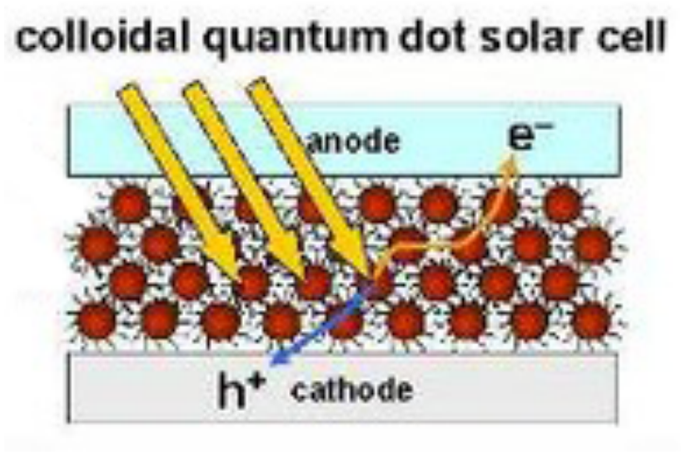
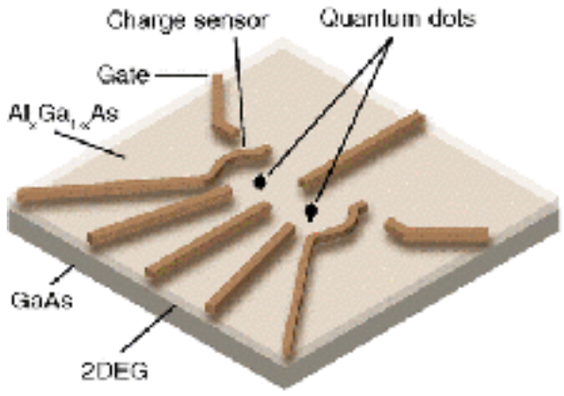
Speeding up computational geometry optimization using statistical methods and neural networks

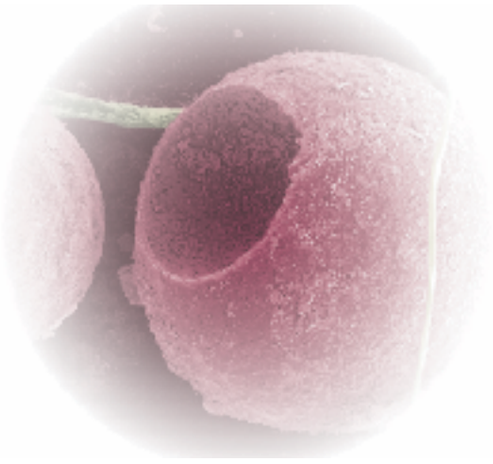
GM20171219 Presentation - Xiang Zhang

Motivation: Quantum Dots and Nanostructures

- Quantum dots are have optoelectronic and biological applications, and are fundamentally interesting.



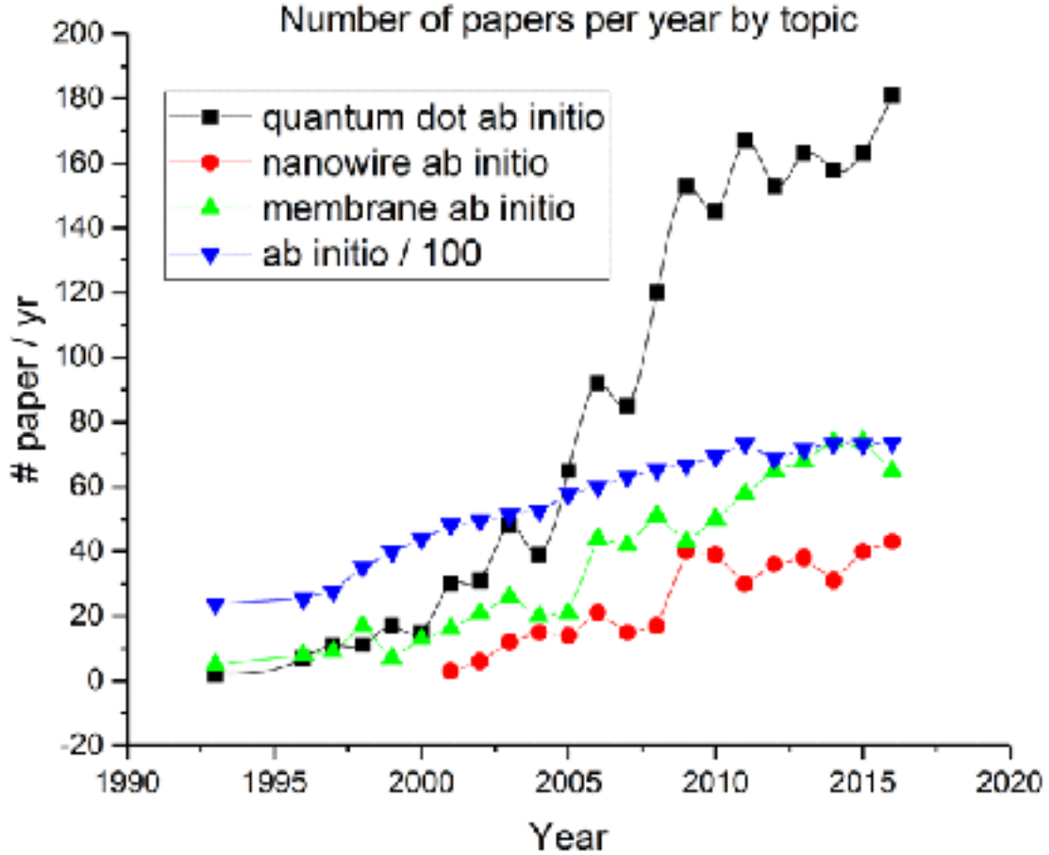




Motivation: Quantum Dots and Nanostructures

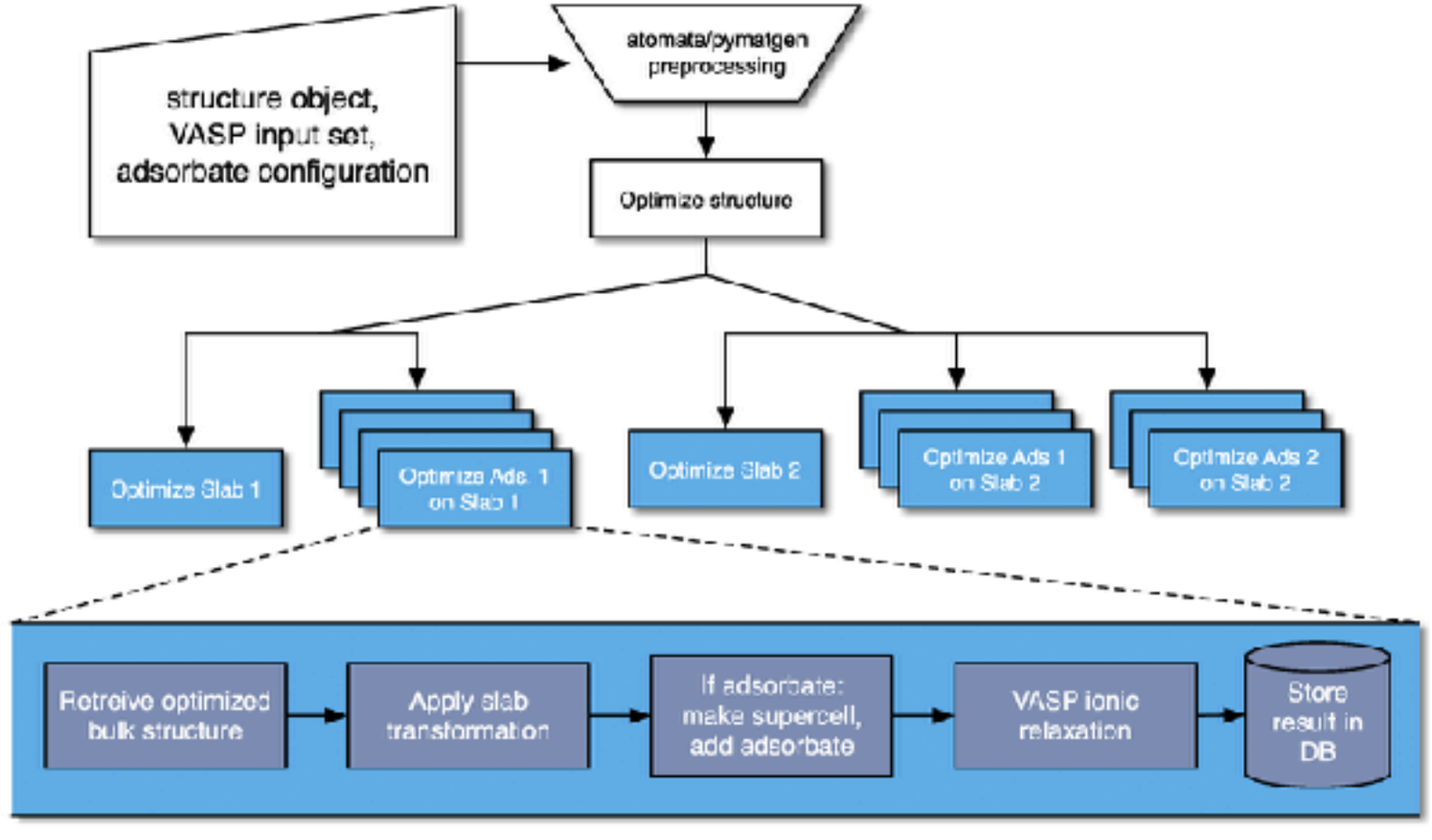
- Quantum dots are have optoelectronic and biological applications, and are fundamentally interesting.

- Computational research on non-crystalline structures are on the rise.



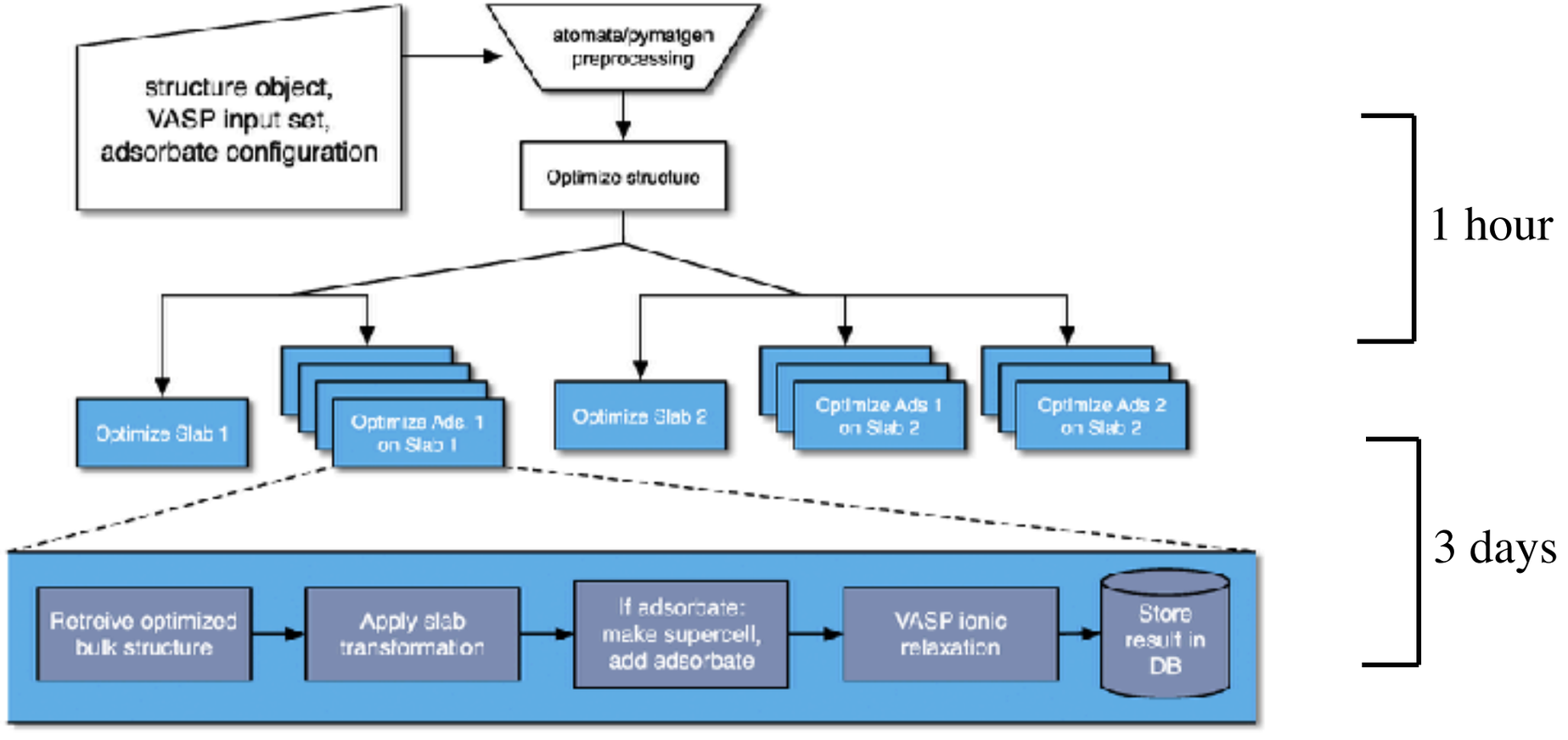
Motivation: Geometry Optimization is Slow

- Computational workflow: optimize → electronic structure



Motivation: Geometry Optimization is Slow

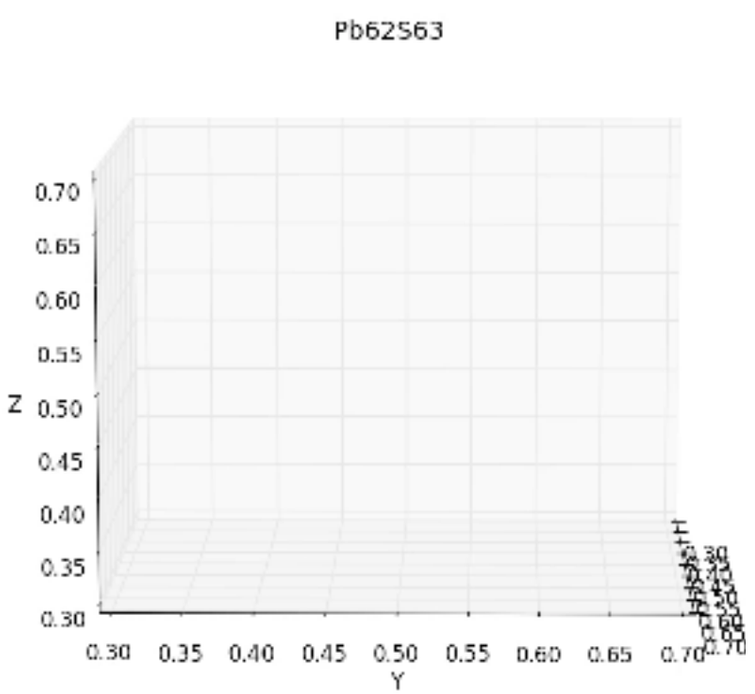
- Computational workflow: optimize → electronic structure
- Computing band structure takes 1 self-consistent step. Geometry optimization can take 200.

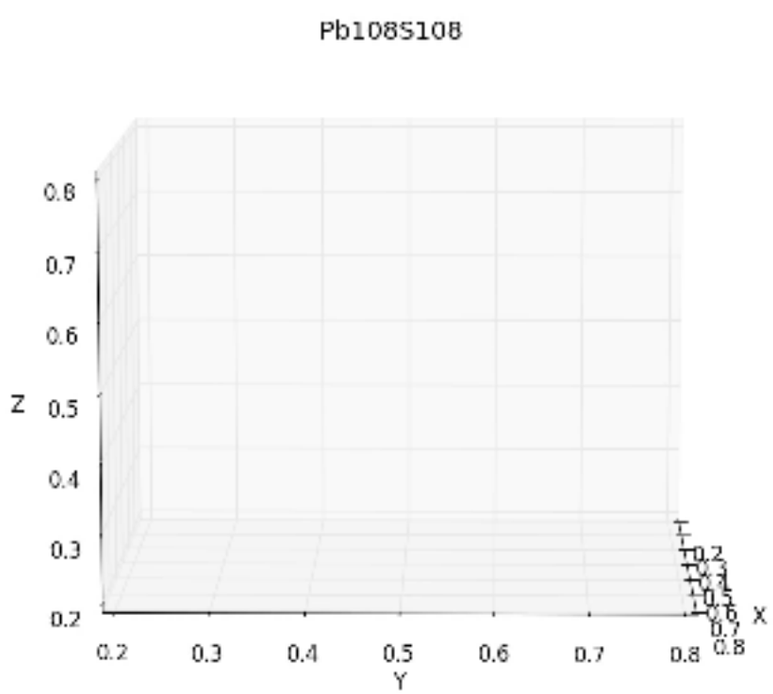


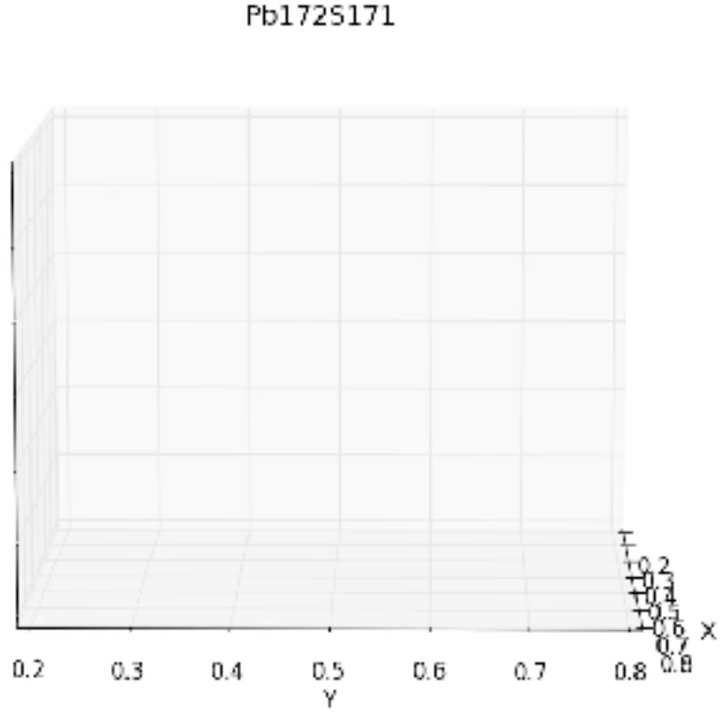
Motivation: Geometry Optimization is Learnable

In geometry optimization, atoms are incrementally moved towards the lowest energy configuration.

Here are some trajectories.

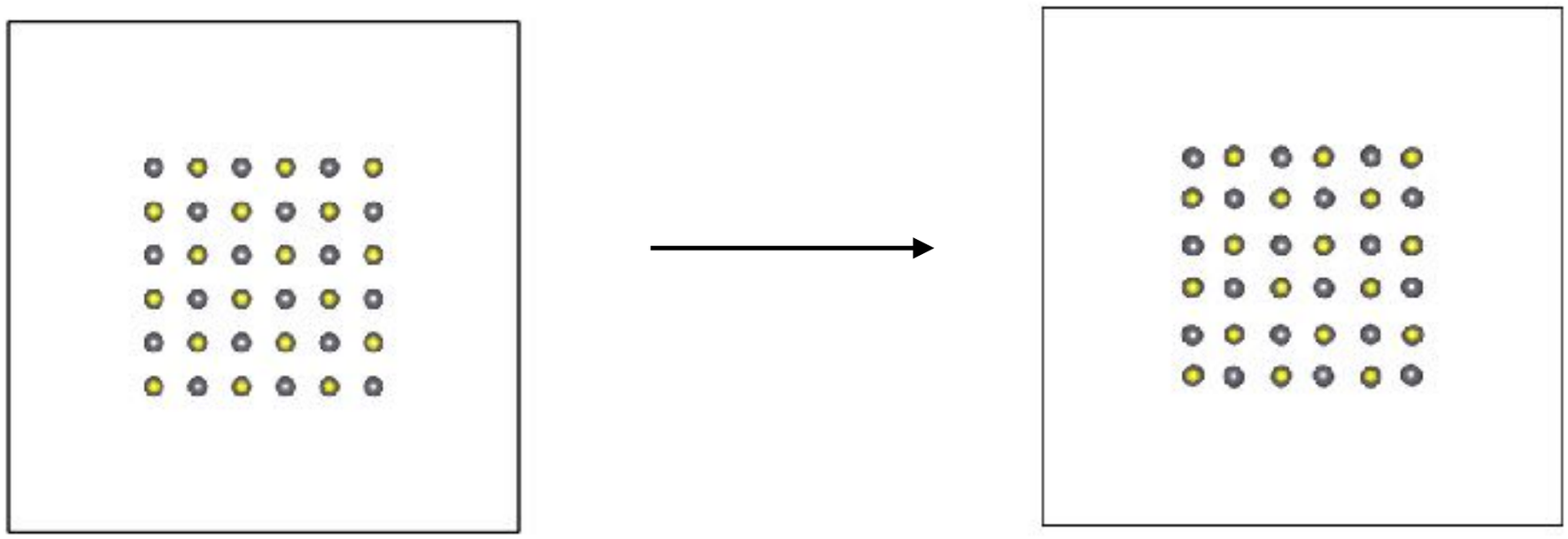






Problem Statement

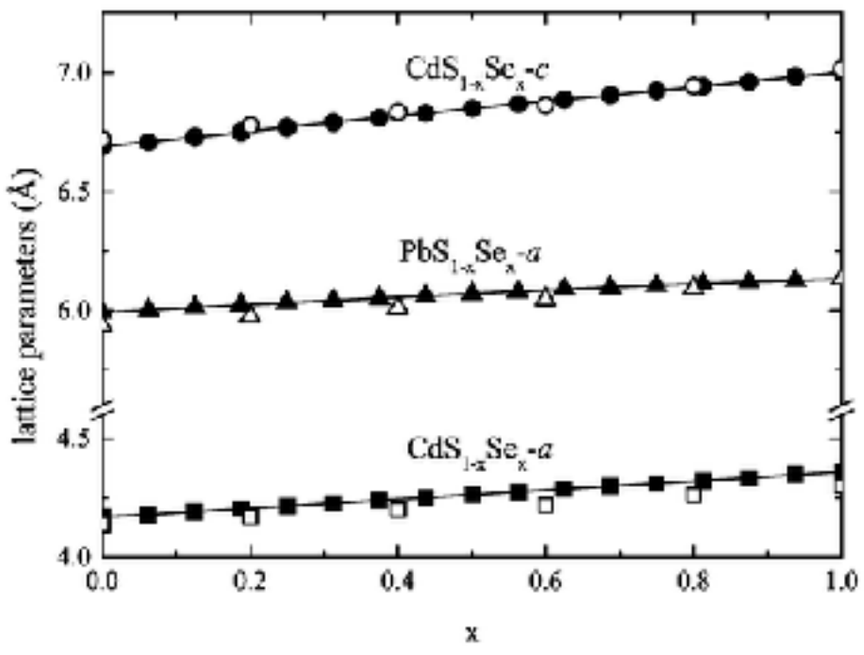
- Predict optimized coordinates from starting coordinates.
- Capture physically insignificant, small optimizations (<0.3 Å).
- Work with relatively few data (~10^1 runs).



Background: Available Force Fields

- Conventional molecular mechanics force fields
  - No commercially available ones for Pb and S
  - Not fitted to nanostructures & not accurate enough

force on one complex	RMS error (kcal/mol/Å)
composite	19.6
dot size 1	21.4
dot size 2	23.0
dot size 3	16.8
dot size 4	20.9
dot size 5	18.6

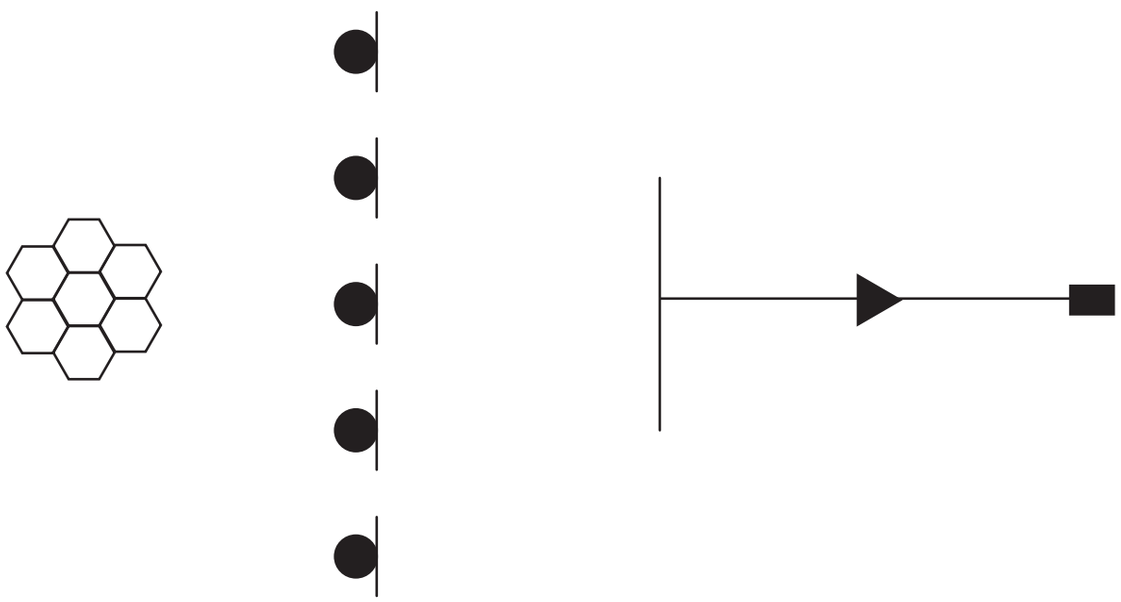


Background: Available Force Fields

- Conventional force fields
  - No commercially available ones for PbS
  - Some active research going on
- Neural network potentials
  - Recipes exist, but no ready-made ones for PbS
  - 0.1eV / Å force accuracy at 10^4 data points

Background: the Canonical NN potential

$$\overrightarrow{x_1},\overrightarrow{x_2},\ldots,\overrightarrow{x_n} \rightarrow f_1,f_2,\ldots,f_m \rightarrow E \mid \vec{F} \rightarrow \overrightarrow{dx}$$



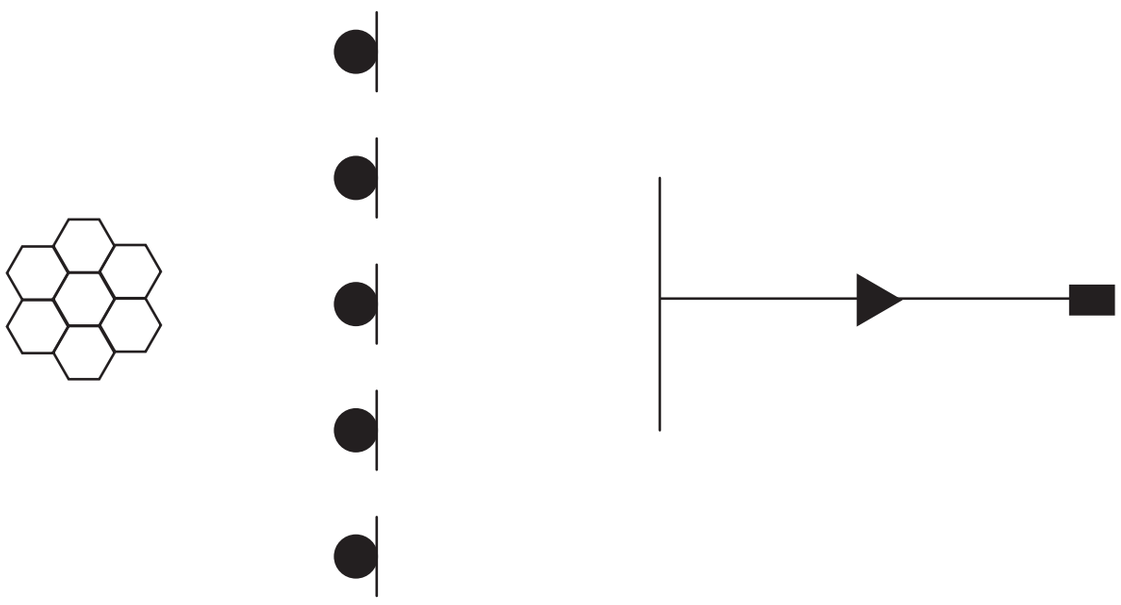
Symmetry function features:

$$f_2 = \sum_{i,j}(1+\lambda\cos\theta_{ij})^{\zeta} e^{-\eta(r_i^2+r_j^2)} \cos\pi(\frac{r_i}{R_C}+1)\cos\pi(\frac{r_j}{R_C}+1)$$

$$E = ANN(f_2), \vec{F} = -\frac{\partial E}{\partial \vec{x}}$$

Model

$$\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n} \rightarrow f_1, f_2, \ldots, f_m \rightarrow E \mid \vec{F} \rightarrow \overrightarrow{dx}$$



Symmetry function features:

$$f_2 = \sum_{i,j} (1 + \lambda \cos\theta_{ij})^{\zeta} e^{-\eta(r_i^2 + r_j^2)} \cos\pi(\frac{r_i}{R_C} + 1) \cos\pi(\frac{r_j}{R_C} + 1)$$

$$E = ANN(f_2), \vec{F} = -\frac{\partial E}{\partial \vec{x}}$$

Every permutationally invariant function can be expressed as a cluster expansion of functions representable by neural nets:

$$E = \sum_i h_1(x_i) + \sum_{ij} h_2(x_i, x_j) + \cdots$$

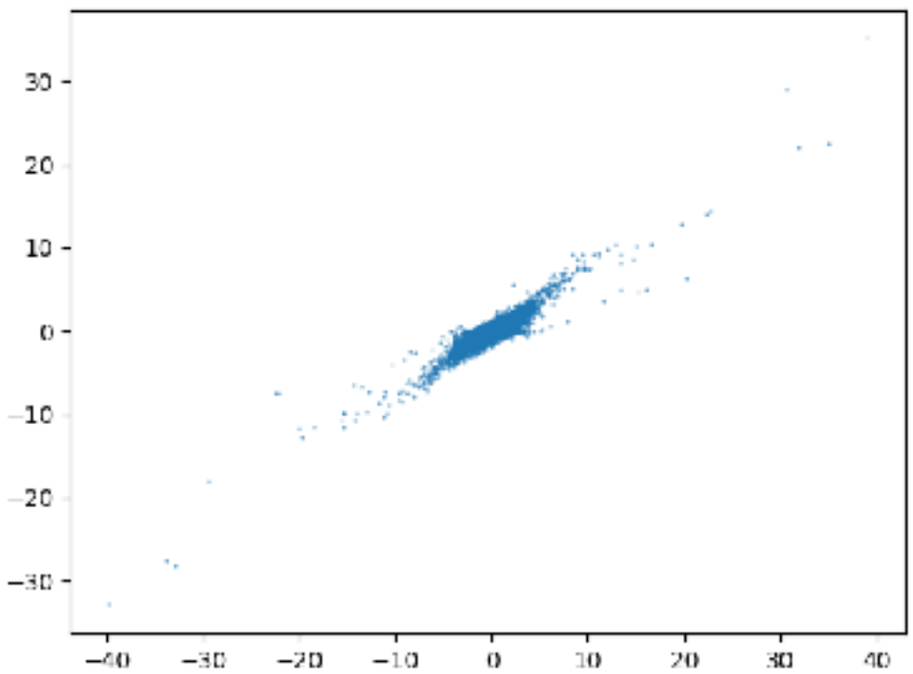
Components of a normal sum does not 'interact' with each other. Thus 1-order cluster expansion accounts for 1-body interaction.

LSTM neural networks can add long-range correlation as well as alleviate exploding gradients in long sums.

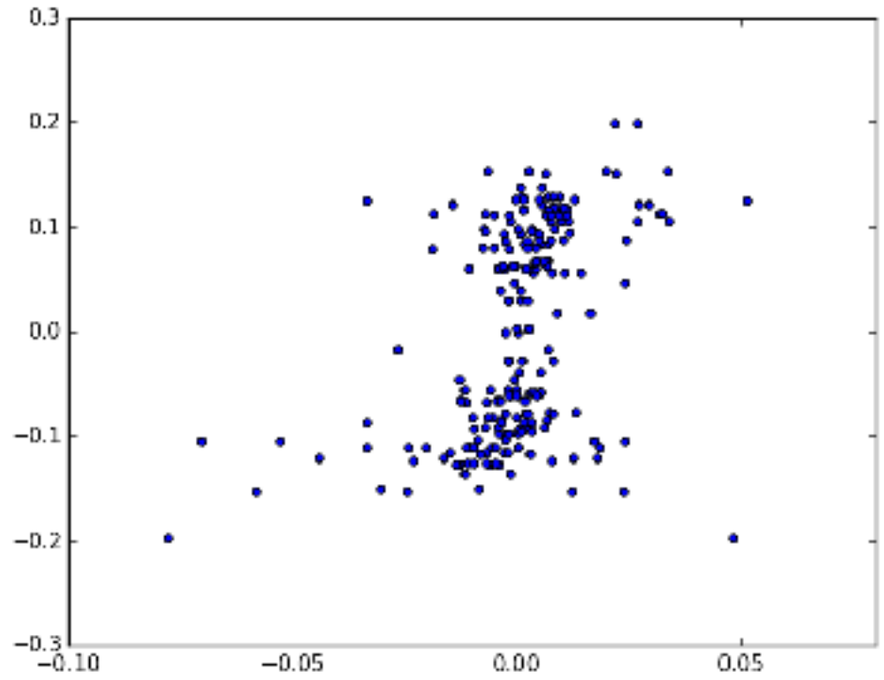
$$E_{LSTM} = \sum_i {}^{LSTM} h(x_i)$$

Result: F({xi}) and dx({xi})

Predicted vs. actual forces



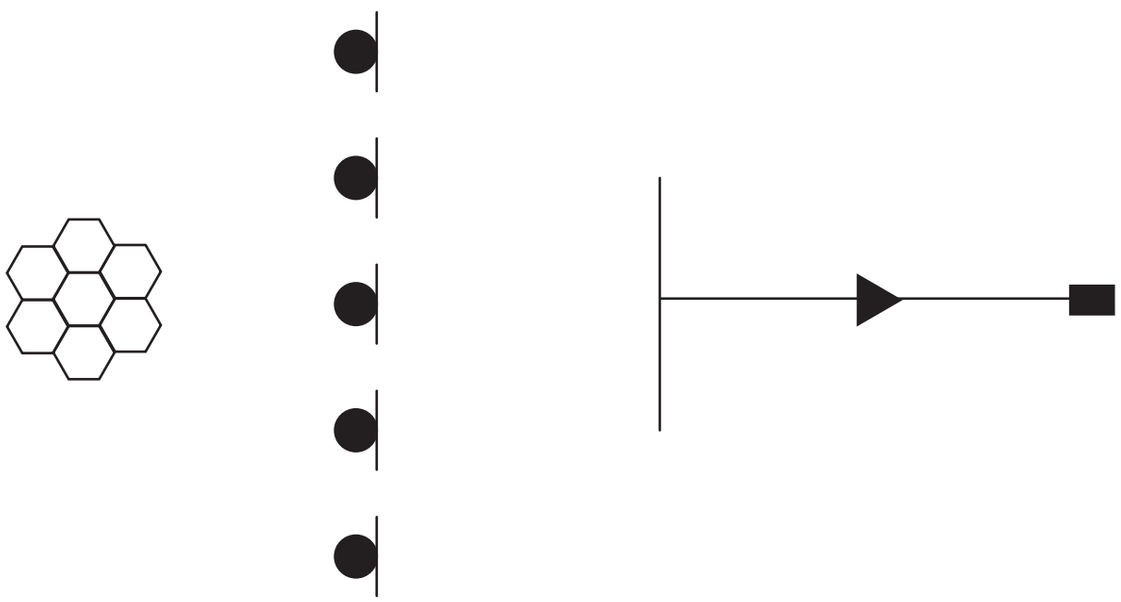
Predicted vs. actual optimized coordinates
(time per run ~ 4h)



Training set: 91 quantum dots (single-point force calculation).
Test set: 1 quantum dot (geometry optimization).

Model

$$\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n} \rightarrow \overrightarrow{dx}$$



For dx({x_i}), coordinates are discrete. Instead of learning a infinite-degree-of-freedom function, we now only have to learn a finite set of values.

$$\overrightarrow{dx} = \sum_i \overrightarrow{h_1}(x_i) = \sum_i \overrightarrow{h_i}$$

Symmetries can be included:

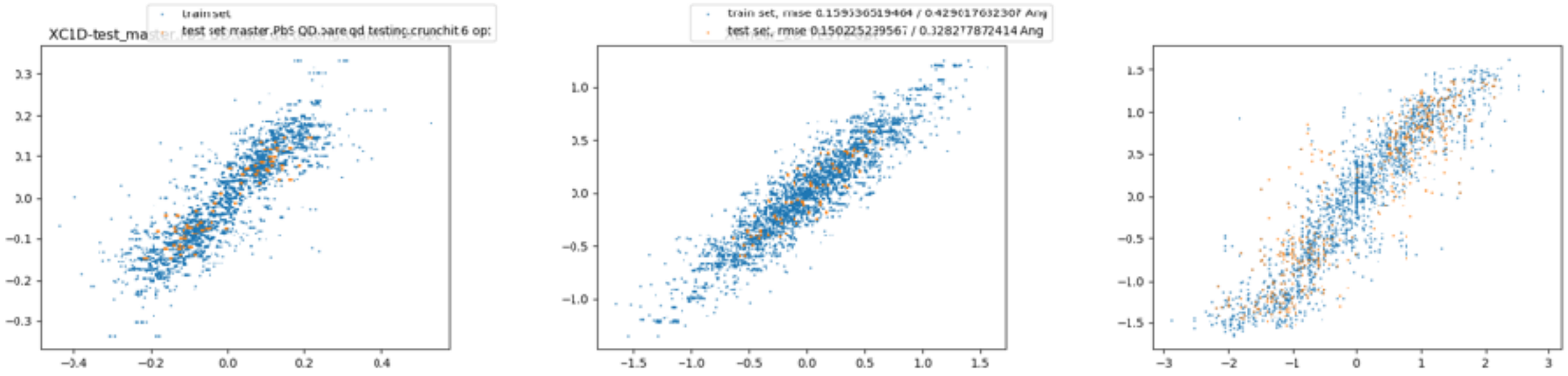
$$\overrightarrow{dx}(\overrightarrow{x_i}) = dx(|x_i|) \cdot \widehat{x_i}$$

which is extendable to higher dimensions:

$$h_{x_i, x_j} = h_{ij1}\widehat{x_i} + h_{ij2}\widehat{x_j} + h_{ij3}(\widehat{x_i} \times \widehat{x_j})$$

Result: dx({xi})

Predicted vs. actual (x_i - x_i^0) /Å, training and test set



Training set: 13 quantum dots (geometry optimization).
Test set: 1 quantum dot (geometry optimization).
Time per run: ~ 20s

Summary

- Machine learning models (MLP, LSTM, Lasso) for pre-optimizing quantum dot structures

- Small range, high accuracy, low data requirement, at the cost of generality

Thank you