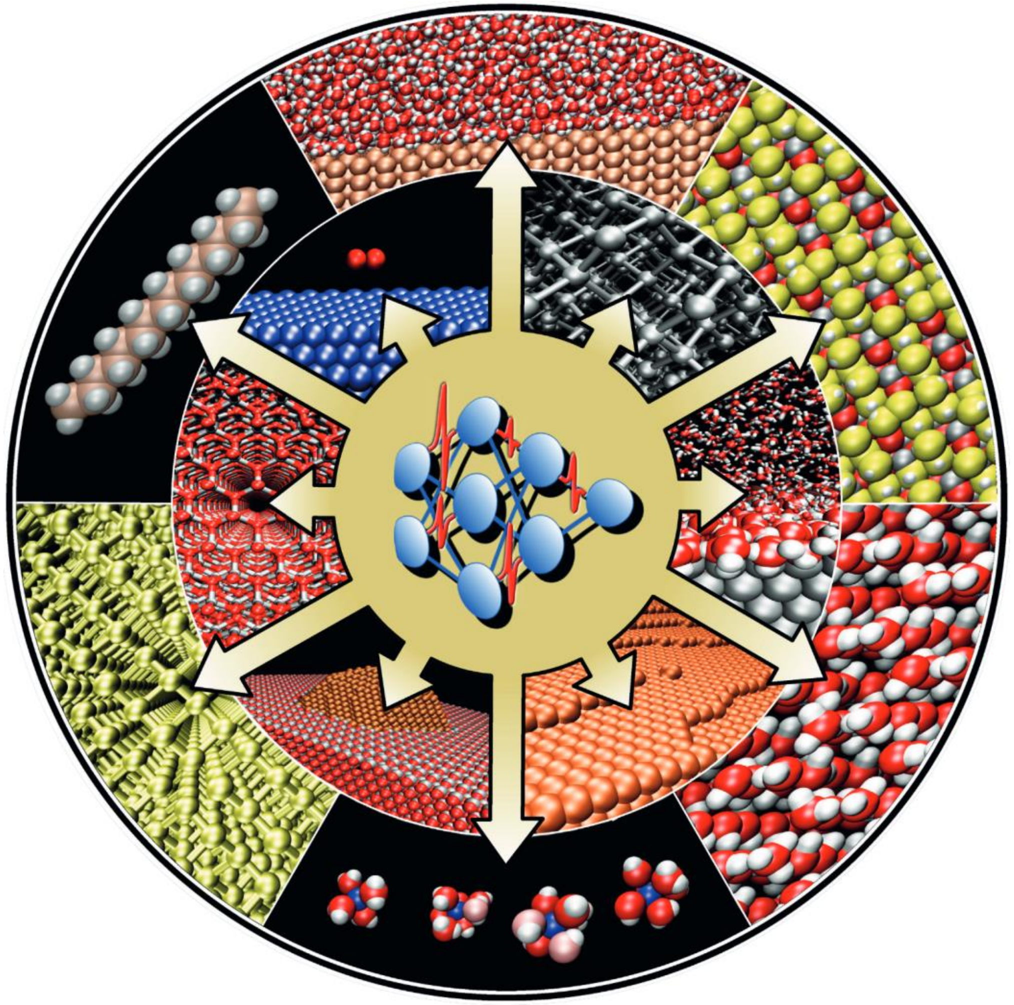


Image credits: [1]

Accessing large-scale molecular simulations through machine learning

Flavio Giuliani

1st year PhD student in
Computational Condensed Matter Physics

Supervisors:
Prof. Riccardo Mazzarello
Prof. Lilia Boeri

Overview

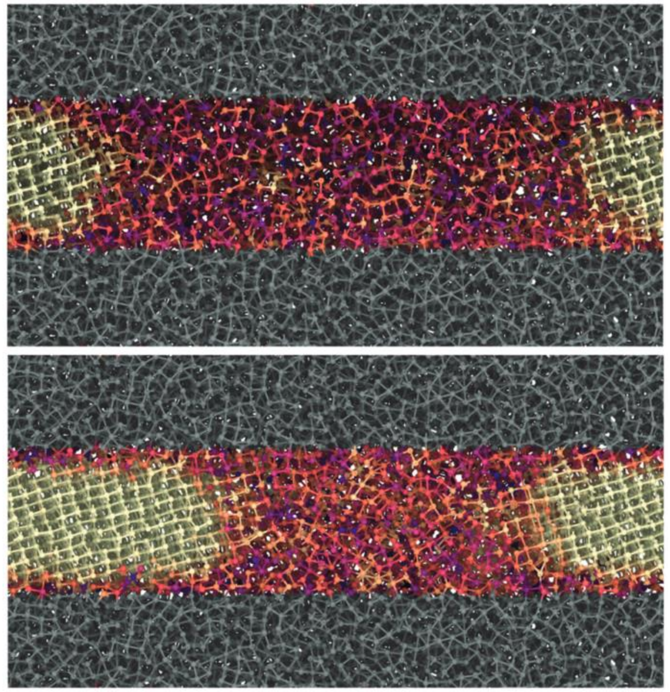
- Why large-scale?
- The bottleneck of molecular simulations.
- Short introduction to Neural Networks.
- Neural-Network methods for molecular simulations.

- My research topic: application to *Phase-Change materials*.

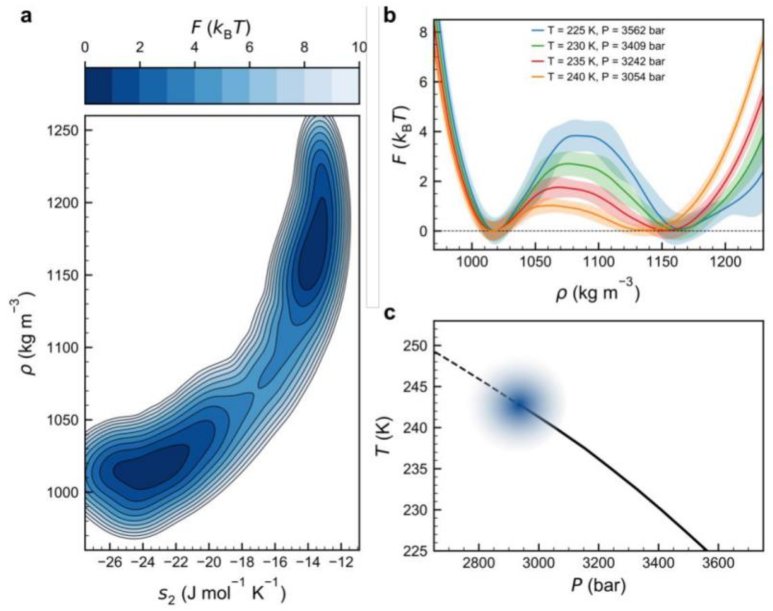
Why large scale?

Simulation of crystals can be done on small scale (unit cell), because of their spatial symmetry.

But amorphous/disordered systems (liquids, glasses, biomolecules, complex systems) require large spatial and time scales:


Growth of the crystal front in thin Sb. Dragoni et al., Nanoscale (2021).

- Long-wavelength vibrational modes.
- Slow relaxation dynamics near criticality.
- Study of collective phenomena.
- Free energy computations.
- Reduction of finite-size effects.



Liquid-Liquid critical point for a water model. Gartner et al., Phys. Rev. Lett. (2022).

Born-Oppenheimer approximation

Electronic+nuclear (e+n) Schrödinger equation:

$$H_{e+n}\psi_E = E\psi_E$$

$$H_{e+n} = [T_e + V_{ee} + V_{en} + V_{nn}] + T_n$$

Legend

$T_a =$ Kinetic energy of system a.

$V_{ab} =$ Coulomb interaction between systems a,b.

$N_e, N =$ Number of electrons/nuclei.

$r^N = \{\mathbf{r}_i\}_{i=1}^N$ Spatial coordinates.

separation of energy/time scales

$$m_n/m_e > 200$$

$\Longrightarrow$

separate the wavefunction:

$$\psi_E(r^{N_e}, r^N) = \sum_\nu \phi_{E\nu}(r^{N_e}|r^N)F_\nu(r^N)$$

Solve the electron problem first, at fixed nuclear positions:

$$[T_e + V_{ee} + V_{en} + V_{nn}]\,\phi_E(r^{N_e}|r^N) = U(r^N)\phi_E(r^{N_e}|r^N)$$

Classical Dynamics of the nuclei

The electrons induce an **effective interaction potential** on the nuclear problem:

$$\left[T_n + U(r^N)\right] F(r^N) = E F(r^N)$$

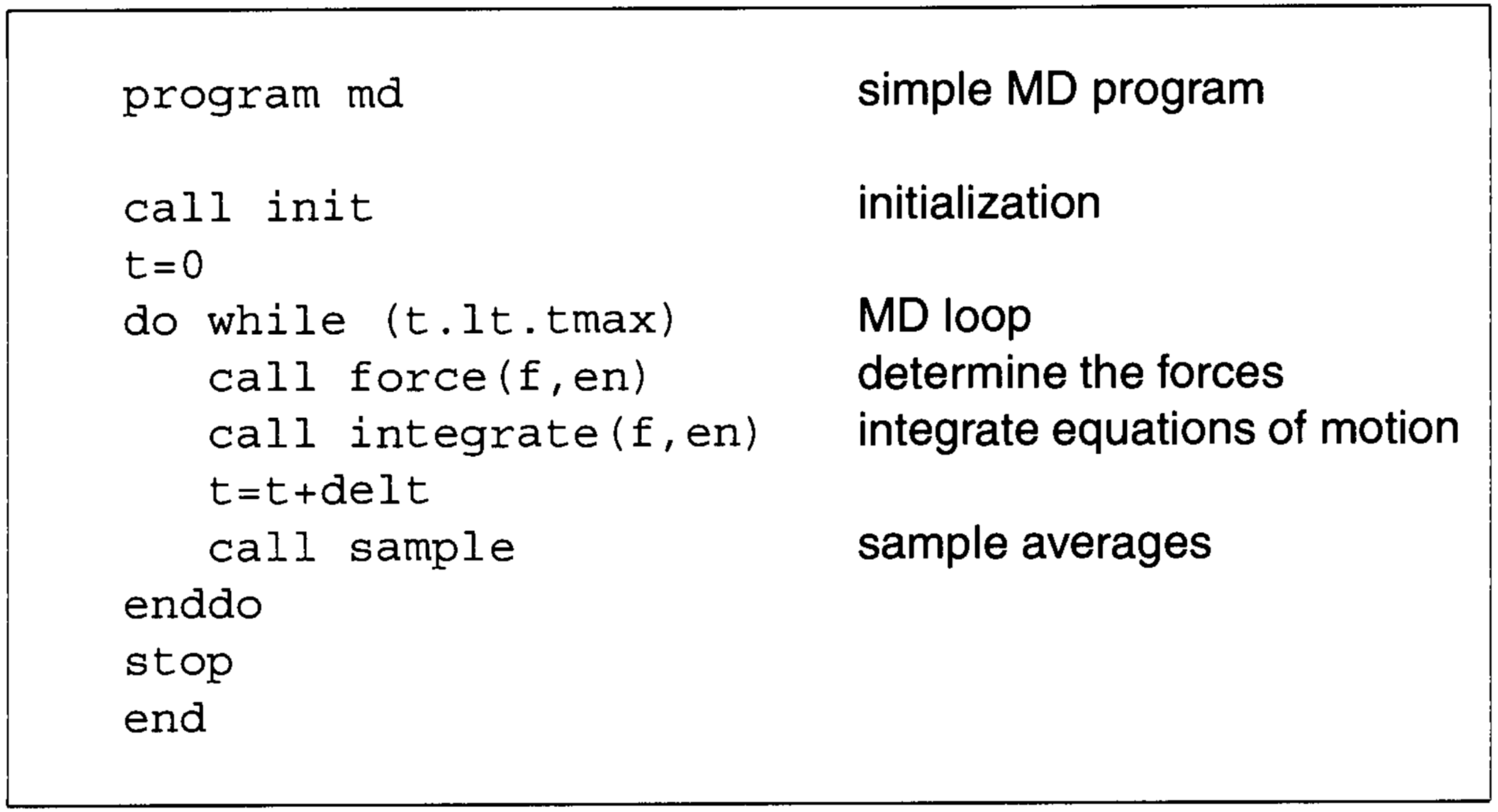
Effective Hamiltonian for the nuclei: $H = T_n + U(r^N) = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U(\mathbf{r}_1, \ldots, \mathbf{r}_N)$

In a classical treatment, the dynamics follows Hamilton equations:

$$\begin{cases} \dot{r}_{i,\alpha} = \dfrac{\partial H}{\partial p_{i,\alpha}} = \dfrac{p_{i,\alpha}}{m_i} \\ \dot{p}_{i,\alpha} = -\dfrac{\partial H}{\partial r_{i,\alpha}} = -\dfrac{\partial U}{\partial r_{i,\alpha}} \end{cases}$$

Classical Molecular Dynamics algorithm

Pseudocode (from [2])

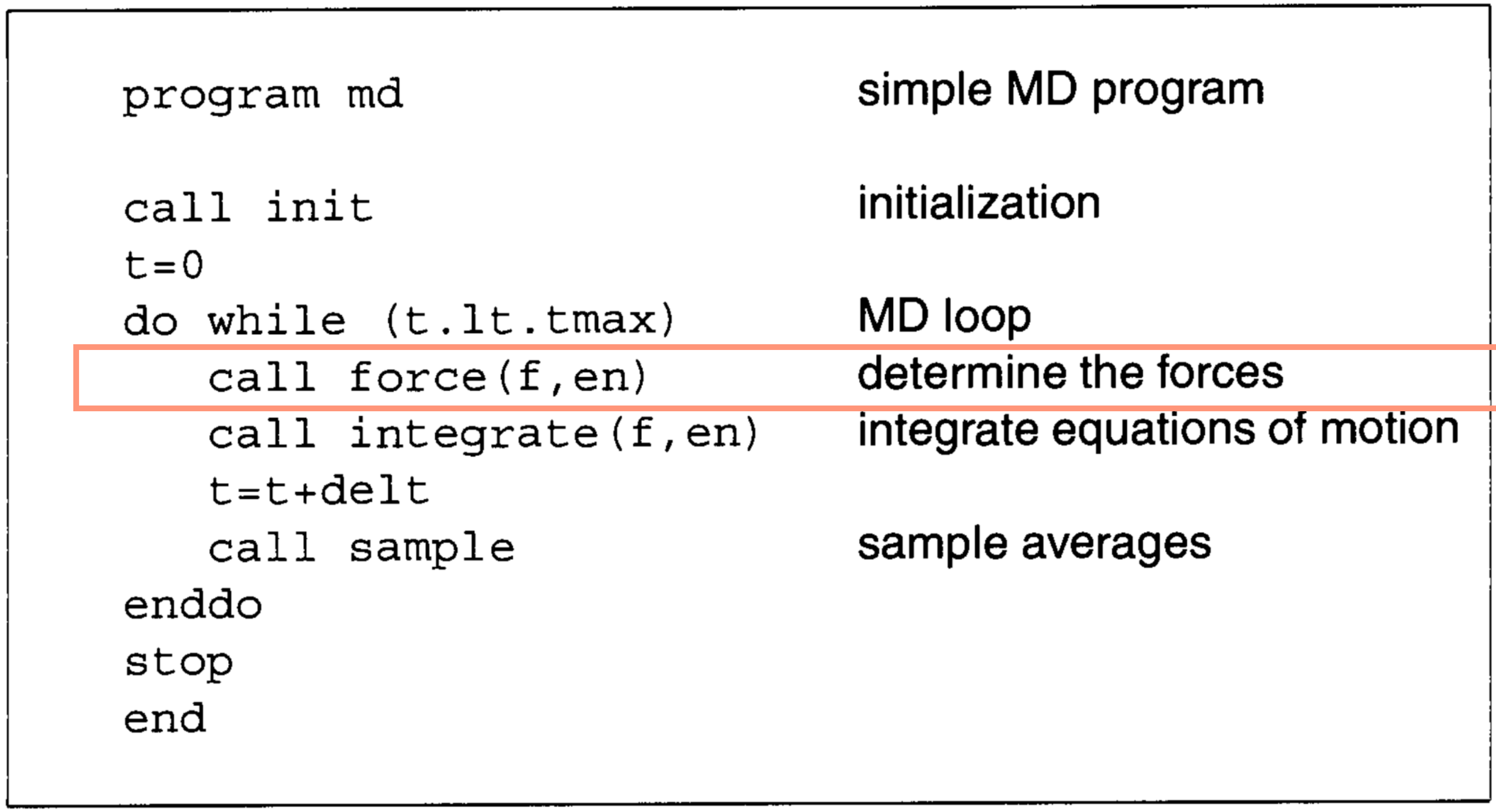


```
program md                          simple MD program

call init                           initialization
t=0
do while (t.lt.tmax)                MD loop
   call force(f,en)                 determine the forces
   call integrate(f,en)             integrate equations of motion
   t=t+delt
   call sample                      sample averages
enddo
stop
end
```

Classical Molecular Dynamics algorithm

Pseudocode (from [2])



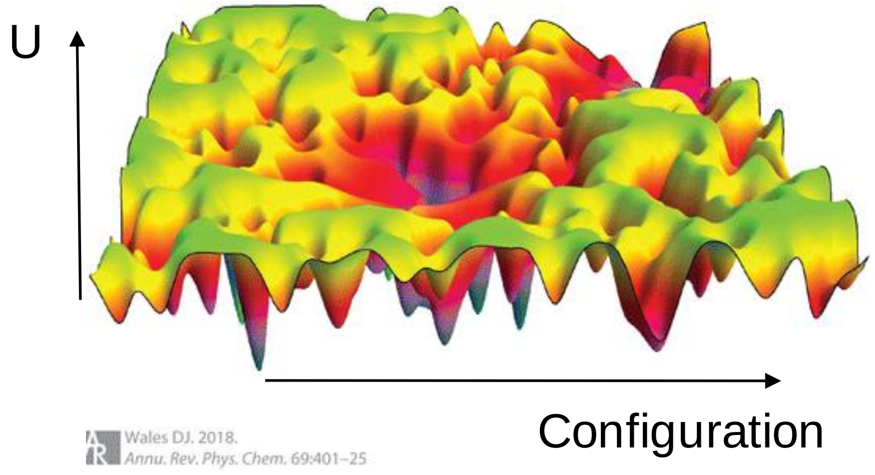
```
program md                        simple MD program

call init                         initialization
t=0
do while (t.lt.tmax)              MD loop
   call force(f,en)               determine the forces
   call integrate(f,en)           integrate equations of motion
   t=t+delt
   call sample                    sample averages
enddo
stop
end
```

Bottleneck!

Interaction Potential

a.k.a. Potential Energy Surface (PES)

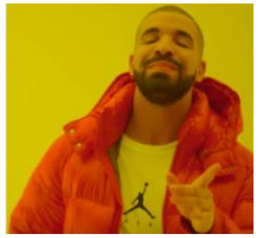


PES U(r)

For example,
Density Functional
Theory (DFT)

Ab-Initio
method

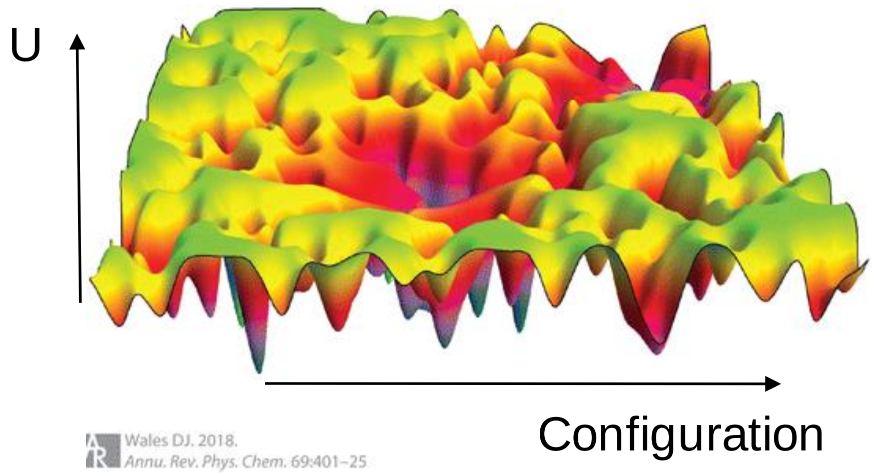
Slow implementation.



Accurate PES.
Dynamical chemical bonds.

Interaction Potential

a.k.a. Potential Energy Surface (PES)



PES U(r)

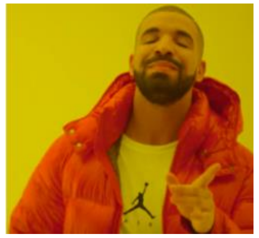
For example, Density Functional Theory (DFT)

Ab-Initio method

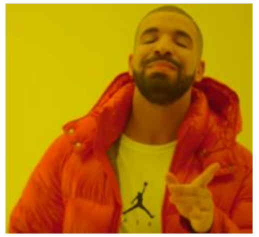
Empirical parametrization

For example, Lennard-Jones pairwise potential.

Slow implementation.



Accurate PES.
Dynamical chemical bonds.

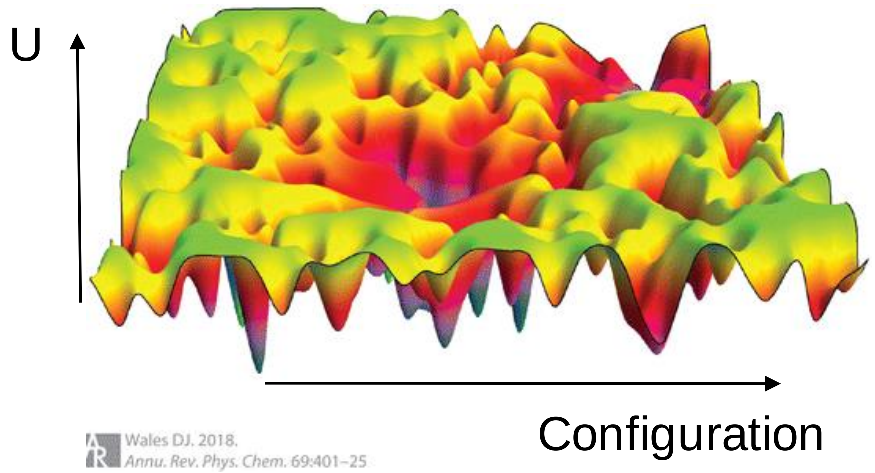


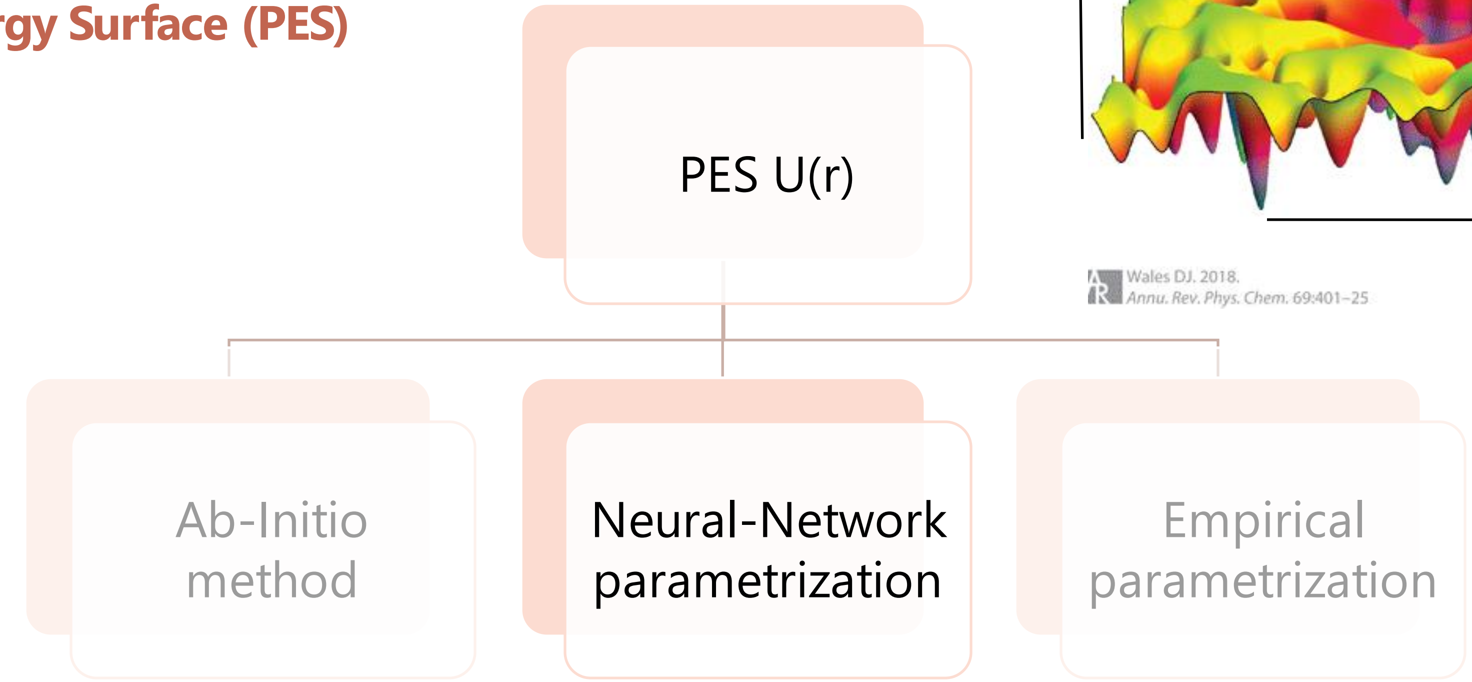
Fast implementation.

Inaccurate PES (usually).
Static chemical bonds.

Interaction Potential

a.k.a. Potential Energy Surface (PES)





For example, Density Functional Theory (DFT)

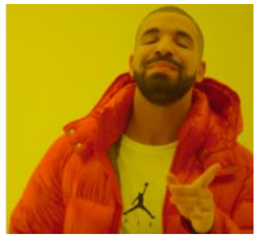
For example, Lennard-Jones pairwise potential.

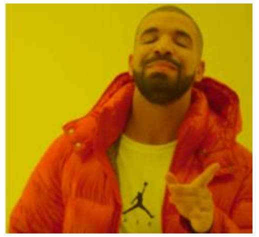
Slow implementation.

Accurate.
Dynamical chemical bonds.

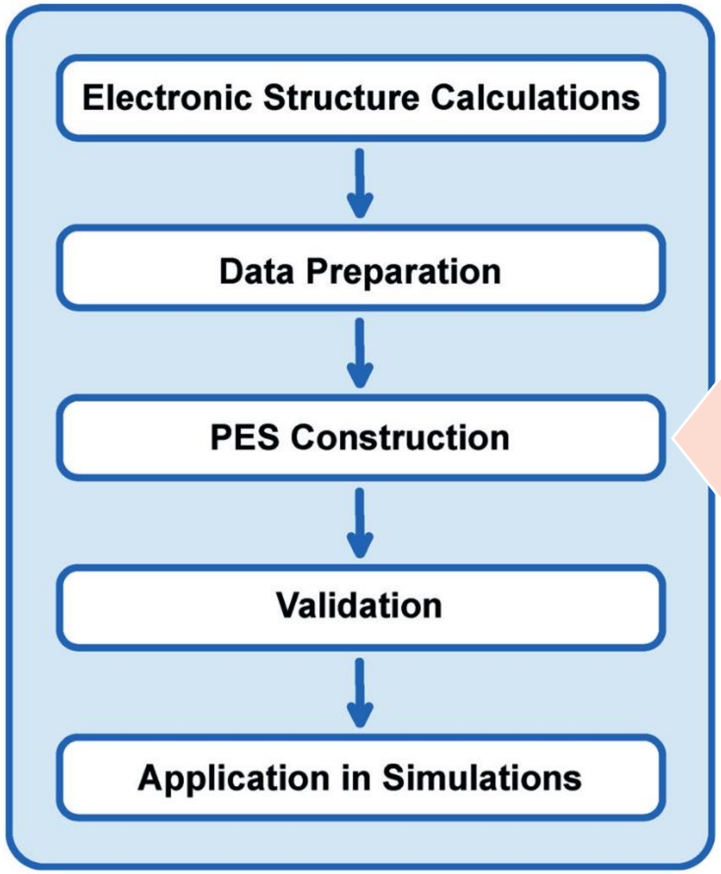
Fast implementation.

Inaccurate in most cases.
Static chemical bonds.



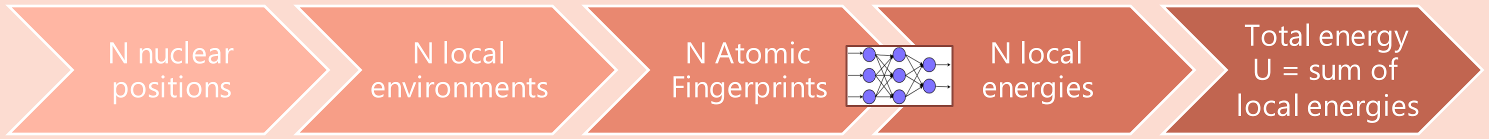


How to build a Neural-Network model for the PES



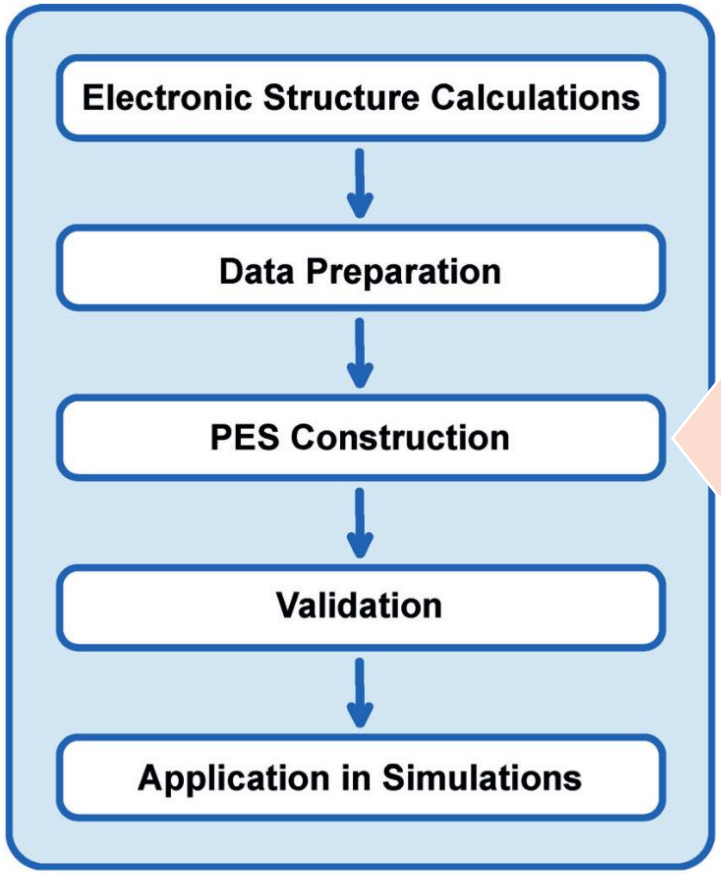
Flow chart from [1].

Model:

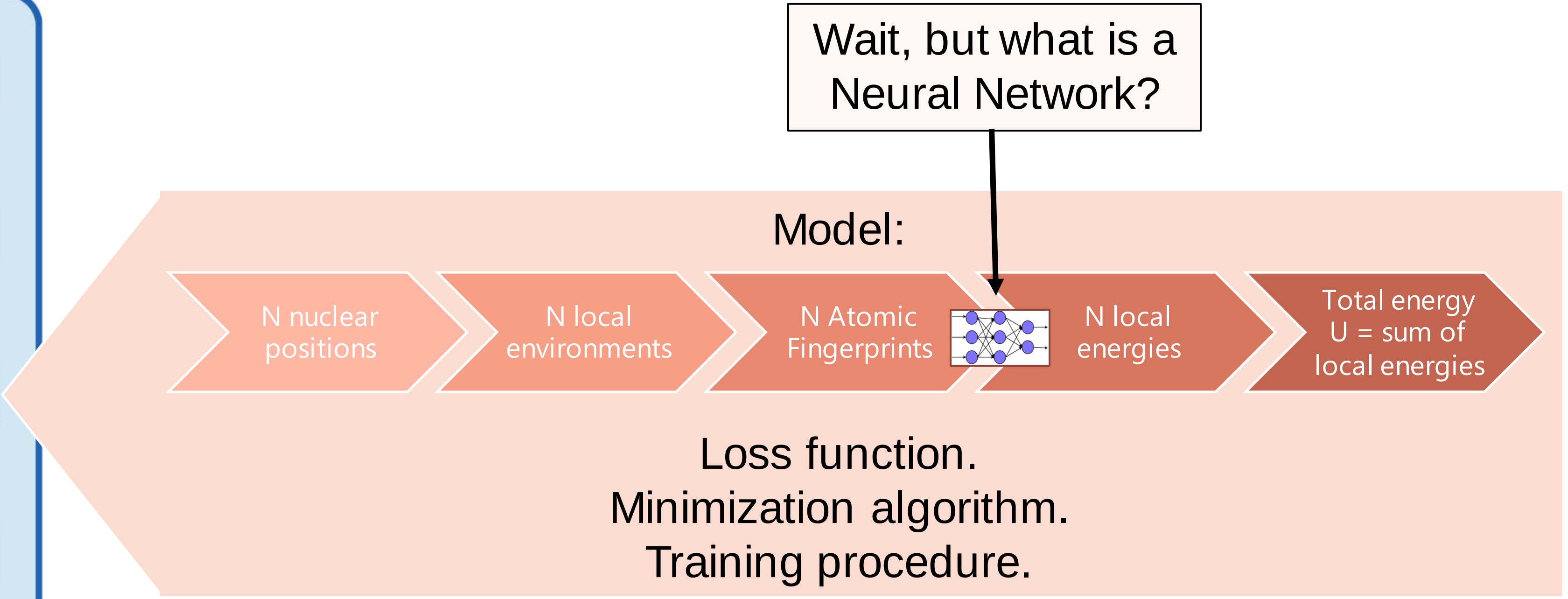


Loss function.
Minimization algorithm.
Training procedure.

How to build a Neural-Network model for the PES



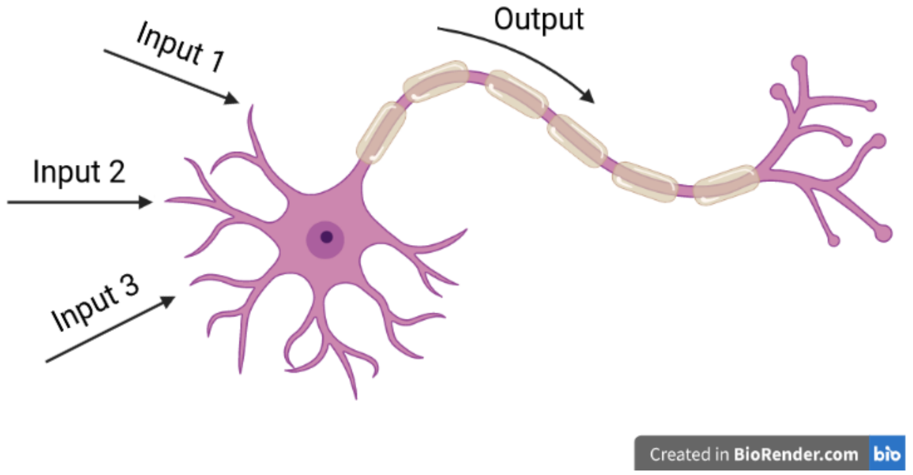
Flow chart from [1].



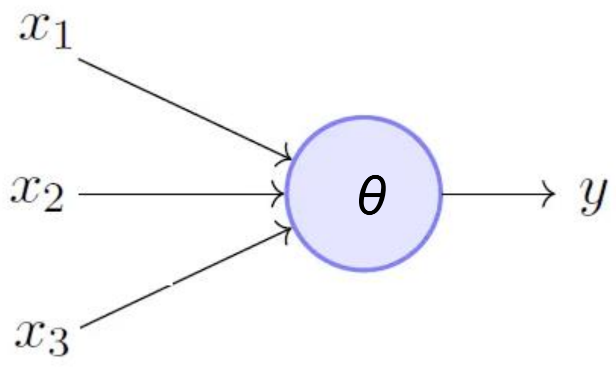
Loss function.
Minimization algorithm.
Training procedure.

Basics of Artificial Neural Networks

Biological neuron



McCulloch & Pitts (1943) model

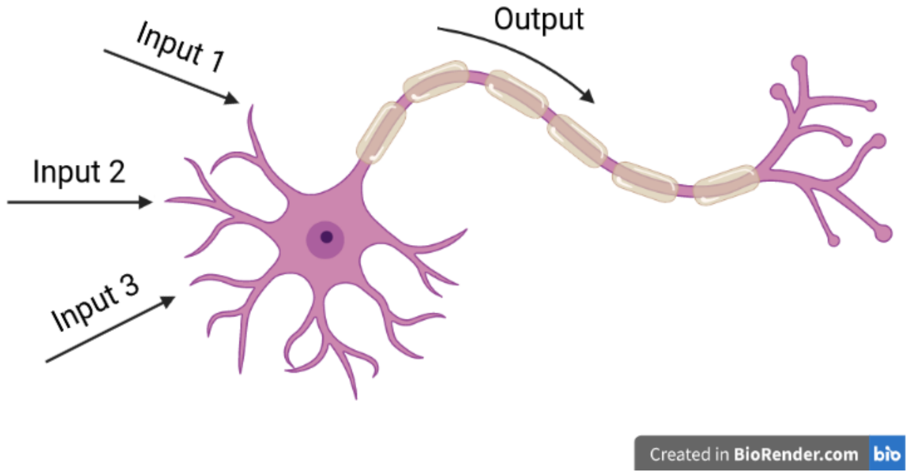


$$y = f(\mathbf{x}|\theta) = \begin{cases} 1 & \text{if} \quad \sum_k x_k > \theta \quad \text{and neuron is active} \\ 0 & \text{otherwise} \end{cases}$$

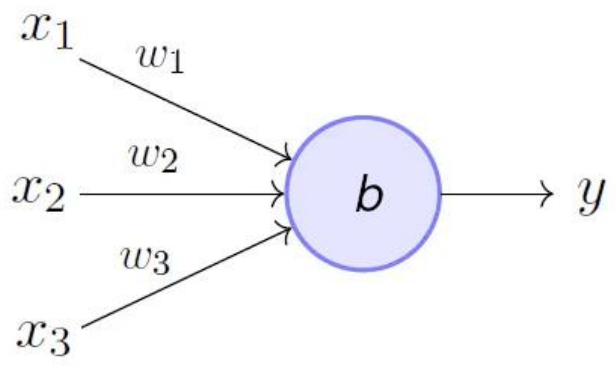
binary input/output with a sum threshold θ

Basics of Artificial Neural Networks

Biological neuron



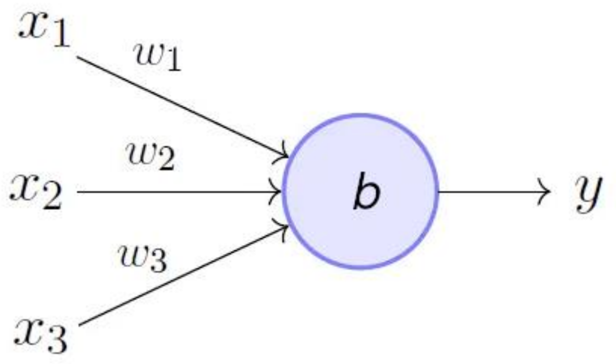
Rosenblatt's *Perceptron* (1958) model



$$y = f(\mathbf{x}|\mathbf{w}, b) = \begin{cases} 1 & \text{if} \quad \sum_k w_k x_k + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

linear mapping + non-linear threshold

Perceptron



$$y = f(\mathbf{x}|\mathbf{w}, b) = \sigma\left(\underbrace{\mathbf{w}^T \cdot \mathbf{x} + b}_{\text{linear mapping with weights } (\mathbf{w},b)}\right)$$

non-linear activation function σ (e.g. tanh, sigmoid, …)

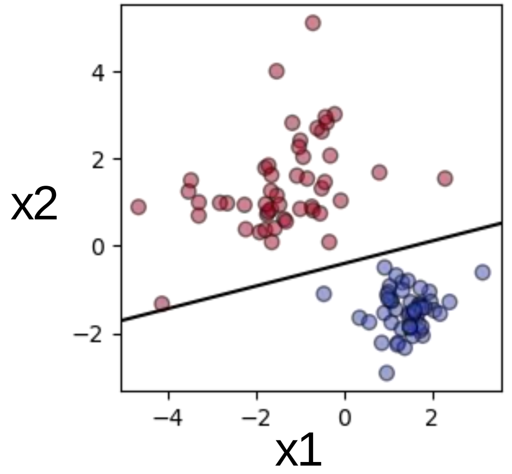
linear mapping with weights (**w**,b)

The decision boundary is a **hyperplane**:

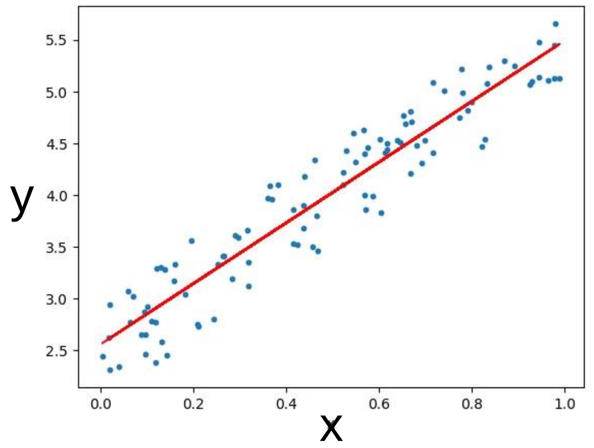
$$b + \mathbf{w}^T\mathbf{x} = 0$$

The task of finding the optimal parameters (**w**, b) for classifying **x** can be seen both as:

- **Linear classification**: find the best hyperplane dividing the two classes.

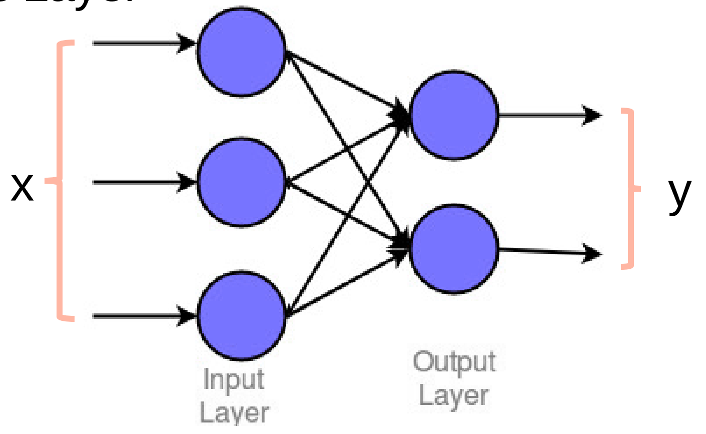


- **Linear regression:** find the best fitting hyperplane to the boundary between the two classes.



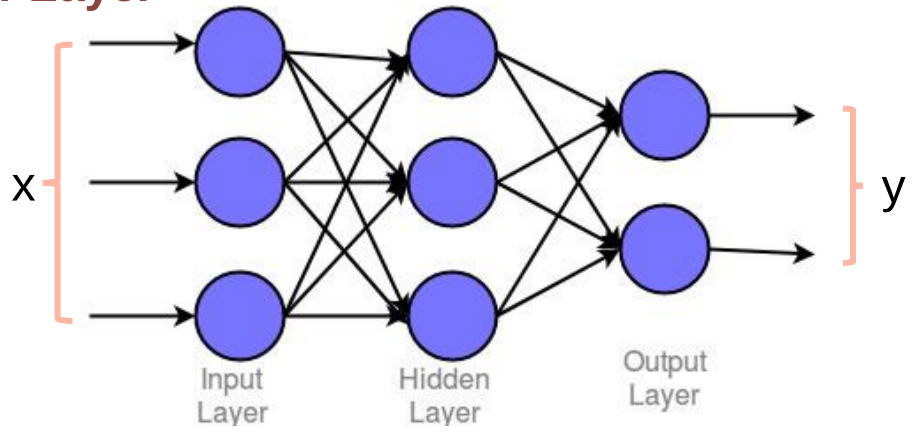
Non-linear Regression with Multi-Layer Perceptrons (MLP)

Single Layer



Non-linear Regression with Multi-Layer Perceptrons (MLP)

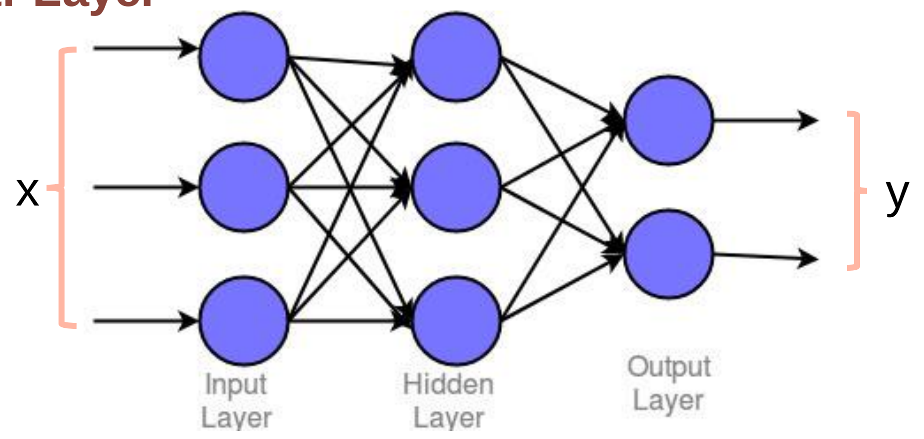
Multi-Layer

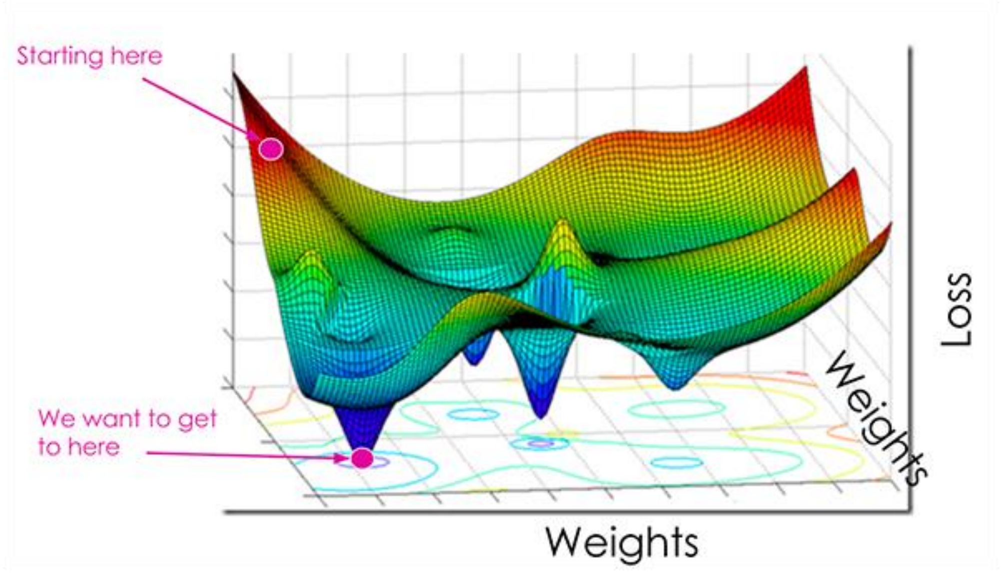


Universal approximation theorem:
A MLP is a universal approximator, if deep enough.

Non-linear Regression with Multi-Layer Perceptrons (MLP)

Multi-Layer





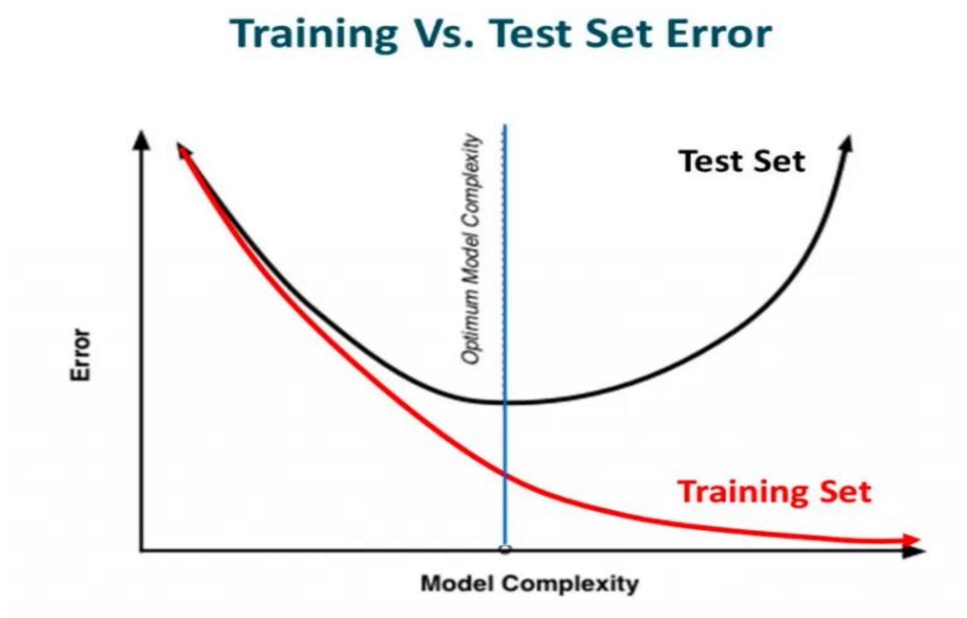
Universal approximation theorem:
A MLP is a universal approximator, if deep enough.

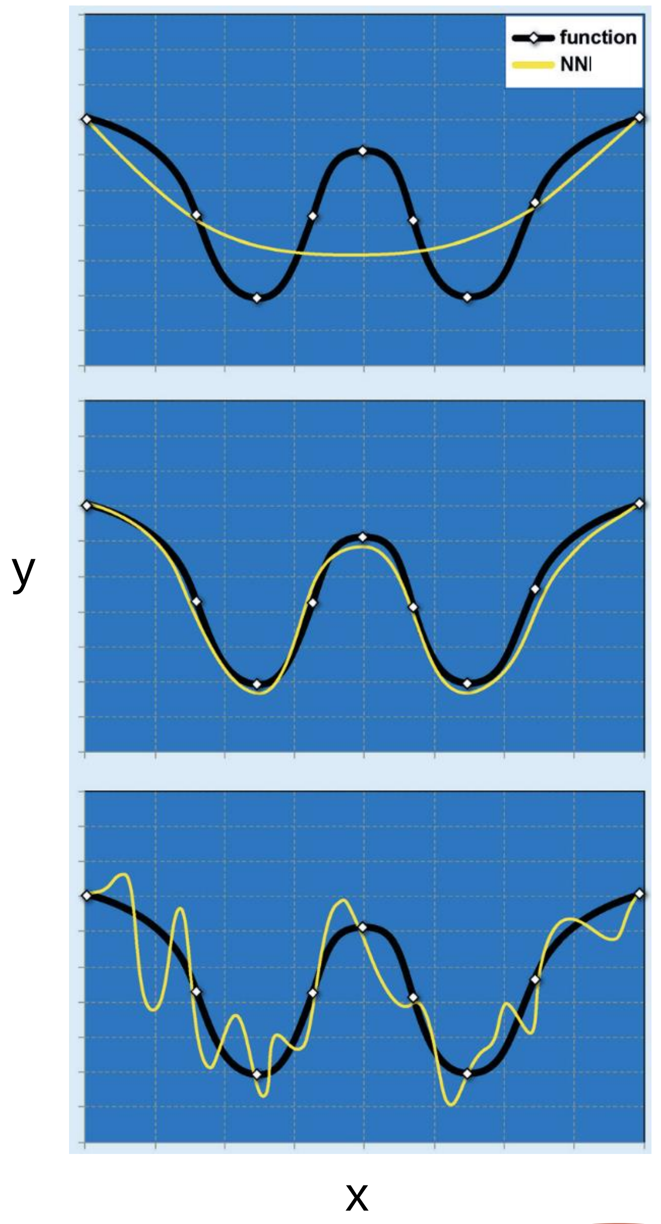
How to find the optimal parameters, at fixed network architecture?

1. Define a loss function L(w) between the target y and the prediction $\hat{y} = f(x|w)$.
2. Use a minimization algorithm on L(w).

Common problems in NN applications

- Overfitting and generalization problems

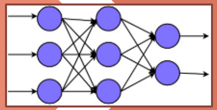




Common problems in NN applications

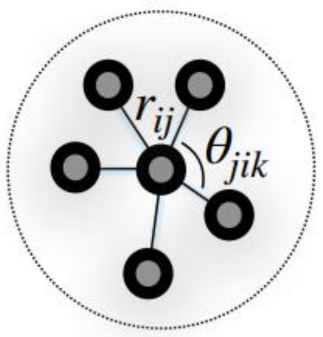
- Overfitting and generalization problems
- Dataset size and variability
- Choice of the input features

Neural-Network Interaction Potential

N nuclear positions → N local environments → N Atomic Fingerprints →  → N local energies → Total energy U = sum of local energies

$$\mathbf{r}^N = \{\mathbf{r}_i\}_{i=1}^N$$

$$\partial_i =$$



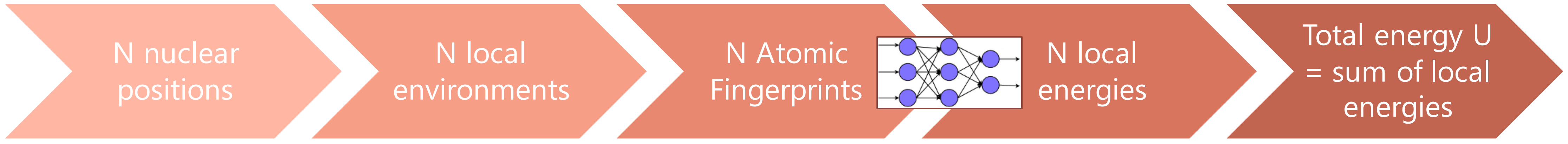
Input of the Neural Network.

Output of the Neural Network.

They must be **differentiable**, to compute derivatives of U (forces and stress tensor).

The form of U is invariant to particle permutations and independent of system size.

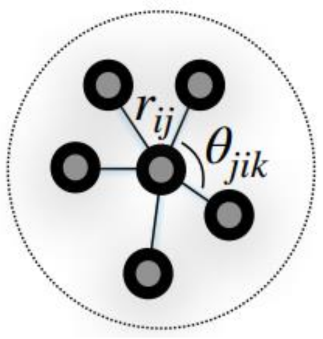
Neural-Network Interaction Potential



$$\mathbf{r}^N = \{\mathbf{r}_i\}_{i=1}^N$$

This is where methods get different!

$$\partial_i =$$



The form of U is invariant to particle permutations and independent of system size.

Behler-Parrinello (2007)

N nuclear positions → N local environments → N Atomic Fingerprints → N local energies → Total energy U = sum of local energies

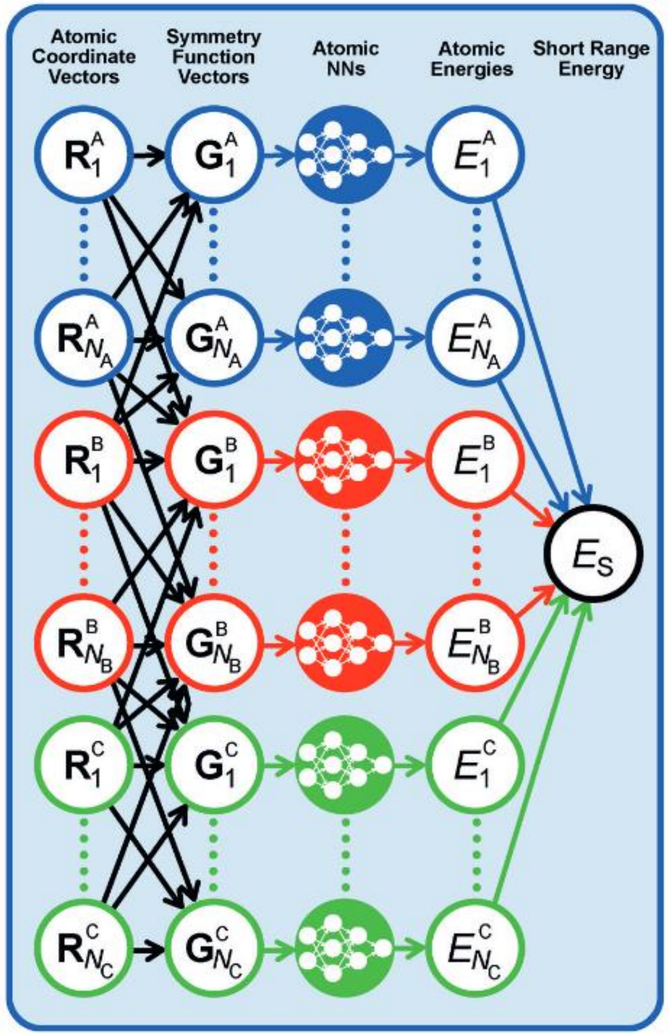
2-body and 3-body symmetry functions with **manually-tuned** parameters.

$$G_i^1 = \sum_{j\neq i}^{\text{all}} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij}).$$

$$G_i^2 = 2^{1-\zeta} \sum_{j,k\neq i}^{\text{all}} (1+\lambda\cos\theta_{ijk})^{\zeta} \times e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk})$$

with $$f_c(R_{ij}) = \begin{cases} 0.5\times\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right)+1\right] & \text{for } R_{ij}\le R_c \\ 0 & \text{for } R_{ij} > R_c. \end{cases}$$

A MLP for each species, each with 2 hidden layers of ~40 nodes.

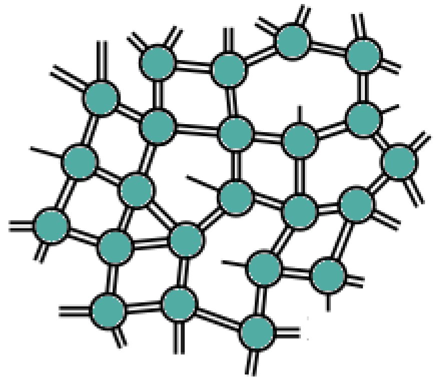


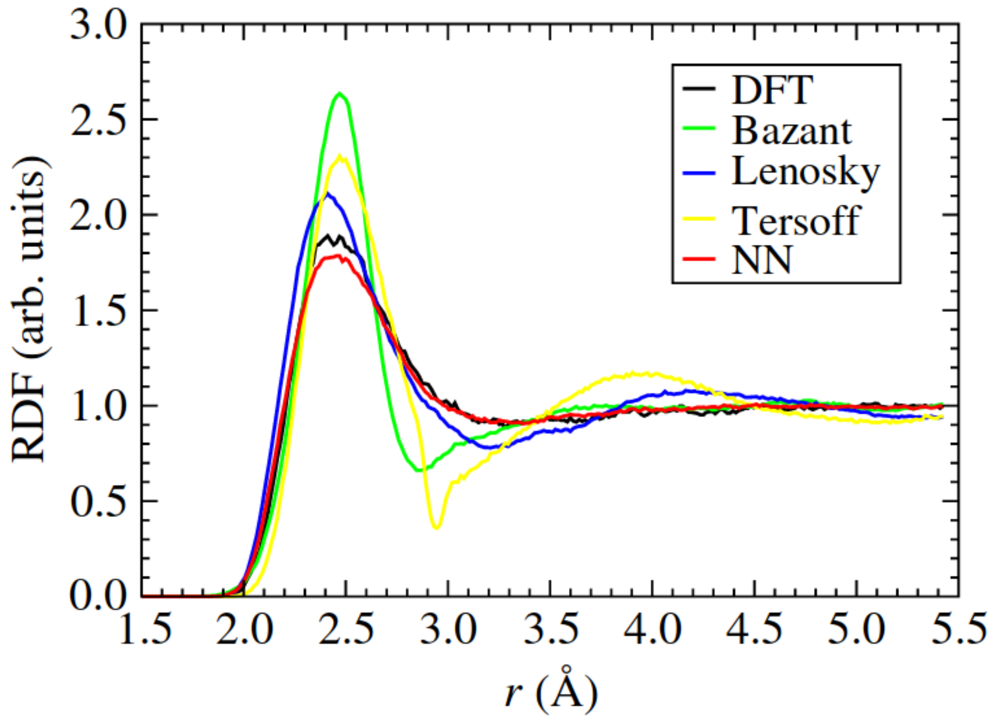
Example architecture for a 3-species system.

Good results on Silicon

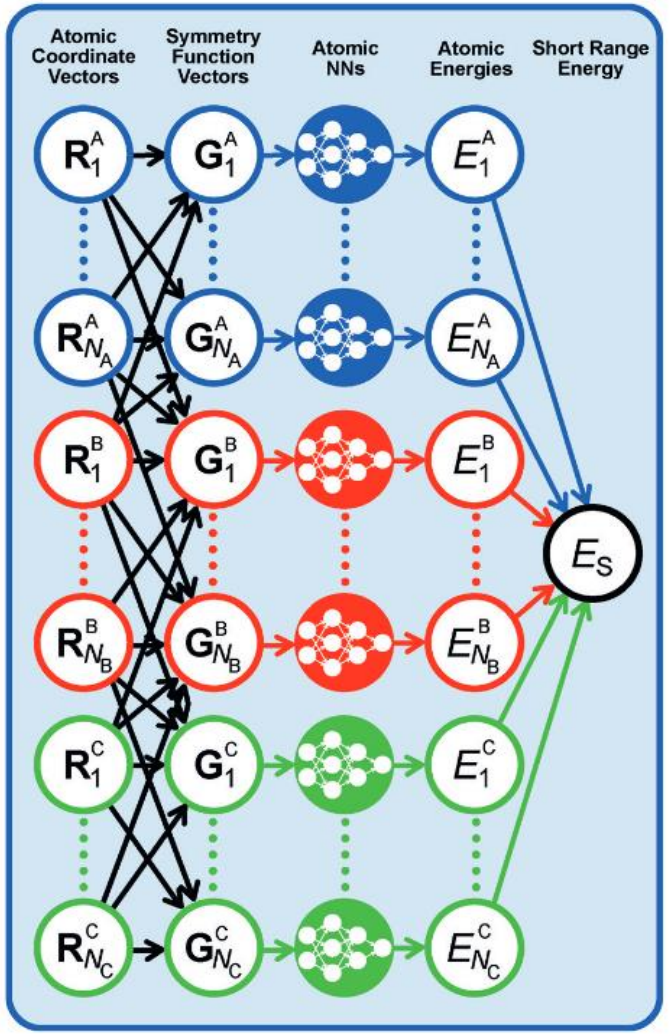
with 48 symmetry functions

Energy error: ~ 5 meV/atom
Force error: ~ 200 meV/Å

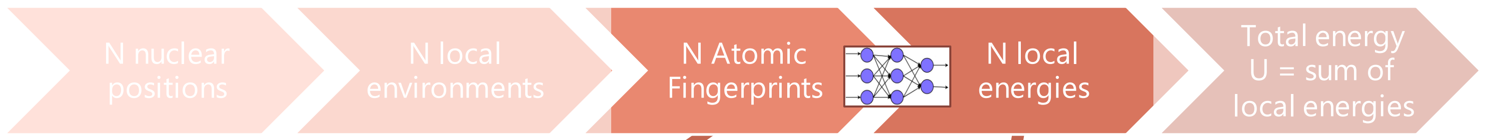




Predicted Radial pair correlation (red) compares well with the DFT one (black).



Mattioli-Sciortino-Russo (2023)



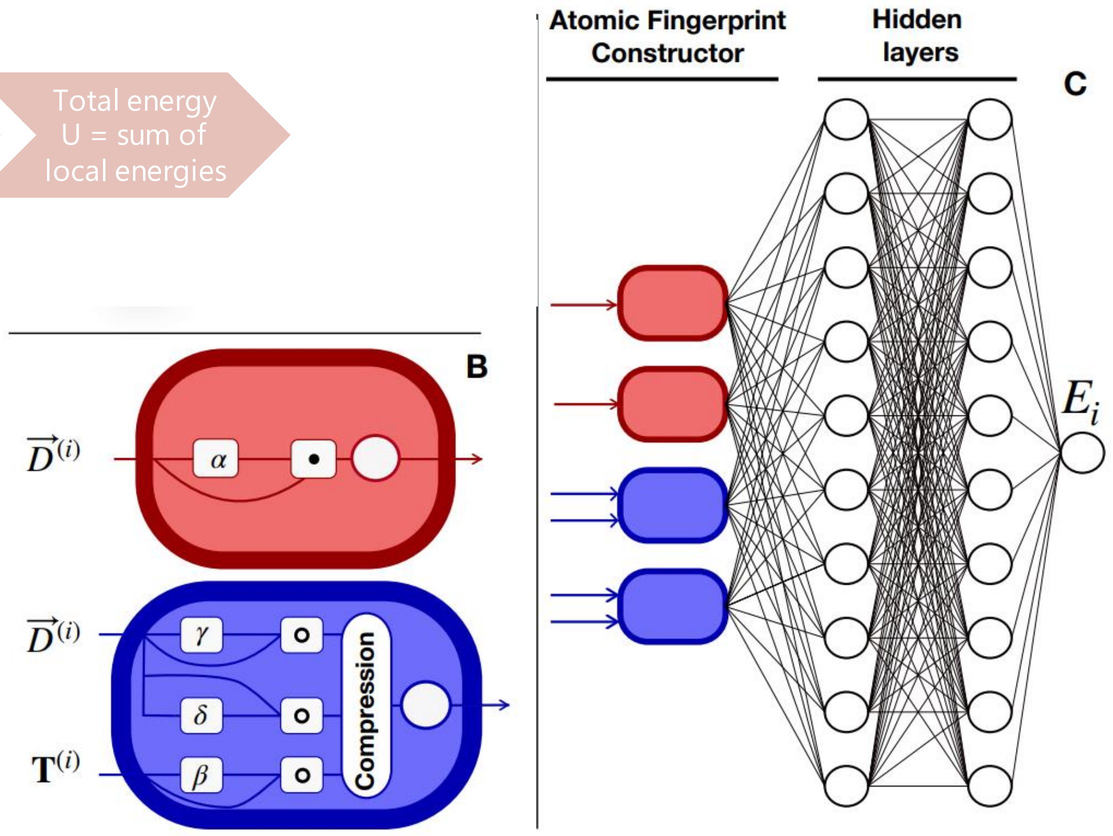
2-body and 3-body descriptors with **learnable** exponential weights.

$$D_j^{(i)}(r_{ij};R_c) = \begin{cases} \frac{1}{2}\left[1+\cos\left(\pi\frac{r_{ij}}{R_c}\right)\right] & r_{ij} \le R_c \\ 0 \quad r_{ij} > R_c \end{cases}$$

$$T_{jk}^{(i)}(r_{ij},r_{ik},\theta_{jik}) = \frac{1}{2}\left[1+\cos\left(\theta_{jik}\right)\right]\ D_j^{(i)}(r_{ij};R_c{}')\ D_k^{(i)}(r_{ik};R_c{}')$$

A MLP with 2 hidden layers of ~25 nodes.

Inspired by the **attention** mechanism in deep learning.



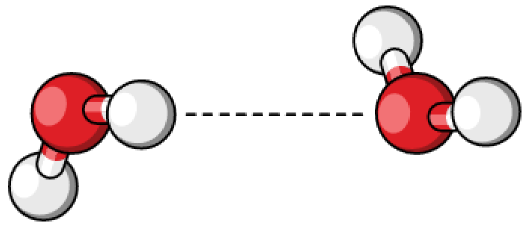
Architecture for a monospecies system

Amazing performance when trained on the "mW" water potential

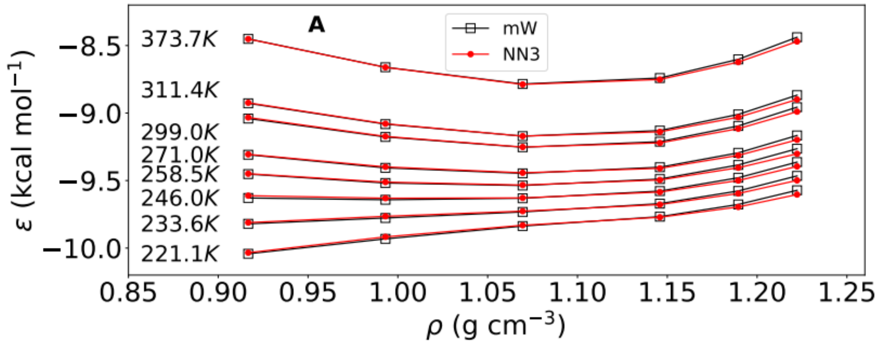
(a classical single-species PES but with non-trivial 3-body interaction)

With only 10 atomic descriptors:

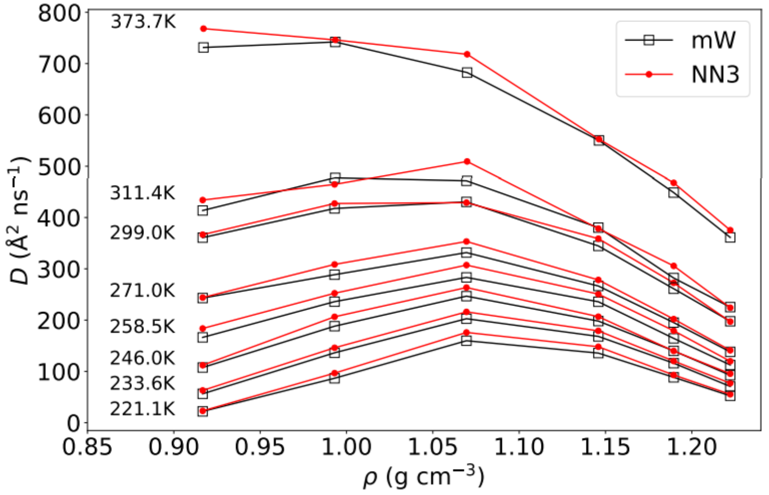
- Energy error ~ 0.4 - 1.3 meV/atom.
- Force error ~ 6.7 meV/Å.

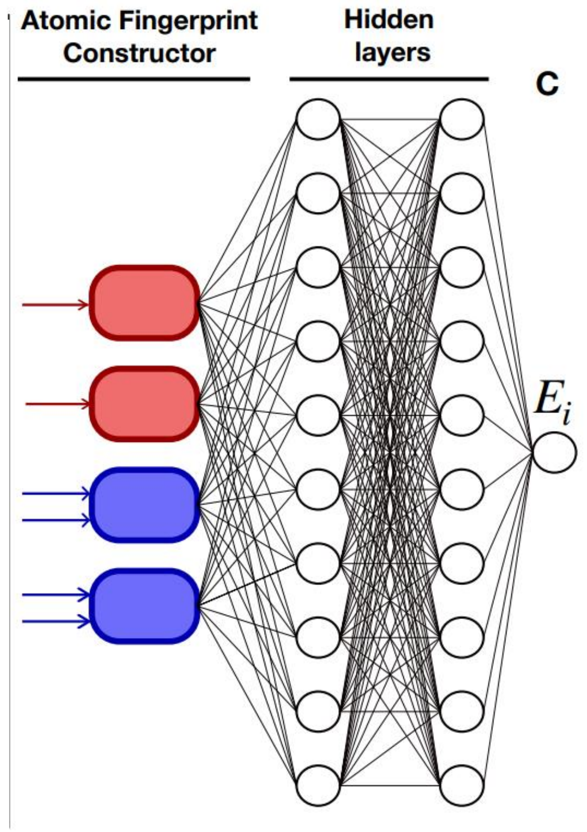


Total energy



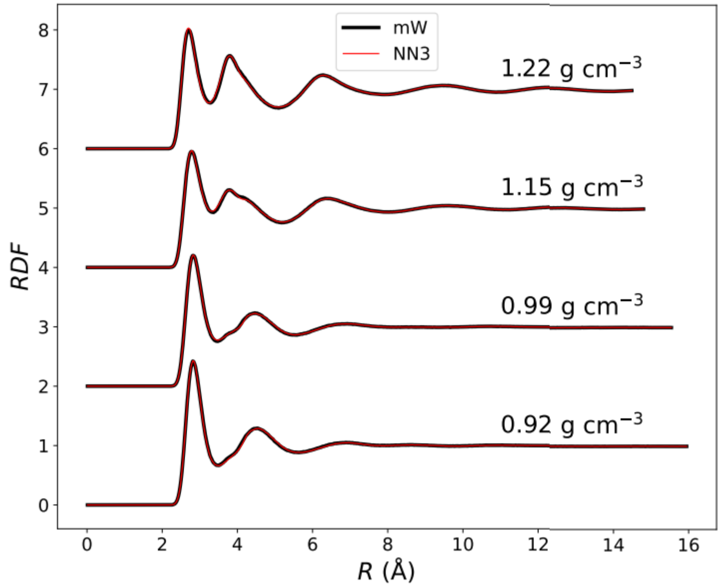
Diffusion coefficient



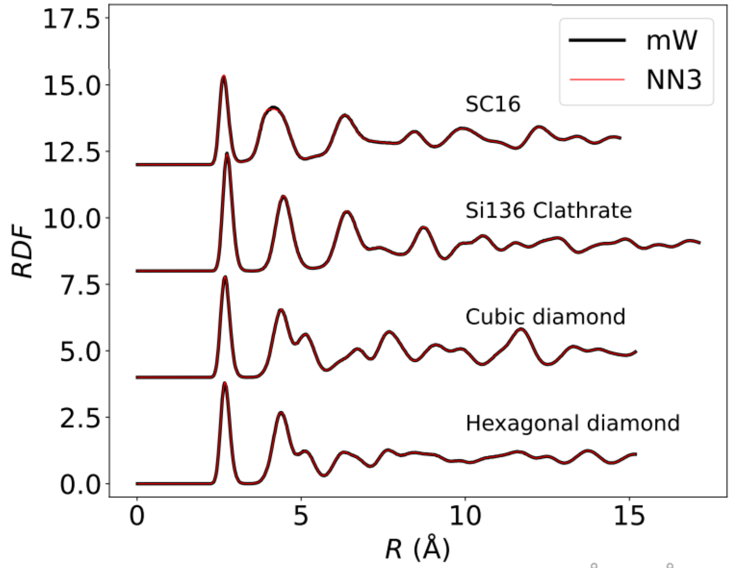


Amazing performance when trained on the “mW” water potential

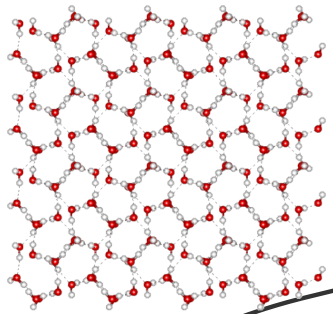
Predicted radial pair correlations perfectly match the real ones:

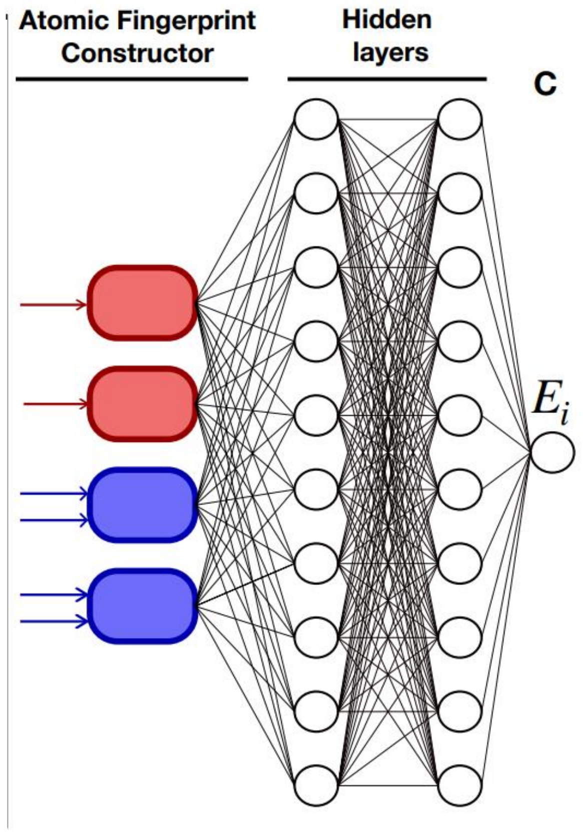


At different densities...



... and even in extrapolated crystal configurations!





Conclusion and Future work

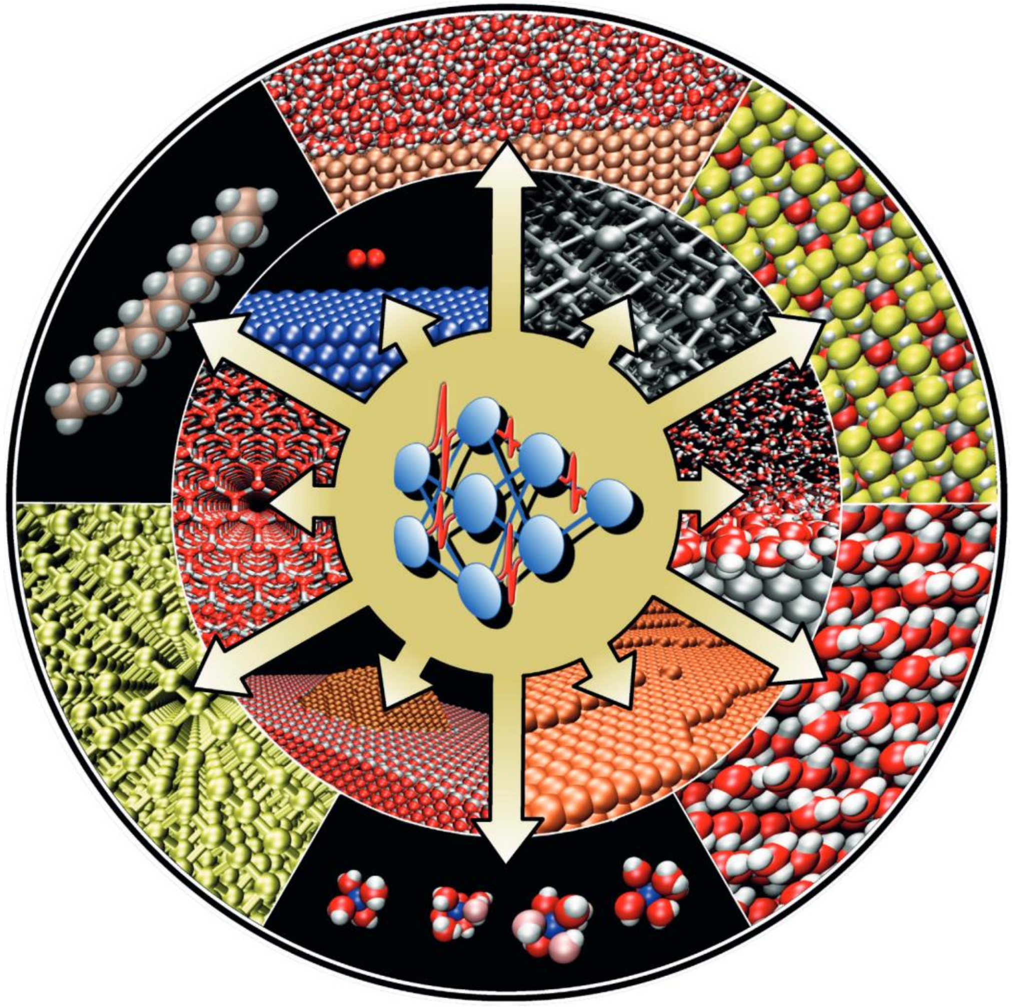
The bottleneck of large-scale quantum-accurate molecular dynamics simulations is the computation of forces.

Neural-network parametrization[1] allows to compute forces both accurately and efficiently.

Part of my project is to apply the very recent method by Mattioli et al. [4] to some *Phase-Change Materials* (alloys of Ge, Sb, Te) for large-scale simulations in the supercooled phase.

[1] And other machine-learning methods which I did not cover today.

Thanks for your attention!



References

1) Behler, *First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems.* Angew. Chem. Int. Ed. (2017, review).
2) Frenkel, Smit, *Understanding Molecular Simulations.* (1996).
3) Behler and Parrinello, *Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces.* Phys. Rev. Lett. (2007).
4) Mattioli, Sciortino, Russo, *A neural network potential with self-trained atomic fingerprints: a test with the mW water potential.* J. Chem. Phys. (2023).