psiflow
modular and scalable online learning for atomic simulations

Sander Vandenhaute
Prof. Veronique Van Speybroeck
Ghent University (BE)
physical property

IR absorption in H₂
phase diagrams

thermodynamic stability

chemical reactivity

hundreds
thousands
millions/billions of E/f evaluations
scaling
\((N = \text{system size})\)

- \(\exp N\)
- \(N^6\)
- \(N^5\)
- \(N^4\)
- \(N^3\)
- \(N \log N\)

force fields, DFT, hybrid DFT, MP2/RPA, CCSD, full CI

the accuracy we want
the scaling we can afford
Use equivariant neural networks to learn $E/\phi$

**STEP 1**

atomic environment

$r_{\text{cut}}$

feature representation

use ‘equivariant’ features

**STEP 2**

equivariant neural network

energy

should remain invariant under translations or rotations!

much less training data! (~100)

Tensor Field Networks: arXiv.1802.08219v2 [cs.LG]

Clebsch-Gordan Nets: arXiv.1806.09231v2 [stat.ML]
while error(model) is high:

data = generate_atomic_data(model)  # CPU/
data = evaluate_DFT_energy_forces(data)  # C
model.train(data)  # GPU
average computational chemist is not very computational
- hide Parsl API as much as possible

enforce “write once, run anywhere”
- separate configuration from high-level workflow definition
WHAT?
high-level workflow definition
using abstractions:
   Model
   Dataset
   Walker
   Learning
...

HOW?
run CP2K on $N_{cores}$
train model for $M$ minutes
   Docker/Apptainer URIs
   MPI/OpenMP

WHERE?
Parsl providers!

run.py
lumi.py
frontier.py

python run.py  # runs locally using ThreadPoolExecutors
python run.py --psiflow-config lumi.py  # on LUMI
import psiflow
from psiflow.data import Dataset  # wraps around File/DataFuture
from psiflow.models import MACEModel  # wraps around File/DataFuture

psiflow.load()  # build and load Parsl config; ‘compile’ apps

train, valid = Dataset.load('dft.xyz').split(0.9)
model = MACEModel()  # contains DataFuture of untrained model
model.train(train, valid)  # contains DataFuture of trained model

errors = Dataset.get_errors(valid, model.evaluate(valid))  # Future
errors.result()  # NumPy array of validation errors on energy/force
bundled in a Docker/Apptainer image  ▸ ContainerizedLauncher
Big workflows create huge amounts of files
- automatic tarring? Or even archived by default?

Big workflows require a lot of memory
- more extensive use of virtual, on-disk memory?

Debugging Parsl workflows is nontrivial
- lazy failure not always best option?
Veronique Van Speybroeck
Massimo Bocus
Tom Braeckevelt
Pieter Dobbelraere
& others