = \frac{T_{seq}(N_{lookup} + N_{train})}{T_{lookup}N_{lookup} + (T_{train} + T_{learn})N_{train}}$$

- Becomes T_{seq}/T_{train} if ML not used
- Becomes T_{seq}/T_{lookup} (**10⁵ faster in our case**) if inference dominates (will overcome end of Moore's law and win the race to **zettascale**)
- Another factor as inferences uses one core; parallel simulation 128 cores
- **Strong scaling as no need to parallelize more than effective number of nodes**

MLforHPC Simulation Surrogates

MLaroundHPC: b) Learning Outputs from Inputs: Fields from Fields

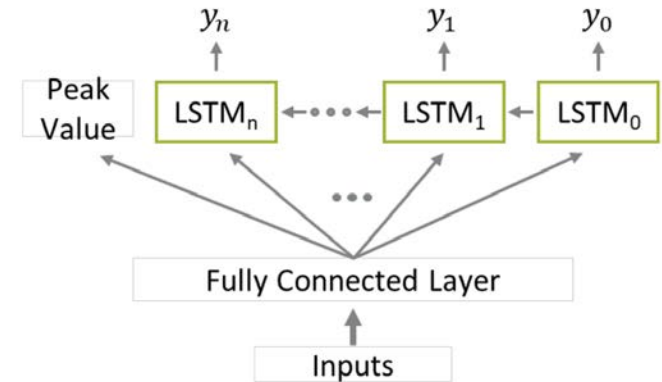
- Here one feeds in initial conditions and the neural network learns the result where initial and final results are fields
- There is also **c) Learning Outputs from Inputs: output fields from Computation defining Parameters** combining *a)* and *b)*



Learning Outputs from Inputs: Fields from Fields

How does one do this for case c) ?

- If you are learning particular output features (as in Computation Results from Computation defining Parameters), then a **simple (not necessarily deep) neural net suffices**
- If you want to learn the output fields (from either input fields or input Computation defining Parameters), then a more sophisticated approach is appropriate
- Recent paper “Massive computational acceleration by using neural networks to emulate mechanism based biological models” uses 501 LSTM units to represent a ~~one-dimensional~~ grid of values which is output of a two-dimensional gene circuit simulation which only depends on radius.
- Note **LSTM models sequences** and one gets
- Sequences in either time (usual LSTM application) or space
- **The ML representation allowed a much richer parameter sweep showing features not in training set**
- **Performance improved by factor ~~30,000~~**



Massive computational acceleration by using neural networks to emulate mechanism based biological models

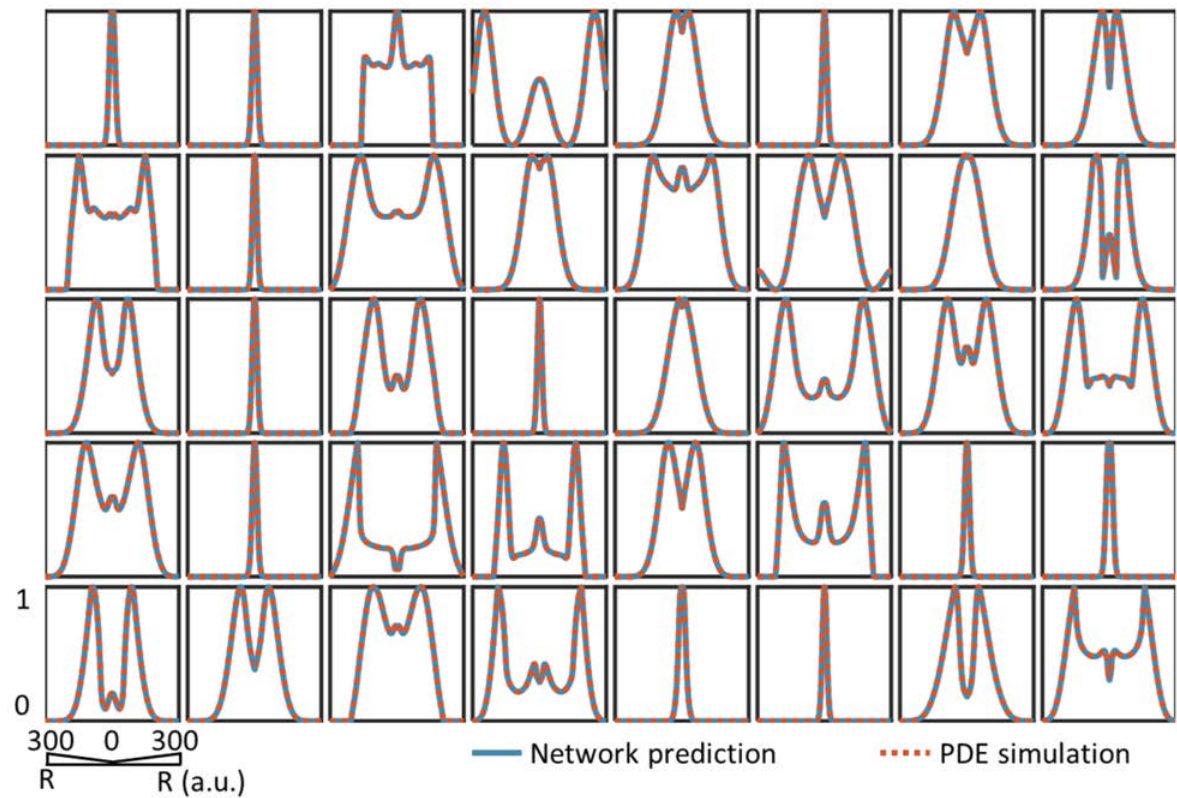


Figure S3. Comparison between predicted distributions generated by neural network and distributions generated by mechanism-based model. These examples are randomly selected from the training dataset.