FLATIRON INSTITUTE
Center for Computational Mathematics

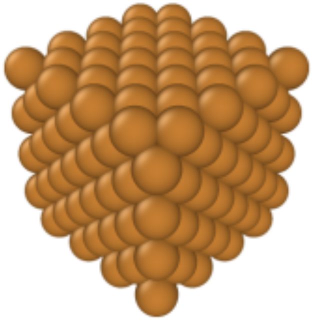
Symmetry-Preserving Neural Networks

Jiequn Han
Flatiron Institute, CCM

FWAM
Oct 15, 2021

Example Problem 1: Potential Energy

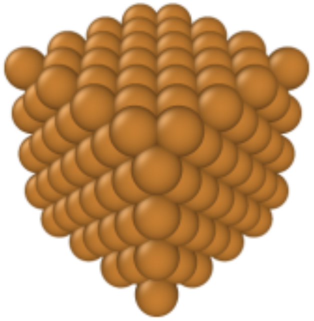
Given coordinates of a group of atoms $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, find its potential energy $E = E(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$



FLATIRON INSTITUTE
Center for Computational Mathematics

Example Problem 1: Potential Energy

Given coordinates of a group of atoms $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, find its potential energy $E = E(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$

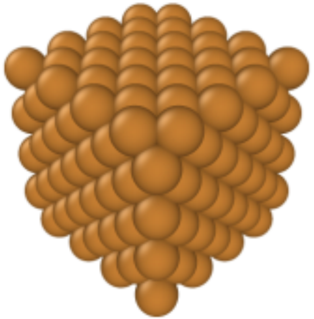


Physical Symmetries

- Translation $E = E(\boldsymbol{x}_1 + \Delta\boldsymbol{x}, \boldsymbol{x}_2 + \Delta\boldsymbol{x}, \cdots, \boldsymbol{x}_n + \Delta\boldsymbol{x})$

Example Problem 1: Potential Energy

Given coordinates of a group of atoms $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, find its potential energy $E = E(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$



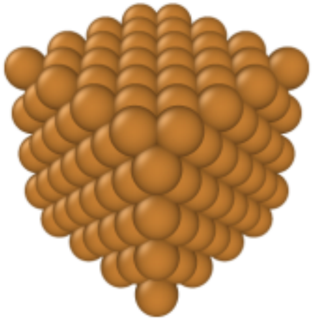
Physical Symmetries

- Translation $\quad E = E(\boldsymbol{x}_1 + \Delta\boldsymbol{x}, \boldsymbol{x}_2 + \Delta\boldsymbol{x}, \cdots, \boldsymbol{x}_n + \Delta\boldsymbol{x})$
- Rotation $\quad E = E(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$

FLATIRON INSTITUTE
Center for Computational Mathematics

Example Problem 1: Potential Energy

Given coordinates of a group of atoms $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, find its potential energy $E = E(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$



Physical Symmetries

- Translation $E = E(\boldsymbol{x}_1 + \Delta\boldsymbol{x}, \boldsymbol{x}_2 + \Delta\boldsymbol{x}, \cdots, \boldsymbol{x}_n + \Delta\boldsymbol{x})$
- Rotation $E = E(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$
- Permutation $E = E(\boldsymbol{x}_{\sigma(1)}, \boldsymbol{x}_{\sigma(2)}, \cdots, \boldsymbol{x}_{\sigma(n)})$

Example Problem 2: Transport Equation

$$\Delta c(\boldsymbol{x}) - \nabla \cdot (\boldsymbol{u}(\boldsymbol{x})c(\boldsymbol{x})) + S(c(\boldsymbol{x})) = 0$$

Example Problem 2: Transport Equation

$$\Delta c(\boldsymbol{x}) - \nabla \cdot (\boldsymbol{u}(\boldsymbol{x})c(\boldsymbol{x})) + S(c(\boldsymbol{x})) = 0$$

Spatial discretization: $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \boldsymbol{u}_1 = \boldsymbol{u}(\boldsymbol{x}_1), \boldsymbol{u}_2 = \boldsymbol{u}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}_n = \boldsymbol{u}(\boldsymbol{x}_n),$

Consider the quantity $I := \int_\Omega c(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx I(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n)$

FLATIRON INSTITUTE
Center for Computational Mathematics

Example Problem 2: Transport Equation

$$\Delta c(\boldsymbol{x}) - \nabla \cdot (\boldsymbol{u}(\boldsymbol{x})c(\boldsymbol{x})) + S(c(\boldsymbol{x})) = 0$$

Spatial discretization: $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \boldsymbol{u}_1 = \boldsymbol{u}(\boldsymbol{x}_1), \boldsymbol{u}_2 = \boldsymbol{u}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}_n = \boldsymbol{u}(\boldsymbol{x}_n),$

Consider the quantity $I := \int_\Omega c(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx I(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n)$

Physical Symmetries

- Translation $I = I(\boldsymbol{x}_1 + \Delta\boldsymbol{x}, \boldsymbol{x}_2 + \Delta\boldsymbol{x}, \cdots, \boldsymbol{x}_n + \Delta\boldsymbol{x}, \boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n)$
- Rotation $I = I(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n, R\boldsymbol{u}_1, R\boldsymbol{u}_2, \cdots, R\boldsymbol{u}_n)$
- Permutation $I = I(\boldsymbol{x}_{\sigma(1)}, \boldsymbol{x}_{\sigma(2)}, \cdots, \boldsymbol{x}_{\sigma(n)}, \boldsymbol{u}_{\sigma(1)}, \boldsymbol{u}_{\sigma(2)}, \cdots, \boldsymbol{u}_{\sigma(n)})$

FLATIRON INSTITUTE
Center for Computational Mathematics

Problem Setup

A function f maps a set of coordinates to a scalar output

$$y = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$$

FLATIRON INSTITUTE
Center for Computational Mathematics

Problem Setup

A function f maps a set of coordinates to a scalar output

$$y = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$$

Supervised learning: fit this function from data and preserve all the symmetries simultaneously

FLATIRON INSTITUTE
Center for Computational Mathematics

Problem Setup

A function f maps a set of coordinates to a scalar output

$$y = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$$

Supervised learning: fit this function from data and preserve all the symmetries simultaneously

Why symmetry-preserving?

- respect the physics
- better data efficiency
- better accuracy

Related Work

- Handcrafted features, kernel method: Gaussian Approximation Potentials (GAP), Smooth Overlap of Atomic Positions (SOAP), etc.

Related Work

- Handcrafted features, kernel method: Gaussian Approximation Potentials (GAP), Smooth Overlap of Atomic Positions (SOAP), etc.
- Behler-Parrinello neural network (BPNN)

Related Work

- Handcrafted features, kernel method: Gaussian Approximation Potentials (GAP), Smooth Overlap of Atomic Positions (SOAP), etc.
- Behler-Parrinello neural network (BPNN)
- Learned features: Deep Potential/Vector Cloud Neural Network, SchNet, etc.

Related Work

- Handcrafted features, kernel method: Gaussian Approximation Potentials (GAP), Smooth Overlap of Atomic Positions (SOAP), etc.
- Behler-Parrinello neural network (BPNN)
- Learned features: Deep Potential/Vector Cloud Neural Network, SchNet, etc.
- Group representation: Group Equivariant Convolutional Networks, Steerable Convolutional Neural Networks, Clebsch–Gordan Nets, etc.

FLATIRON INSTITUTE
Center for Computational Mathematics

Translation and Rotation Symmetry

Translation: always use relative coordinates

$$(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \mapsto \quad (\boldsymbol{x}_1', \boldsymbol{x}_2', \cdots, \boldsymbol{x}_n') = (\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \boldsymbol{x}_2 - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x}_n - \bar{\boldsymbol{x}})$$

Translation and Rotation Symmetry

Translation: always use relative coordinates

$$(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \mapsto \quad (\boldsymbol{x}_1', \boldsymbol{x}_2', \cdots, \boldsymbol{x}_n') = (\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \boldsymbol{x}_2 - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x}_n - \bar{\boldsymbol{x}})$$

Rotation:

$$D_{ij}' = \boldsymbol{x}_i' \cdot \boldsymbol{x}_j' \quad \text{or} \quad D' = X^\top X \ \text{ with } \ X = [\boldsymbol{x}_1', \boldsymbol{x}_2', \cdots, \boldsymbol{x}_n']^\top$$

Translation and Rotation Symmetry

Translation: always use relative coordinates

$$(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \mapsto \quad (\boldsymbol{x}'_1, \boldsymbol{x}'_2, \cdots, \boldsymbol{x}'_n) = (\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \boldsymbol{x}_2 - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x}_n - \bar{\boldsymbol{x}})$$

Rotation:

$$D'_{ij} = \boldsymbol{x}'_i \cdot \boldsymbol{x}'_j \quad \text{or} \quad D' = X^\top X \ \text{ with } \ X = [\boldsymbol{x}'_1, \boldsymbol{x}'_2, \cdots, \boldsymbol{x}'_n]^\top$$

However, it lacks permutational invariance

FLATIRON INSTITUTE
Center for Computational Mathematics

Permutation Symmetry

Deep Sets: a function f operating on a set $\{\boldsymbol{x}_i\}_{i=1}^n$ can be represented by

$$\rho(\sum_{i=1}^{n} \phi(\boldsymbol{x}_i))$$

FLATIRON INSTITUTE
Center for Computational Mathematics

Permutation Symmetry

Deep Sets: a function f operating on a set $\{\boldsymbol{x}_i\}_{i=1}^n$ can be represented by

$$\rho\left(\sum_{i=1}^{n} \phi(\boldsymbol{x}_i)\right)$$

Example: $$z_1^2 + z_2^2 + z_1 z_2 = \frac{1}{2}(z_1 + z_2)^2 + \frac{1}{2}(z_1^2 + z_2^2)$$

Let $\phi(z) = [z, z^2]^\top$ and $\rho([a, b]^\top) = a^2/2 + b/2$

FLATIRON INSTITUTE
Center for Computational Mathematics

Permutation Symmetry

Deep Sets: a function f operating on a set $\{\boldsymbol{x}_i\}_{i=1}^n$ can be represented by

$$\rho(\sum_{i=1}^{n} \phi(\boldsymbol{x}_i))$$

Example: $$z_1^2 + z_2^2 + z_1 z_2 = \frac{1}{2}(z_1 + z_2)^2 + \frac{1}{2}(z_1^2 + z_2^2)$$

Let $\phi(z) = [z, z^2]^\top$ and $\rho([a, b]^\top) = a^2/2 + b/2$

Ansatz: parameterize ϕ, ρ with neural networks

FLATIRON INSTITUTE
Center for Computational Mathematics

All Symmetries Simultaneously

Introduce a set of m embedding functions $\{\phi_k(\cdot)\}_{k=1}^{m}$

$$L_{kj} = \frac{1}{n}\sum_{i=1}^{n} \phi_k(|\boldsymbol{x}_i'|)\boldsymbol{x}_{ij}', \quad k = 1,\cdots,m,\ j = 1,2,3$$

or $\quad L = \dfrac{1}{n}G^\top X$ with $G_{ki} = \phi_k(|\boldsymbol{x}_i'|)$

FLATIRON INSTITUTE
Center for Computational Mathematics

All Symmetries Simultaneously

Introduce a set of m embedding functions $\{\phi_k(\cdot)\}_{k=1}^{m}$

$$L_{kj} = \frac{1}{n}\sum_{i=1}^{n}\phi_k(|\boldsymbol{x}_i'|)\boldsymbol{x}_{ij}', \quad k = 1,\cdots,m,\ j = 1,2,3$$

or $\quad L = \frac{1}{n}G^\top X$ with $G_{ki} = \phi_k(|\boldsymbol{x}_i'|)$

This leads to symmetry-preserving feature matrix $D = LL^\top = \frac{1}{n^2}G^\top XX^\top G$

All Symmetries Simultaneously

Introduce a set of m embedding functions $\{\phi_k(\cdot)\}_{k=1}^{m}$

$$L_{kj} = \frac{1}{n}\sum_{i=1}^{n} \phi_k(|\boldsymbol{x}_i'|)\boldsymbol{x}_{ij}', \quad k = 1,\cdots,m, \ j = 1,2,3$$

or $\quad L = \frac{1}{n}G^\top X \;$ with $\; G_{ki} = \phi_k(|\boldsymbol{x}_i'|)$

This leads to symmetry-preserving feature matrix $D = LL^\top = \frac{1}{n^2}G^\top X X^\top G$

Map to the final output through a general function $\rho(\text{vec}(D))$

FLATIRON INSTITUTE
Center for Computational Mathematics

All Symmetries Simultaneously

Introduce a set of m embedding functions $\{\phi_k(\,\cdot\,)\}_{k=1}^{m}$

$$L_{kj} = \frac{1}{n}\sum_{i=1}^{n} \phi_k(|\boldsymbol{x}_i'|)\boldsymbol{x}_{ij}', \quad k = 1,\cdots,m,\ j = 1,2,3$$

or $\quad L = \frac{1}{n}G^\top X$ with $G_{ki} = \phi_k(|\boldsymbol{x}_i'|)$

This leads to symmetry-preserving feature matrix $D = LL^\top = \frac{1}{n^2}G^\top XX^\top G$

Map to the final output through a general function $\rho(\mathrm{vec}(D))$

Ansatz: parameterize ϕ, ρ with neural networks

FLATIRON INSTITUTE
Center for Computational Mathematics

Extensions

- Guarantee equivariance if the output is

$$\text{vector: } \boldsymbol{r} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad R\boldsymbol{r} = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

$$\text{or tensorr: } Q = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad RQR^\top = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

Extensions

- Guarantee equivariance if the output is

$$\text{vector: } \boldsymbol{r} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad R\boldsymbol{r} = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

$$\text{or tensorr: } Q = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad RQR^\top = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

- Guarantee invariance and equivariance if we have additional scalar/vector/tensor features attached to each point

FLATIRON INSTITUTE
Center for Computational Mathematics

Extensions

- Guarantee equivariance if the output is

$$\text{vector: } \boldsymbol{r} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad R\boldsymbol{r} = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

$$\text{or tensorr: } Q = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad RQR^\top = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

- Guarantee invariance and equivariance if we have additional scalar/vector/tensor features attached to each point

- Use high-order information to do embedding

FLATIRON INSTITUTE
Center for Computational Mathematics

Extensions

- Guarantee equivariance if the output is

$$\text{vector: } \boldsymbol{r} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad R\boldsymbol{r} = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

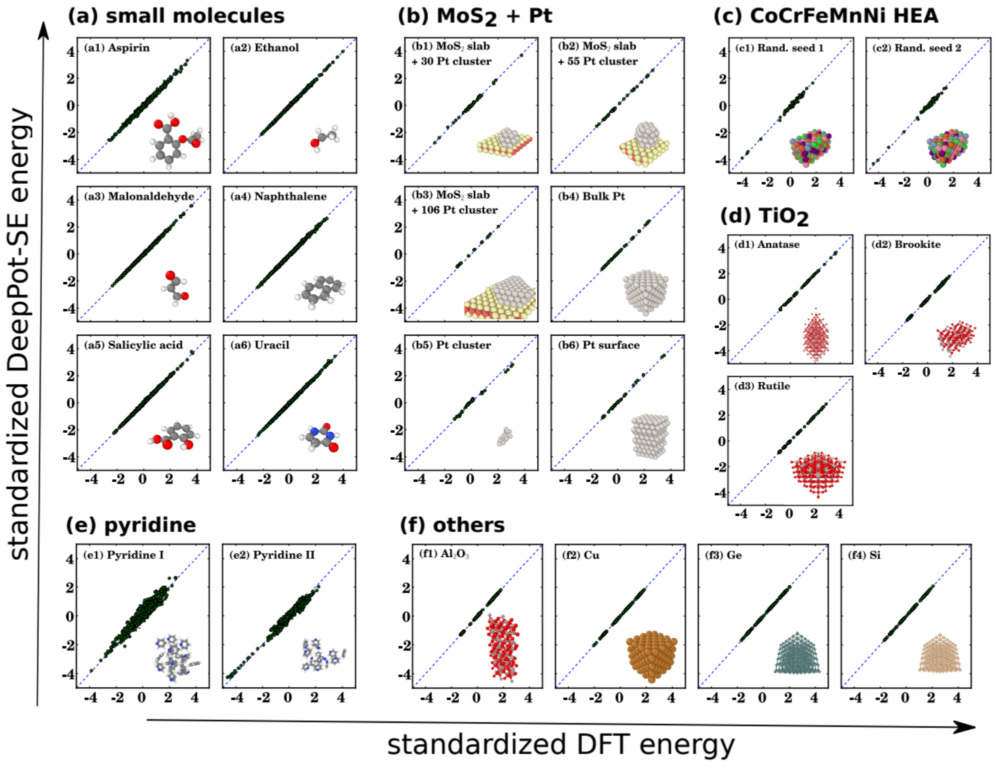
$$\text{or tensorr: } Q = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \quad \rightarrow \quad RQR^\top = f(R\boldsymbol{x}_1, R\boldsymbol{x}_2, \cdots, R\boldsymbol{x}_n)$$

- Guarantee invariance and equivariance if we have additional scalar/vector/tensor features attached to each point

- Use high-order information to do embedding

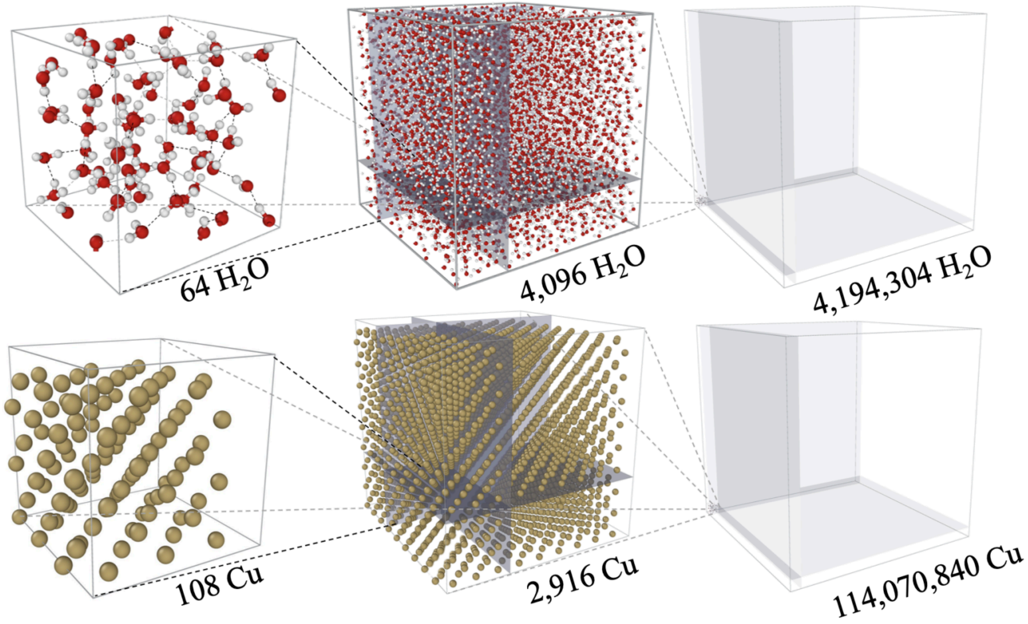
Open question: universal approximation property with practical optimality and scalability

Application: Molecular Dynamics



FLATIRON INSTITUTE
Center for Computational Mathematics

Application: Molecular Dynamics

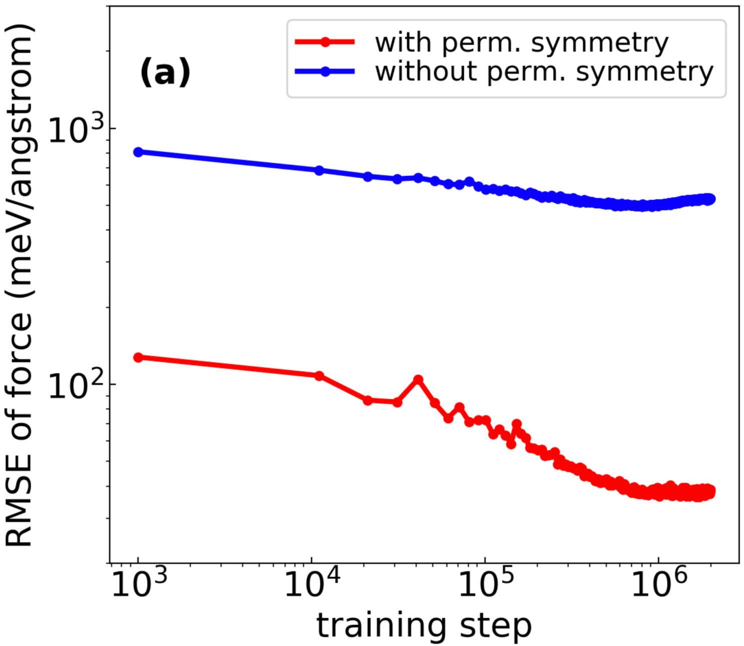


(a) AIMD + HPC; (b) DeePMD+1 GPU @ Home; (c) DeePMD+27360 GPUs @ Summit.

n is bounded as system size increased by short-range effect

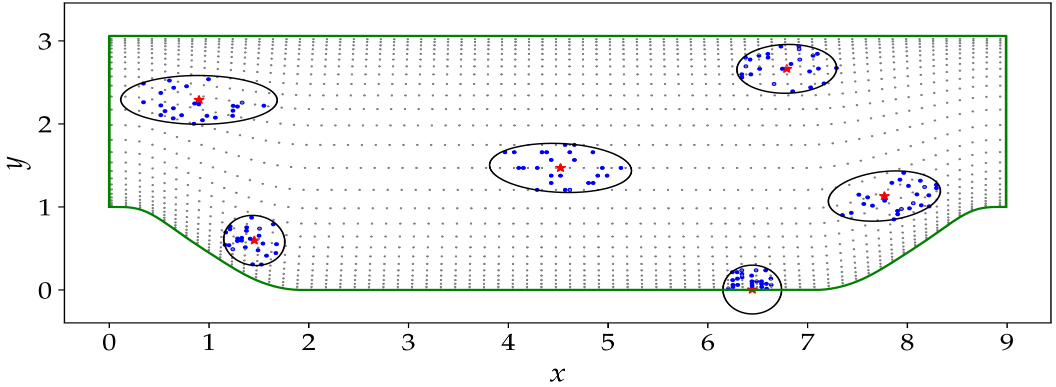
FLATIRON INSTITUTE
Center for Computational Mathematics

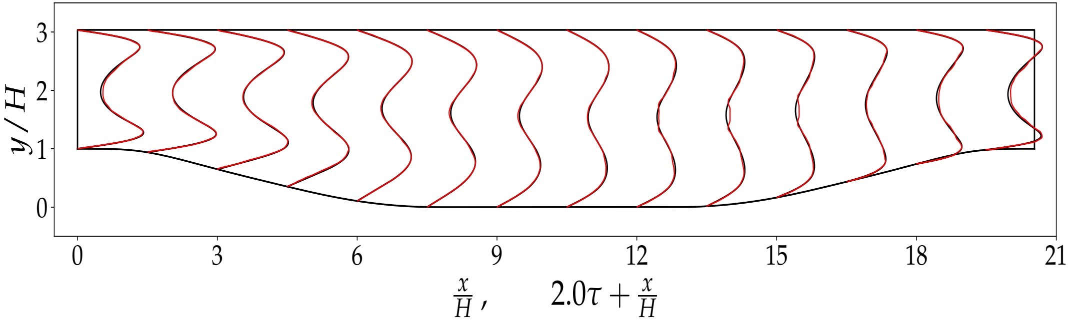
Importance of Symmetry



FLATIRON INSTITUTE
Center for Computational Mathematics

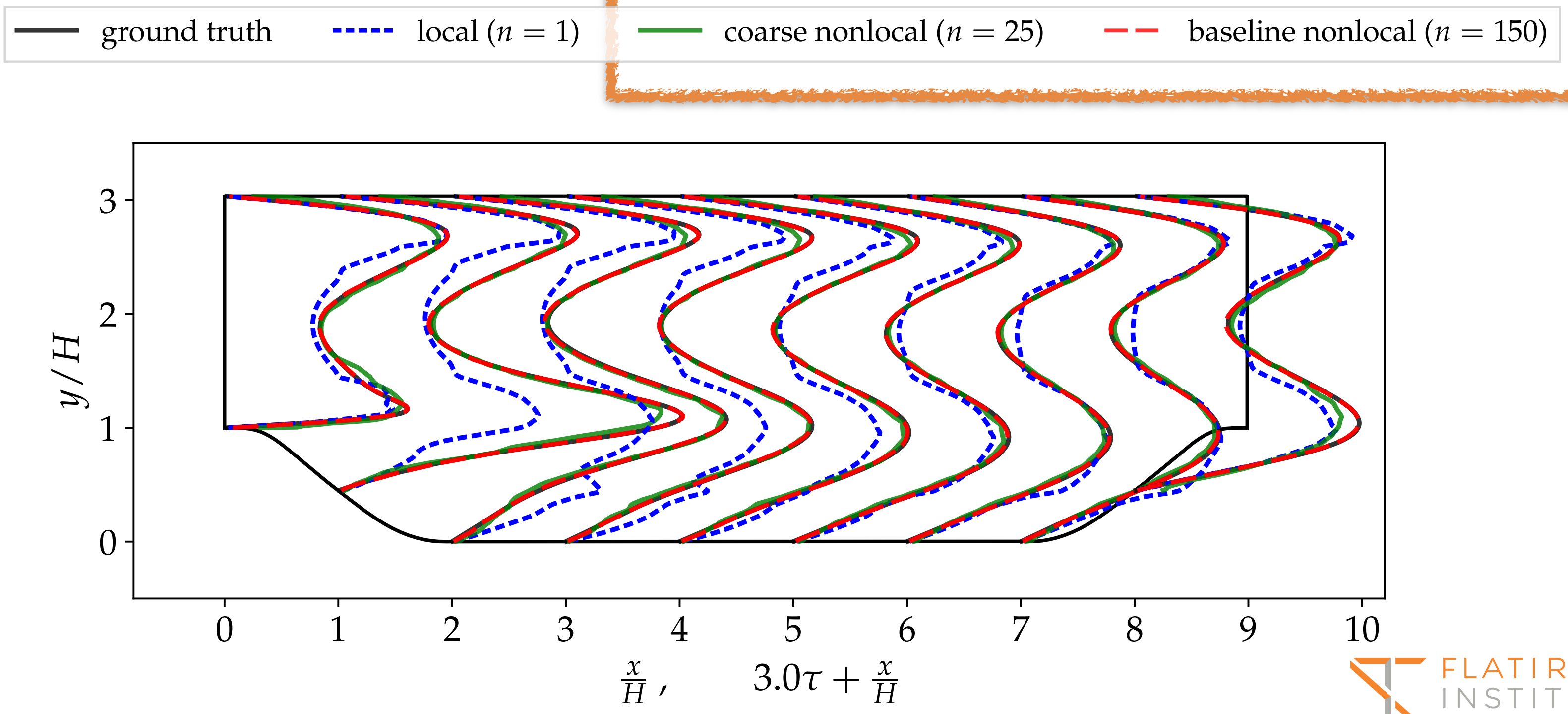
Application: Transport Equation





FLATIRON INSTITUTE
Center for Computational Mathematics

Adaptivity to Different Sizes



FLATIRON INSTITUTE
Center for Computational Mathematics

References

- L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, and W. E, *End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems*, NeurIPS, (2018).
  Code: https://github.com/deepmodeling/deepmd-kit
- W. Jia, H; Wang, M. Chen, D. Lu, J. Liu, L. Lin, R. Car, W. E, L. Zhang, *Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning*, 2020.
- X.-H. Zhou, J. Han, and H. Xiao, *Frame-independent vector-cloud neural network for nonlocal constitutive modelling on arbitrary grids*, arXiv:2103.06685, accepted by CMAME.
  Code: https://github.com/xuhuizhou-vt/VCNN_nonlocal-constitutive-model

Thank You

FLATIRON INSTITUTE
Center for Computational Mathematics