XACS
Xiamen Atomistic Computing Suite
XACScloud.com

MLatom.com

Active learning

Pavlo O. Dral
Xiamen University, P.R. China

Visiting Professor in
Nicolaus Copernicus University, Poland

5 July 2024

Categories of machine learning potentials

Data ?

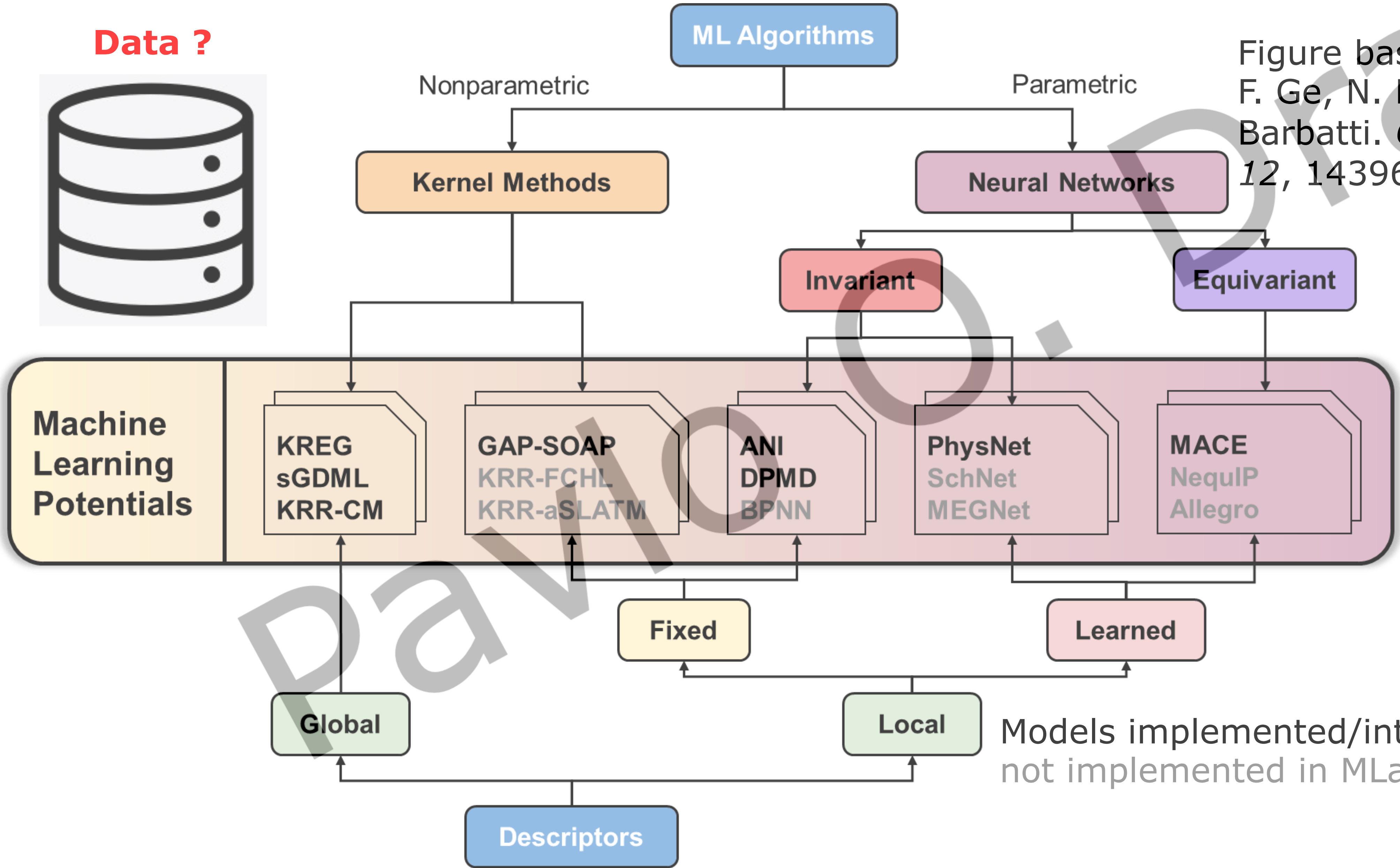


Figure based on M. Pinheiro Jr, F. Ge, N. Ferré, P. O. Dral, M. Barbatti. *Chem. Sci.* **2021**, *12*, 14396–14413

Models implemented/interfaced in MLatom
not implemented in MLatom

Active learning with several NNs

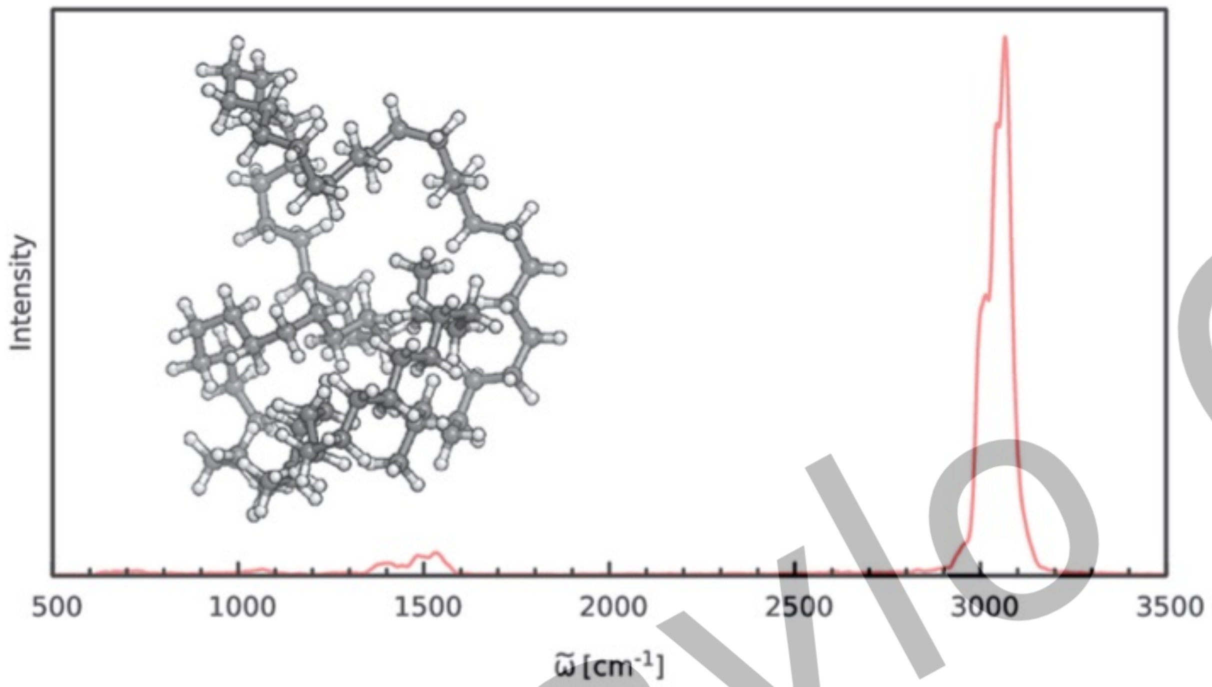
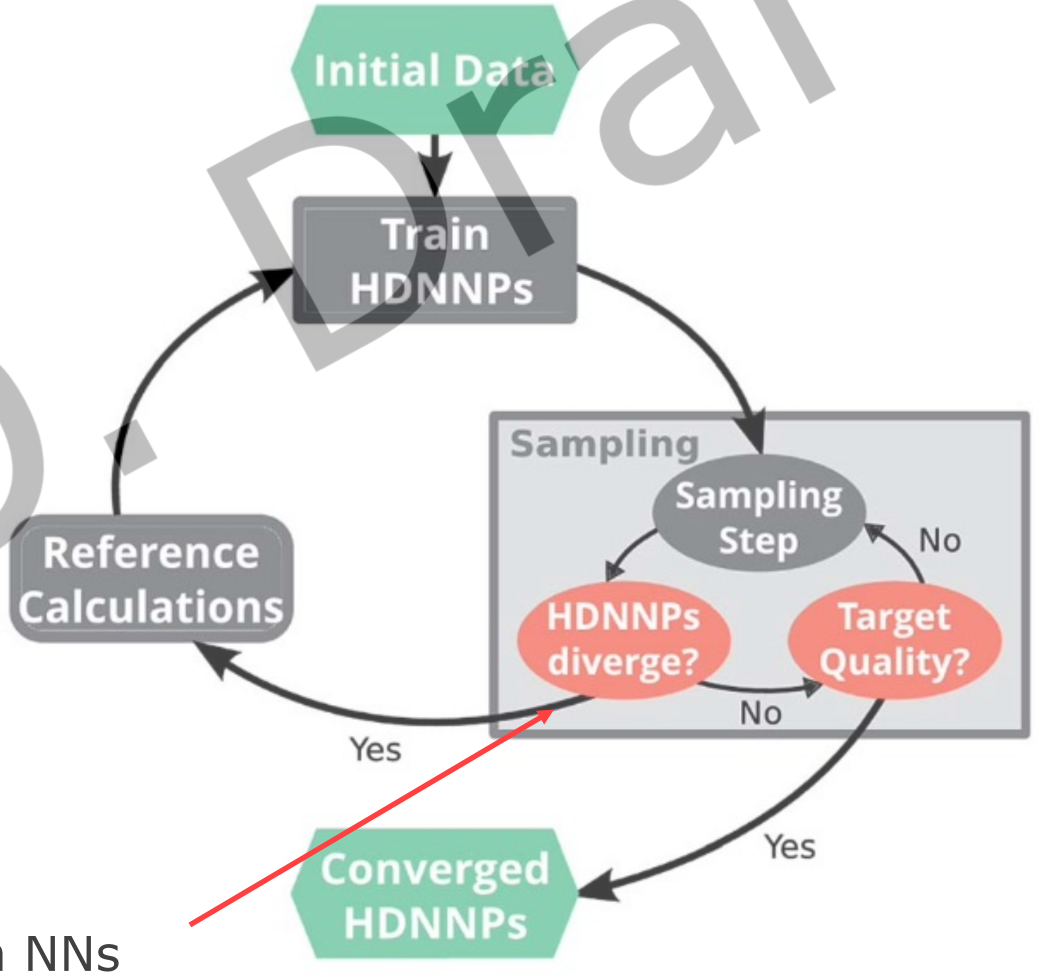
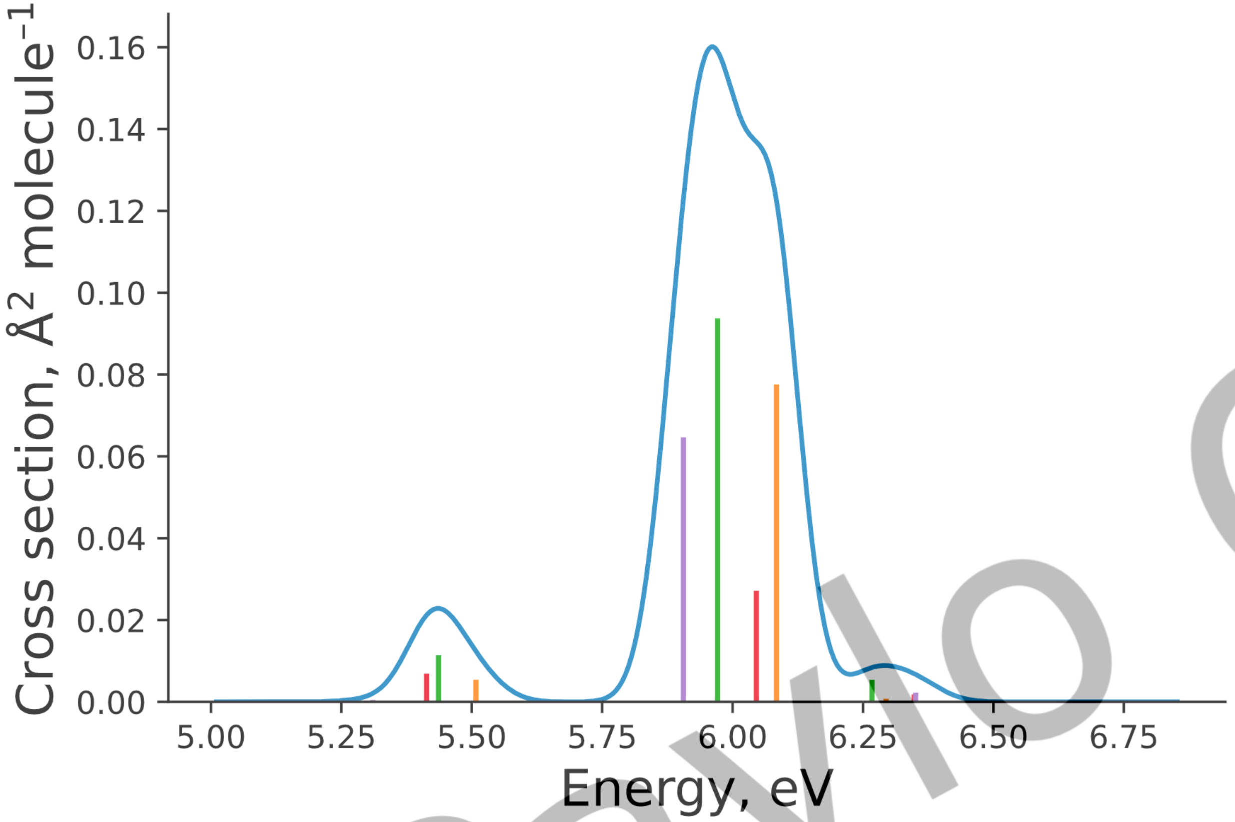


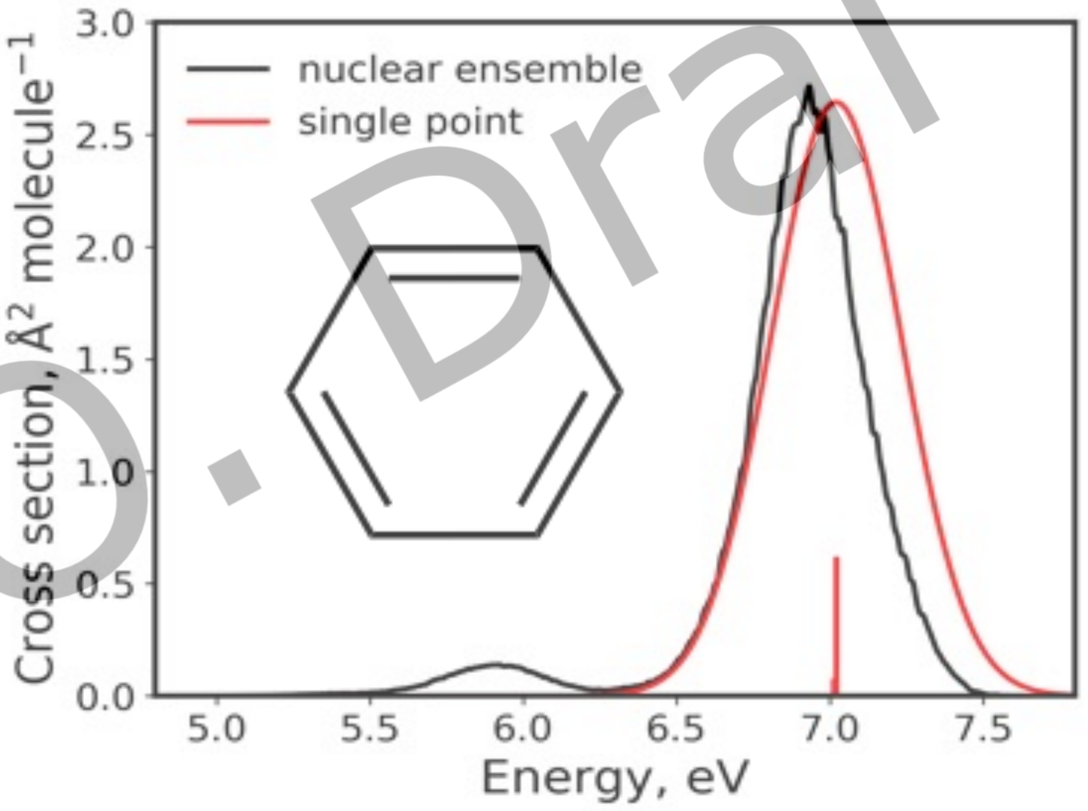
Fig. 6 IR spectrum of the $C_{69}H_{140}$ alkane as predicted by the ML model based on the B2PLYP method.



- Deviation between NNs
- Structural similarity
- Variance...

Machine learning UV/vis absorption spectra





Nuclear Ensemble Approach (NEA) calculates cross section by averaging over multiple normalized broadening functions at different conformations (marked by different colors).

Compared with the single point convolution, NEA correctly predicts absorption intensity for forbidden transitions (example: benzene)

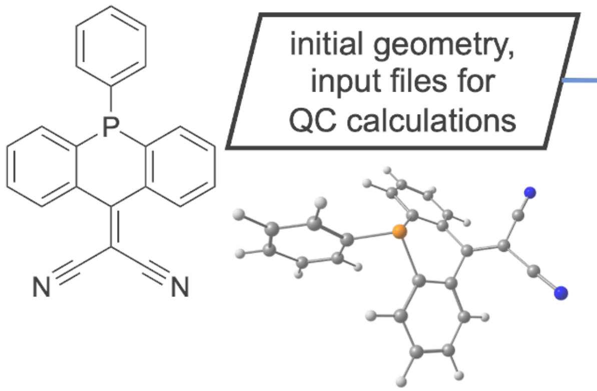
$$\sigma(E) = \frac{\pi e^2 h}{2mc\varepsilon_0 E} \sum_{n}^{N_{fs}} \frac{1}{N_p} \sum_{i}^{N_p} \Delta E_{0n}(\mathbf{x}_i) f_{0n}(\mathbf{x}_i) \frac{1}{\sqrt{2\pi(\delta/2)^2}} \exp\left(-\frac{(E-\Delta E_{0n})^2}{2(\delta/2)^2}\right)$$

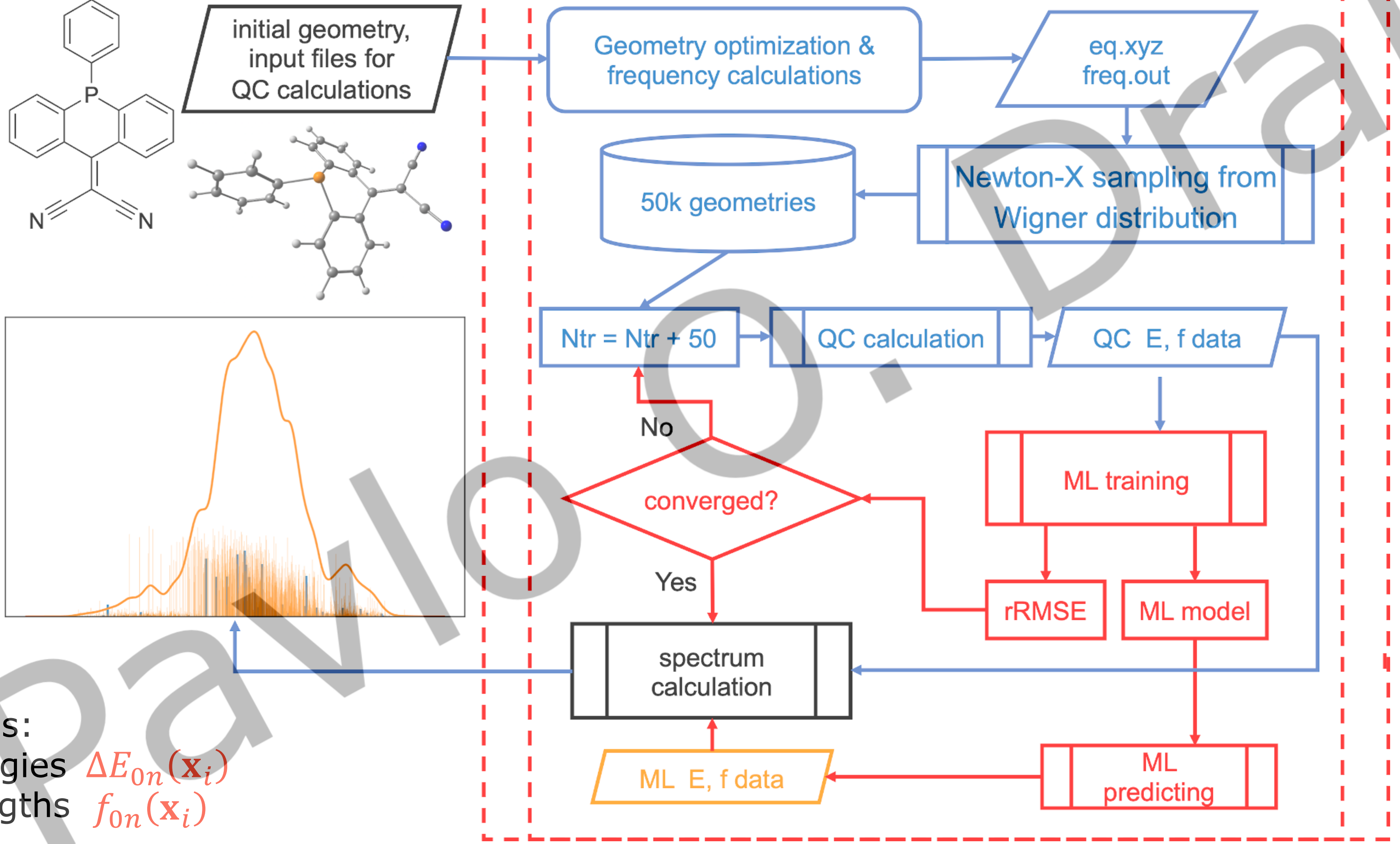
R. Crespo-Otero, M. Barbatti, *Theor. Chem. Acc.* **2012,** *131*, 1237

XACS
Xiamen Atomistic Computing Suite
XACScloud.com

Machine learning UV/vis absorption spectra

MLatom.com





Mario Barbatti

Bao-Xin Xue

Interface:

NX newtonx.org

Learned properties:

- excitation energies $\Delta E_{0n}(\mathbf{x}_i)$
- oscillator strengths $f_{0n}(\mathbf{x}_i)$

ML-NEA method: B.-X. Xue, P. O. Dral, M. Barbatti, *J. Phys. Chem. A* **2020**, *124*, 7199–7210
Implementation in MLatom: P. O. Dral, F. Ge, B.-X. Xue, Y.-F. Hou, M. Pinheiro Jr, J. Huang, M. Barbatti, *Top. Curr. Chem.*, ***2021***, *379*, 27

The KREG model

$$f(\mathbf{x}_i) = \sum_{j=1}^{N_{\text{tr}}} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\sum_s^{N_x}(x_{i,s} - x_{j,s})^2\right)$$

the Gaussian kernel function
(σ is the kernel width)

$$x = \left(\ldots \quad \frac{R^{eq}}{R} \quad \ldots\right)^T$$

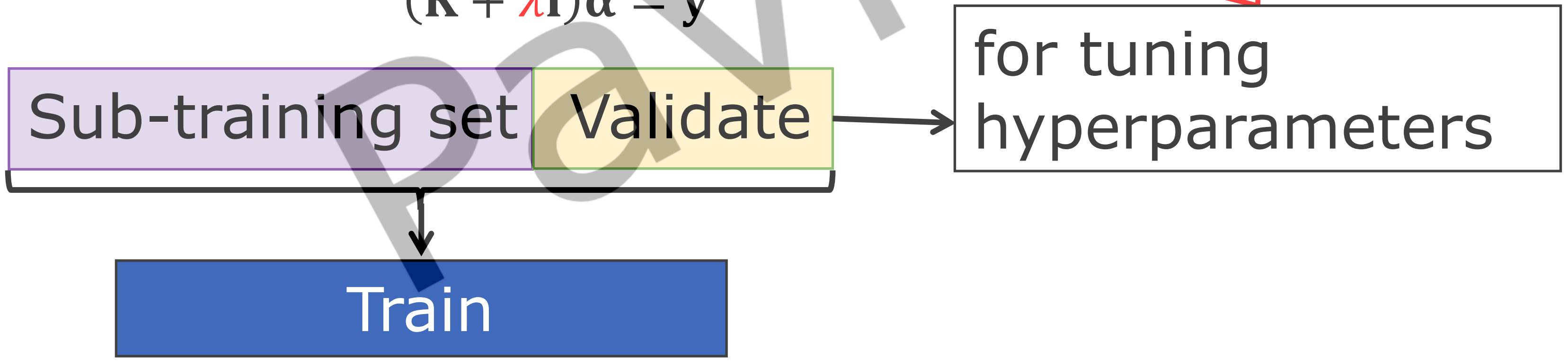
RE descriptor

Analytical solution for the regression coefficients α given N_{tr} training points

$$\begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) + \lambda & \cdots & k\left(\mathbf{x}_1, \mathbf{x}_{N_{\text{tr}}}\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathbf{x}_{N_{\text{tr}}}, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_{N_{\text{tr}}}, \mathbf{x}_{N_{\text{tr}}}\right) + \lambda \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{N_{\text{tr}}} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{N_{\text{tr}}} \end{pmatrix}$$

λ is the regularization parameter ensuring transferability

$$(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$$

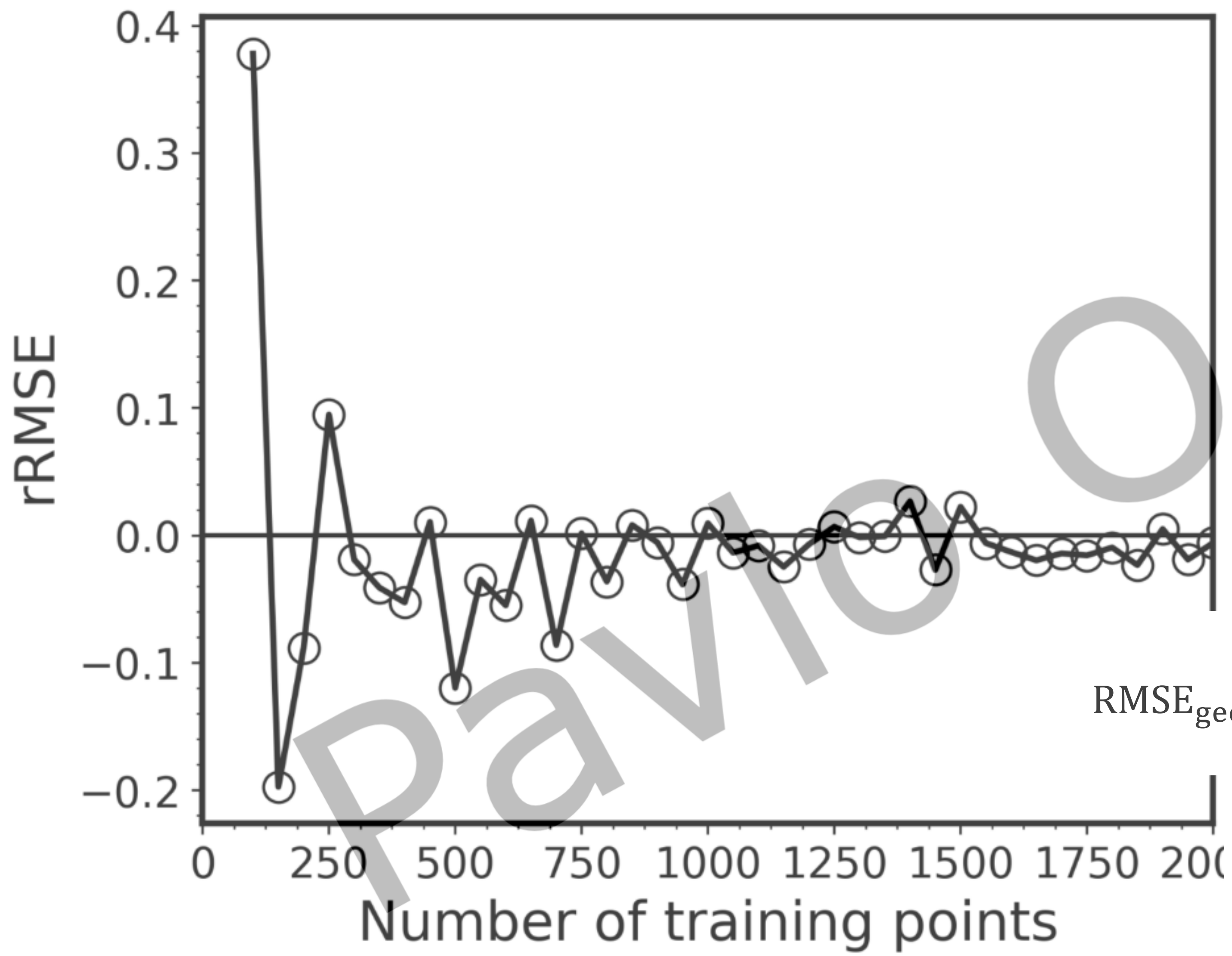


The **KREG** model (**K**ernel ridge regression [KRR] with **RE** descriptor and the **G**aussian kernel function; RE descriptor stands for Internuclear distances **R**elative to **E**quilibrium) to complete all the ML tasks.

XACS
Xiamen Atomistic Computing Suite
XACScloud.com

Automatic determination of training points

MLatom.com



Optimal number of training points can be determined automatically using iterative procedure that stops after the relative change in ML validation errors drops below the threshold (typically rRMSE < 0.1).

$$\mathrm{rRMSE} = \frac{RMSE_{geom}(N_{tr}) - RMSE_{geom}(N_{tr} - N_{step})}{RMSE_{geom}(N_{tr})}$$

$$\mathrm{RMSE_{geom}}(N_{tr}) = \sqrt[2N_S]{\prod_{i=1}^{N_S} \mathrm{RMSE}_{\Delta E_{0n}}(N_{tr}) \cdot \mathrm{RMSE}_{f_{0n}}(N_{tr})}$$

Example ML-spectra-3.

The following task is from our book chapter.

MLatom input file:

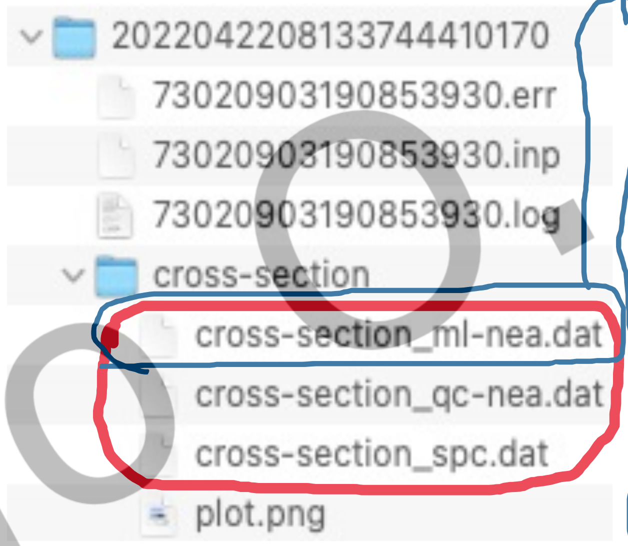
```
cross-section
Nexcitations=30
plotQCNEA
plotQCSPC
deltaQCNEA=0.05
```

These calculations require many data files (reference excitation energies at TDDFT level). These data files are zipped. You should unzip them and upload all of them as auxiliary files to the cloud.

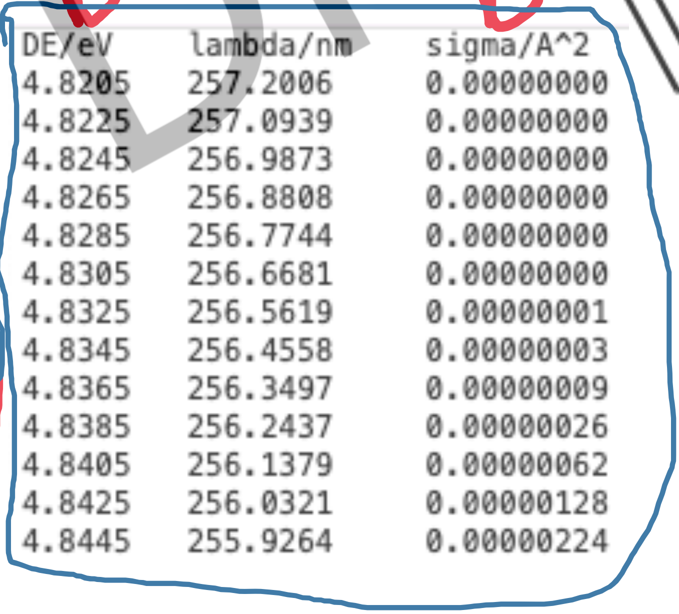
Create input file and submit the job in the File Manager.

Calculations can take more than 5 min. MLatom automatically determines the minimum required number of training points, in this case it needed 200 points for precise spectrum. In the output file you can find that it took 4 iterations to converge:

```
==========================================================================================
run ML-NEA iteratively for spectrum generation ( ML_train_iter ) started at Wed Dec  1 12:00:19 2021 CST
ML-NEA iteration 1: train_number = 50; RMSE_geom = 0.06717941145022376; rRMSE = 1.0

ML-NEA iteration 2: train_number = 100; RMSE_geom = 0.09043318436728051; rRMSE = 0.25713761026721255

ML-NEA iteration 3: train_number = 150; RMSE_geom = 0.06411060145373663; rRMSE = 0.410580813729204

ML-NEA iteration 4: train_number = 200; RMSE_geom = 0.06957370457176553; rRMSE = 0.07852252732055763

ML-NEA iteration ended after 4 iteration!
run ML-NEA iteratively for spectrum generation ( ML_train_iter ) finished at Wed Dec  1 12:08:01 2021 CST |||| total spent 462.02 sec
==========================================================================================
```



```
2022042208133744410170
    73020903190853930.err
    73020903190853930.inp
    73020903190853930.log
  cross-section
    cross-section_ml-nea.dat
    cross-section_qc-nea.dat
    cross-section_spc.dat
    plot.png
```



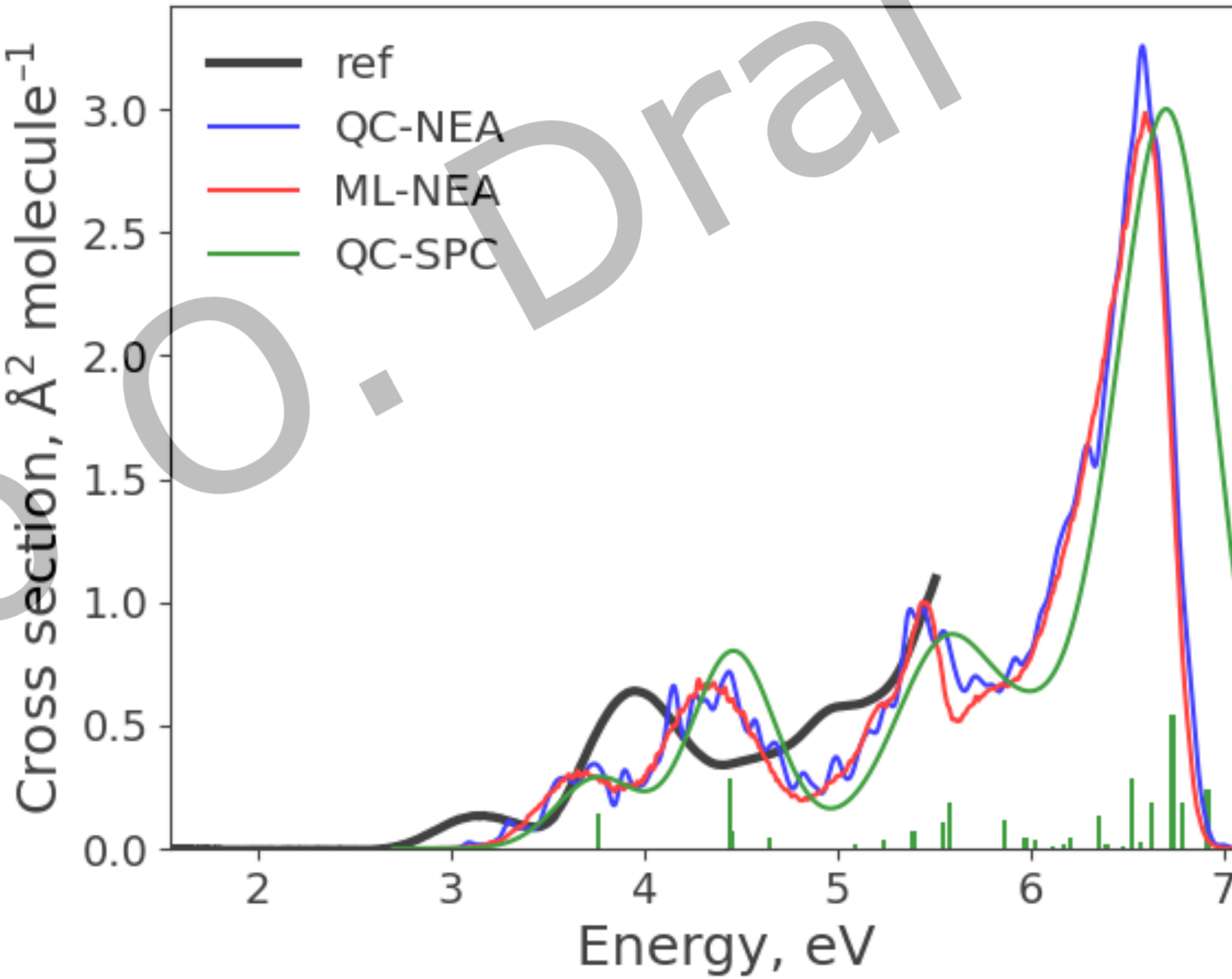
```
DE/eV     lambda/nm    sigma/A^2
4.8205    257.2006     0.00000000
4.8225    257.0939     0.00000000
4.8245    256.9873     0.00000000
4.8265    256.8808     0.00000000
4.8285    256.7744     0.00000000
4.8305    256.6681     0.00000000
4.8325    256.5619     0.00000001
4.8345    256.4558     0.00000003
4.8365    256.3497     0.00000009
4.8385    256.2437     0.00000026
4.8405    256.1379     0.00000062
4.8425    256.0321     0.00000128
4.8445    255.9264     0.00000224
```

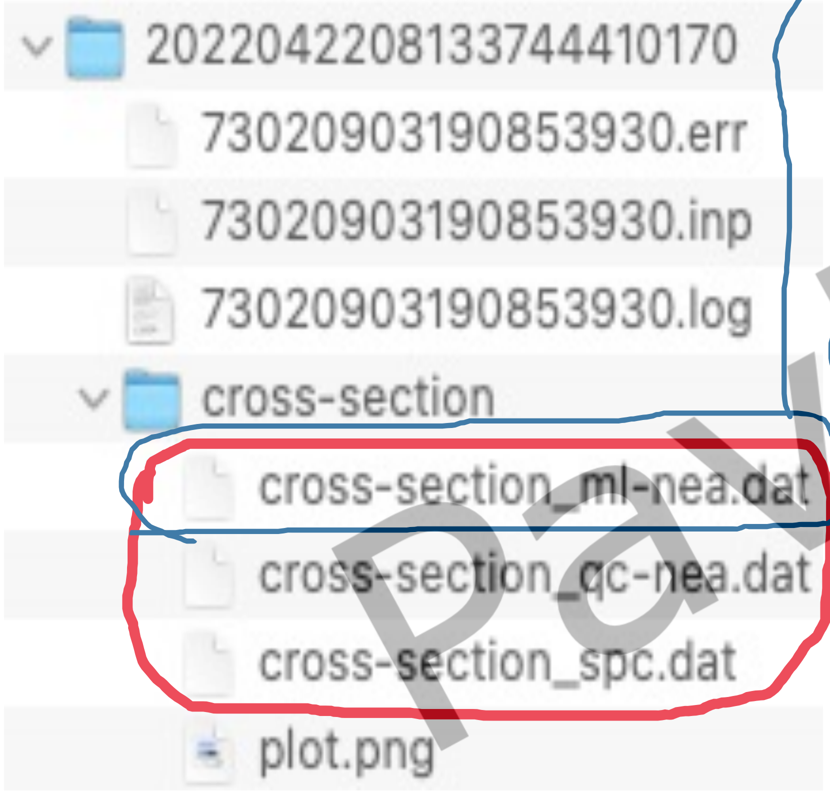
Spectrum plotted for you in cross-section/plot.png

Note: You can also plot raw data from *.dat files with Origin or Excel

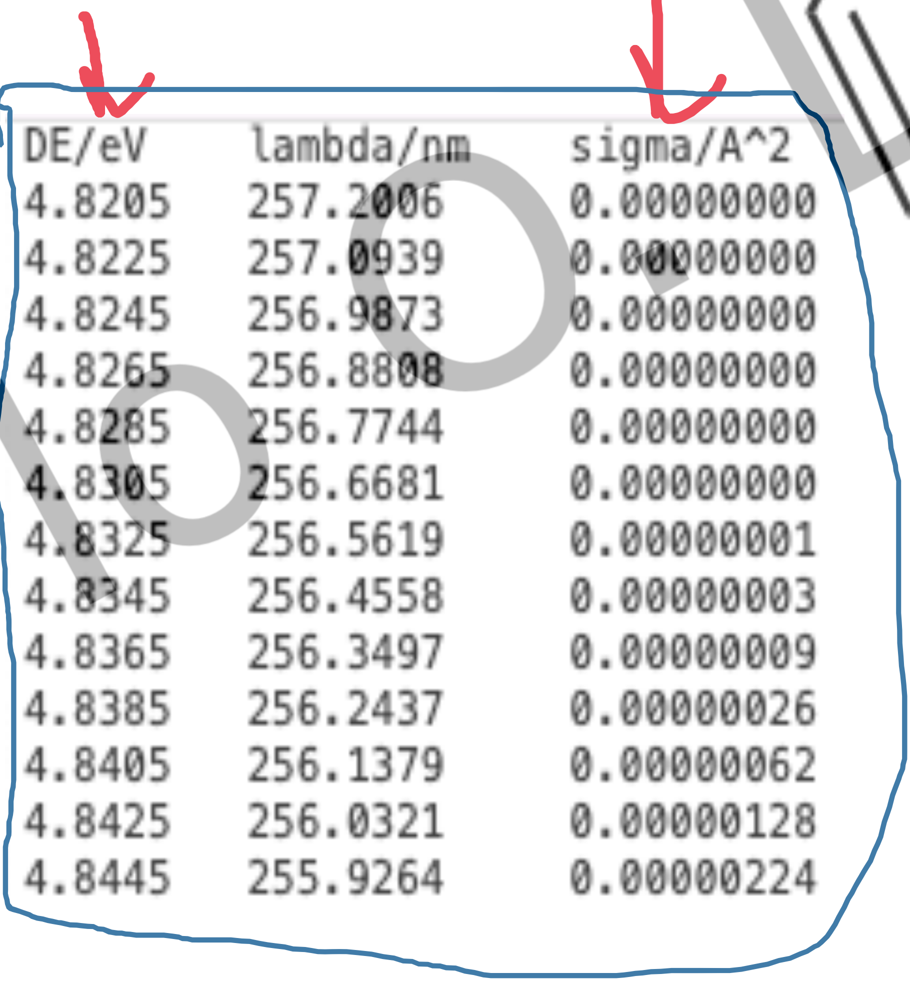
Spectrum plotted for you in cross-section/plot.png

Note: You can also plot raw data from *.dat files with Origin or Excel

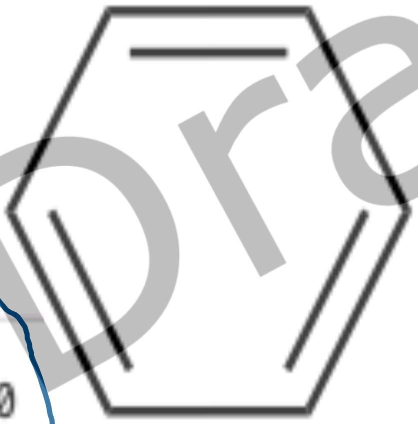




```
v 20220422081337444410170
    730209031908539930.err
    730209031908539930.inp
    730209031908539930.log
  v cross-section
      cross-section_ml-nea.dat
      cross-section_qc-nea.dat
      cross-section_spc.dat
      plot.png
```

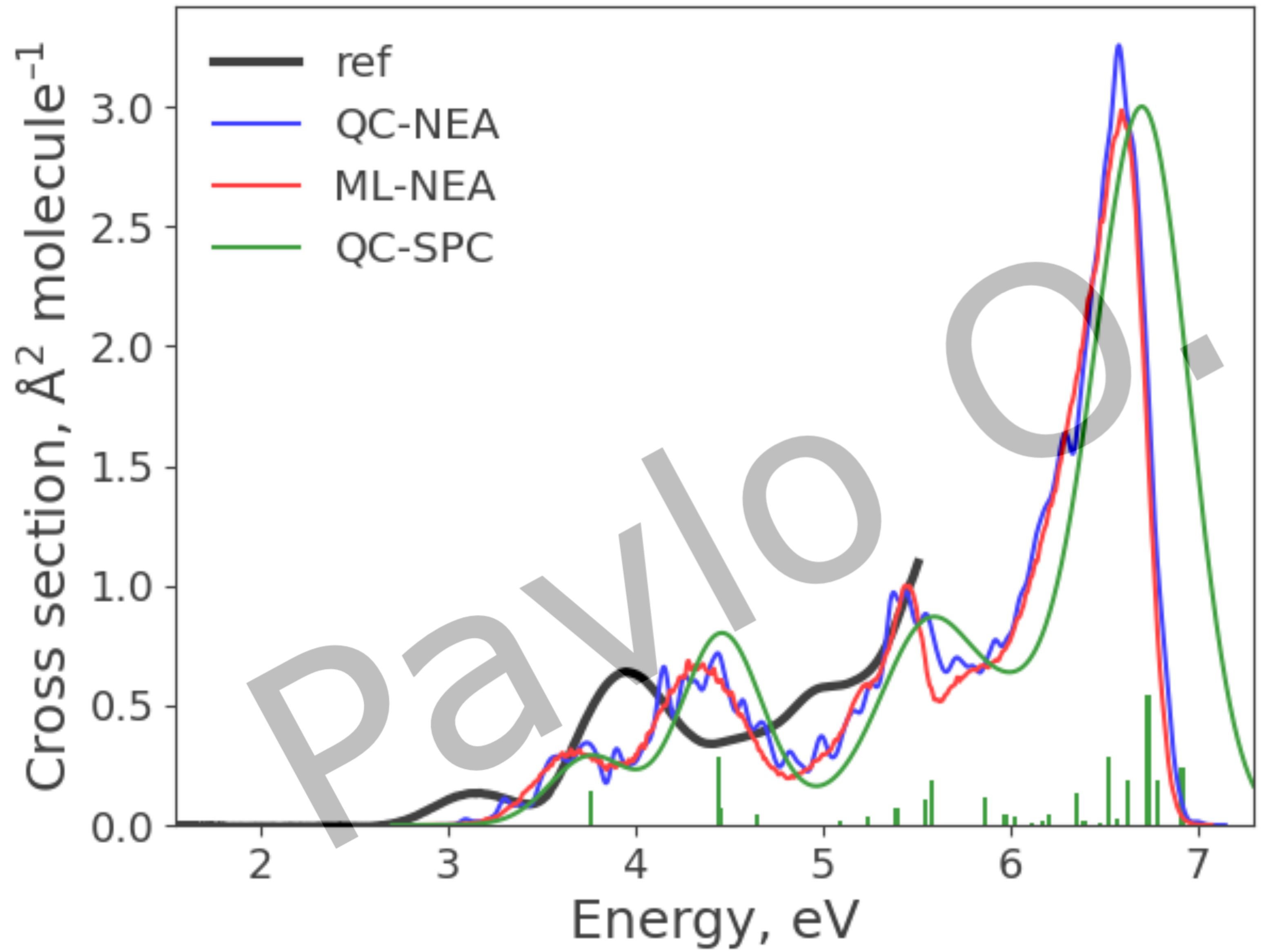


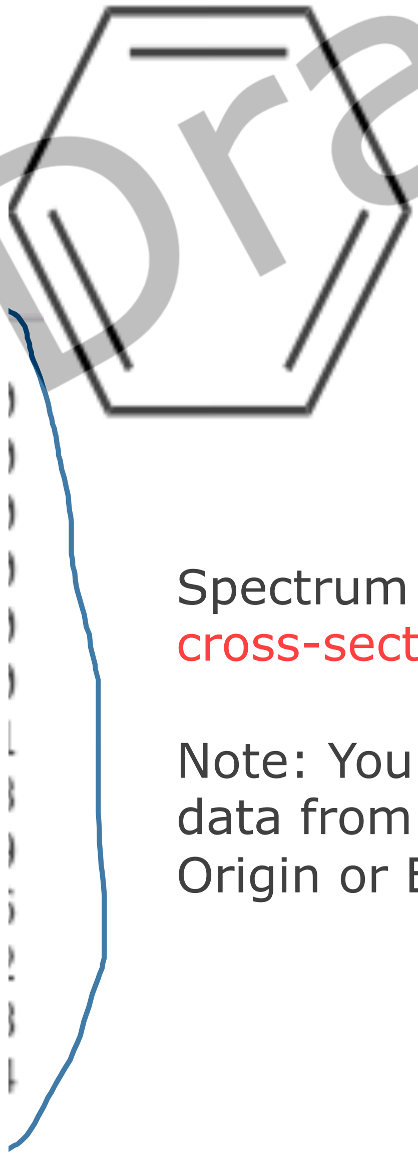
```
DE/eV     lambda/nm     sigma/A^2
4.8205    257.2006      0.00000000
4.8225    257.0939      0.00000000
4.8245    256.9873      0.00000000
4.8265    256.8808      0.00000000
4.8285    256.7744      0.00000000
4.8305    256.6681      0.00000000
4.8325    256.5619      0.00000001
4.8345    256.4558      0.00000003
4.8365    256.3497      0.00000009
4.8385    256.2437      0.00000026
4.8405    256.1379      0.00000062
4.8425    256.0321      0.00000128
4.8445    255.9264      0.00000224
```



Spectrum plotted for you in cross-section/plot.png

Note: You can also plot raw data from *.dat files with Origin or Excel

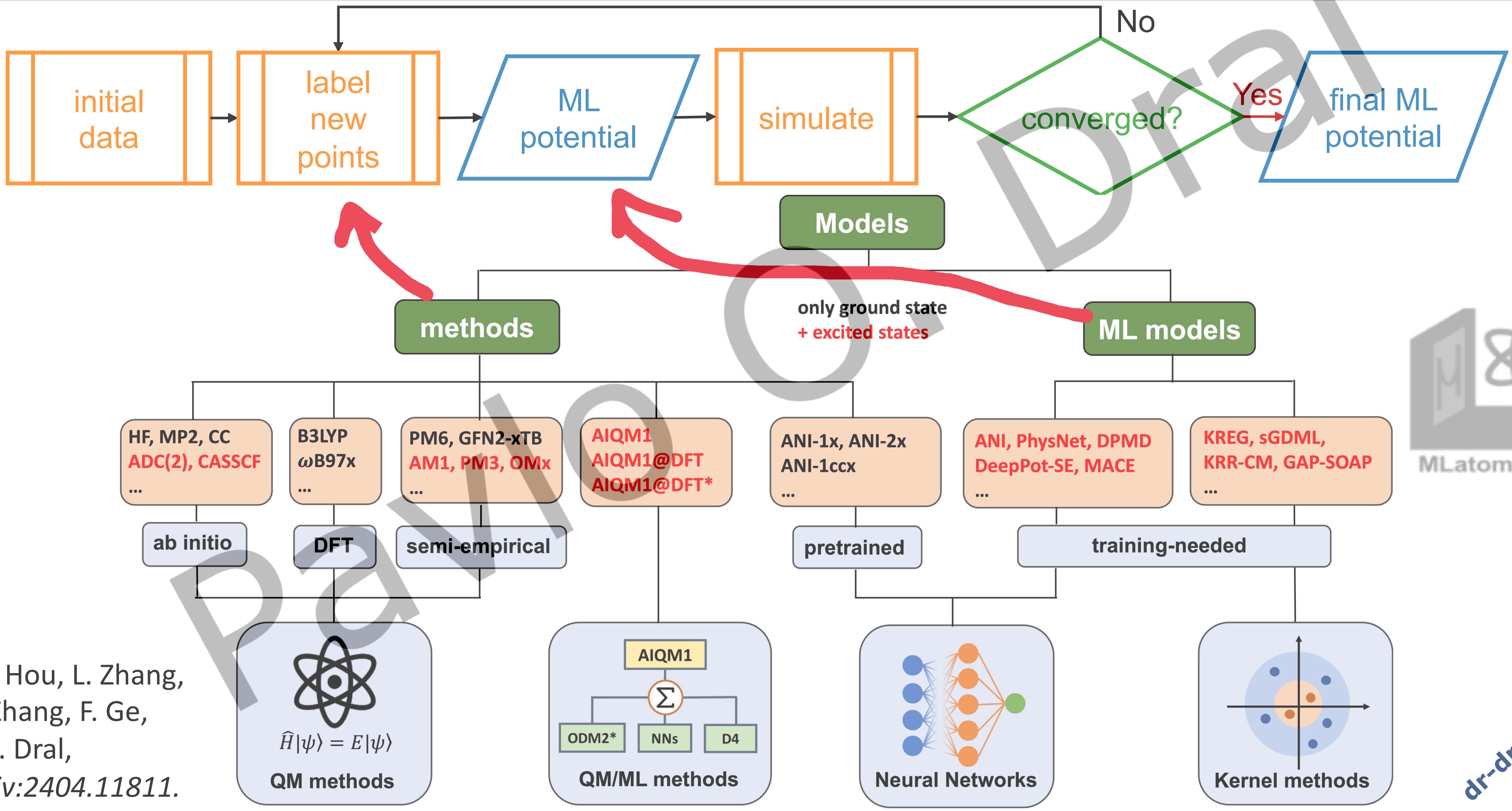




Spectrum plotted for you in cross-section/plot.png

Note: You can also plot raw data from *.dat files with Origin or Excel

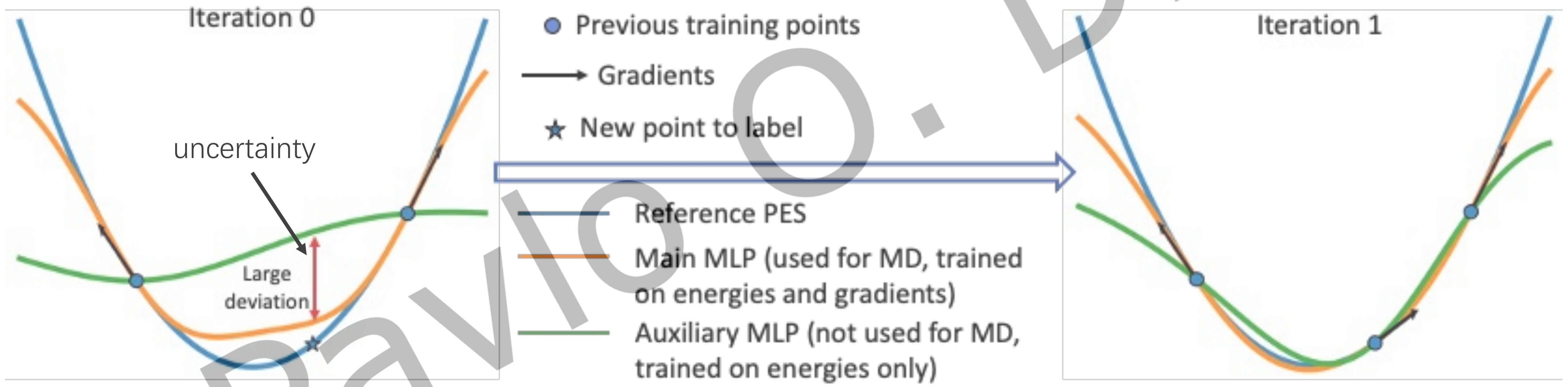
Physics-informed active learning



Y.-F. Hou, L. Zhang, Q. Zhang, F. Ge, P. O. Dral, *arXiv:2404.11811.*

Physics-informed active learning

Utilizing different amount of physics-derived information for uncertainty quantification



Iteration 0
uncertainty
Large deviation
Previous training points
Gradients
New point to label
Reference PES
Main MLP (used for MD, trained on energies and gradients)
Auxiliary MLP (not used for MD, trained on energies only)
Iteration 1

Automatic uncertainty quantification

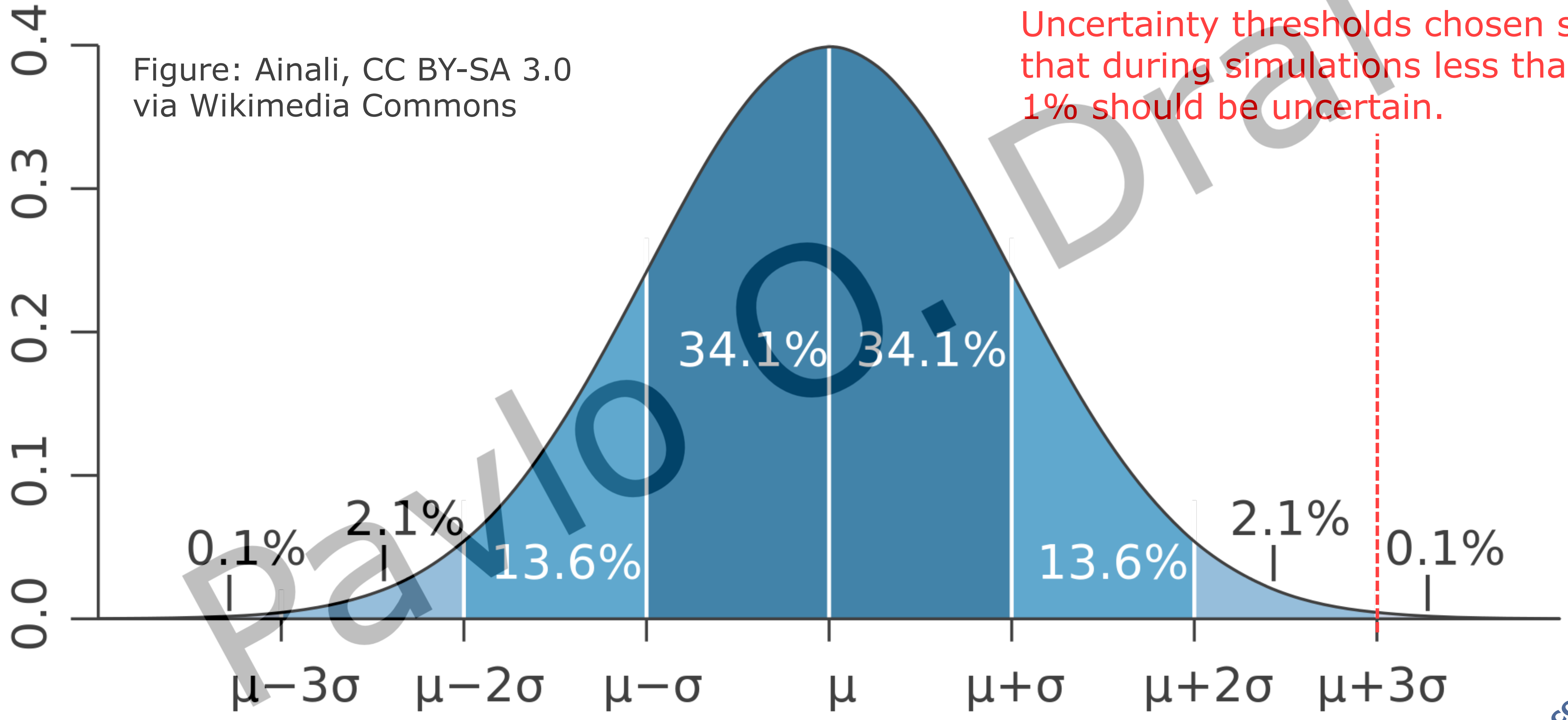
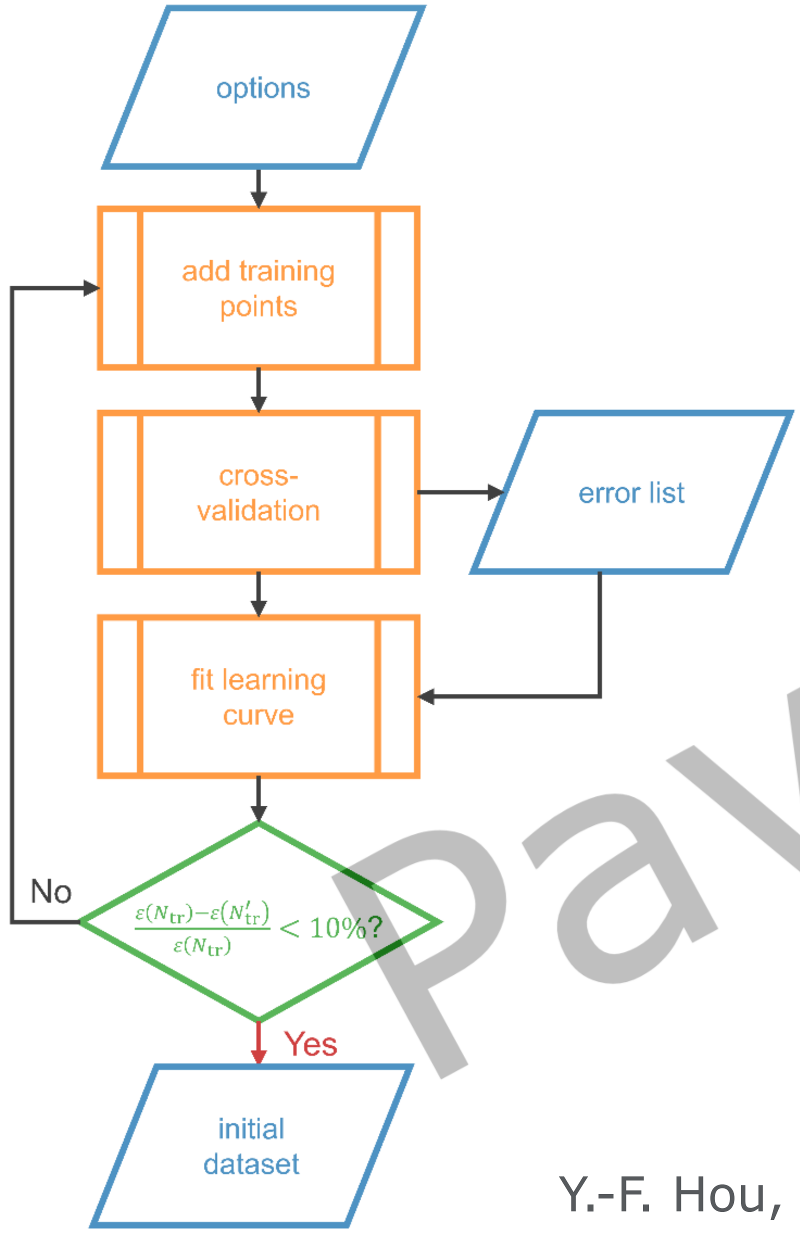
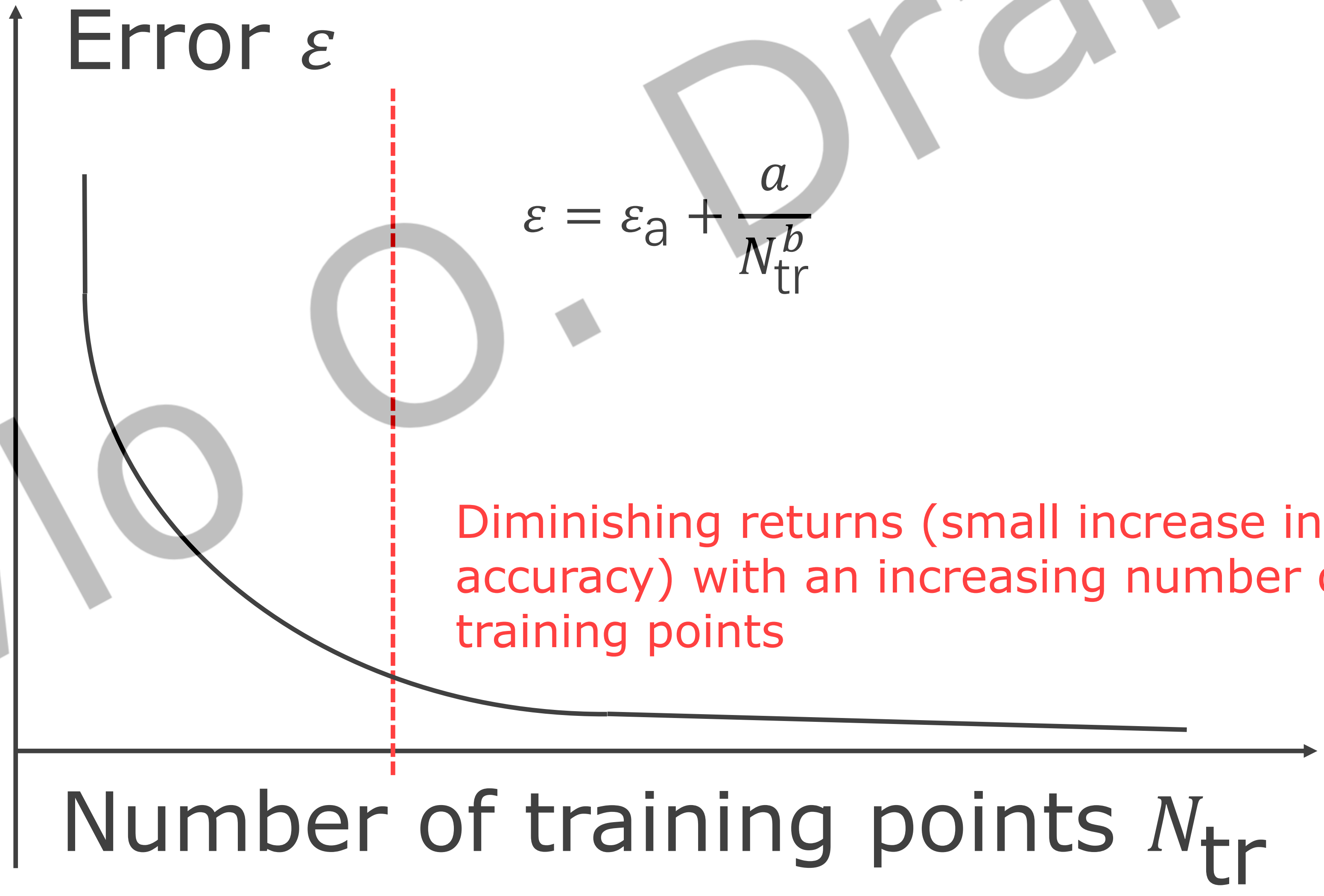


Figure: Ainali, CC BY-SA 3.0 via Wikimedia Commons

Uncertainty thresholds chosen so that during simulations less than 1% should be uncertain.

Automatic initial data building


options
add training points
cross-validation
error list
fit learning curve
$\frac{\varepsilon(N_{\mathrm{tr}})-\varepsilon(N'_{\mathrm{tr}})}{\varepsilon(N_{\mathrm{tr}})} < 10\%?$
No
Yes
initial dataset



$$\varepsilon = \varepsilon_{\mathrm{a}} + \frac{a}{N_{\mathrm{tr}}^{b}}$$

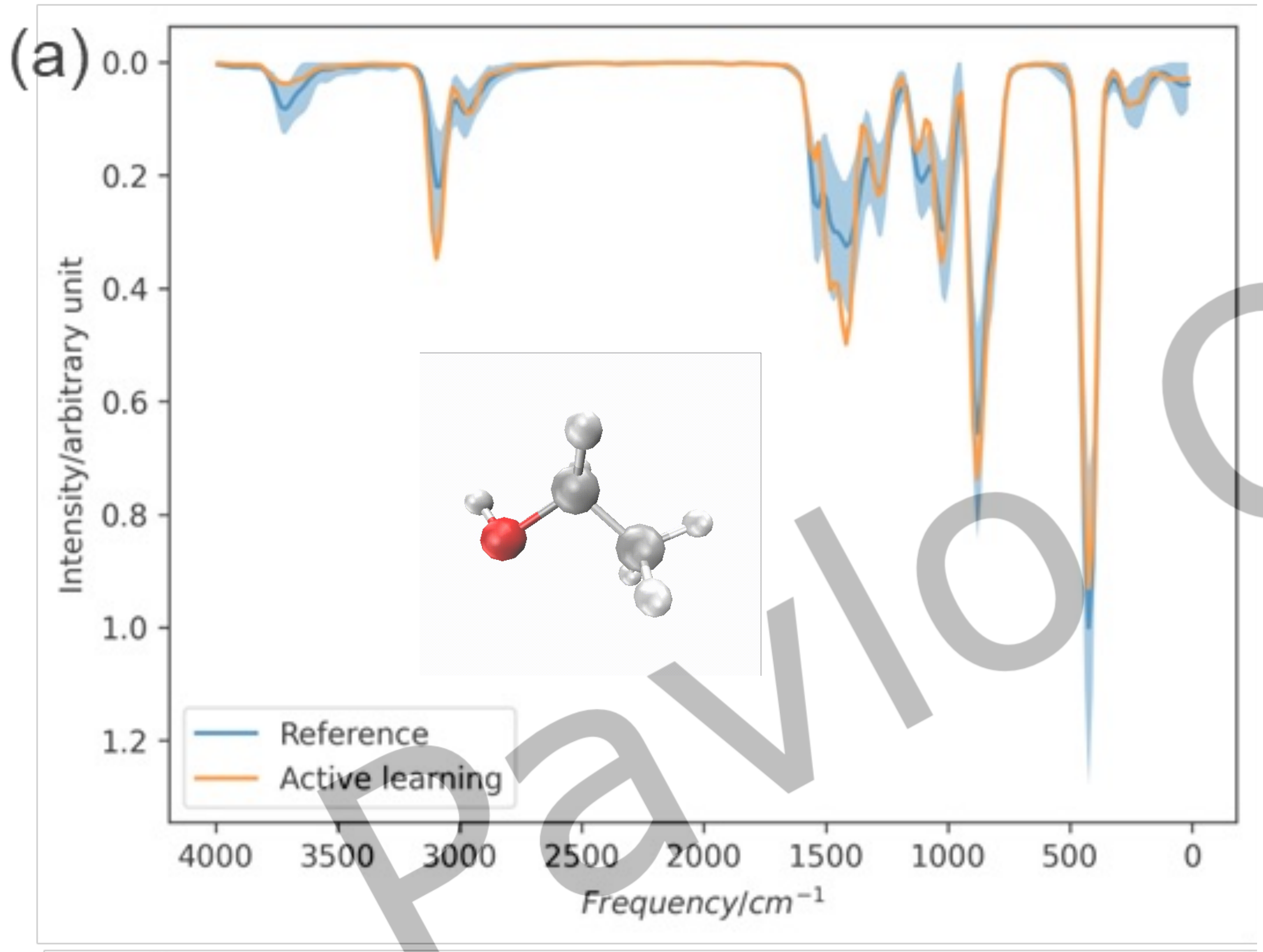
Diminishing returns (small increase in accuracy) with an increasing number of training points

XACS
Xiamen Atomistic Computing Suite
XACScloud.com

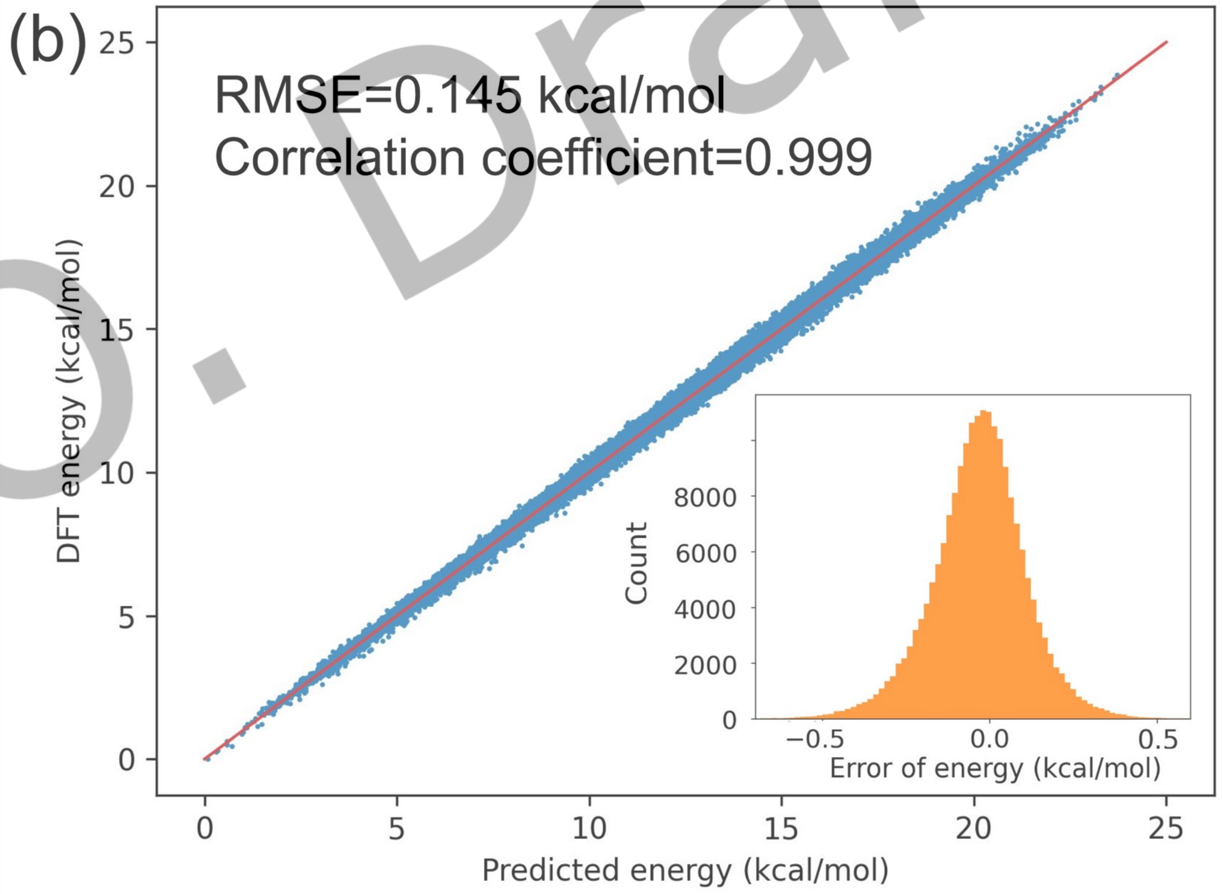
Accurate vibrational spectra

MLatom.com

< 1000 training points with ANI potential



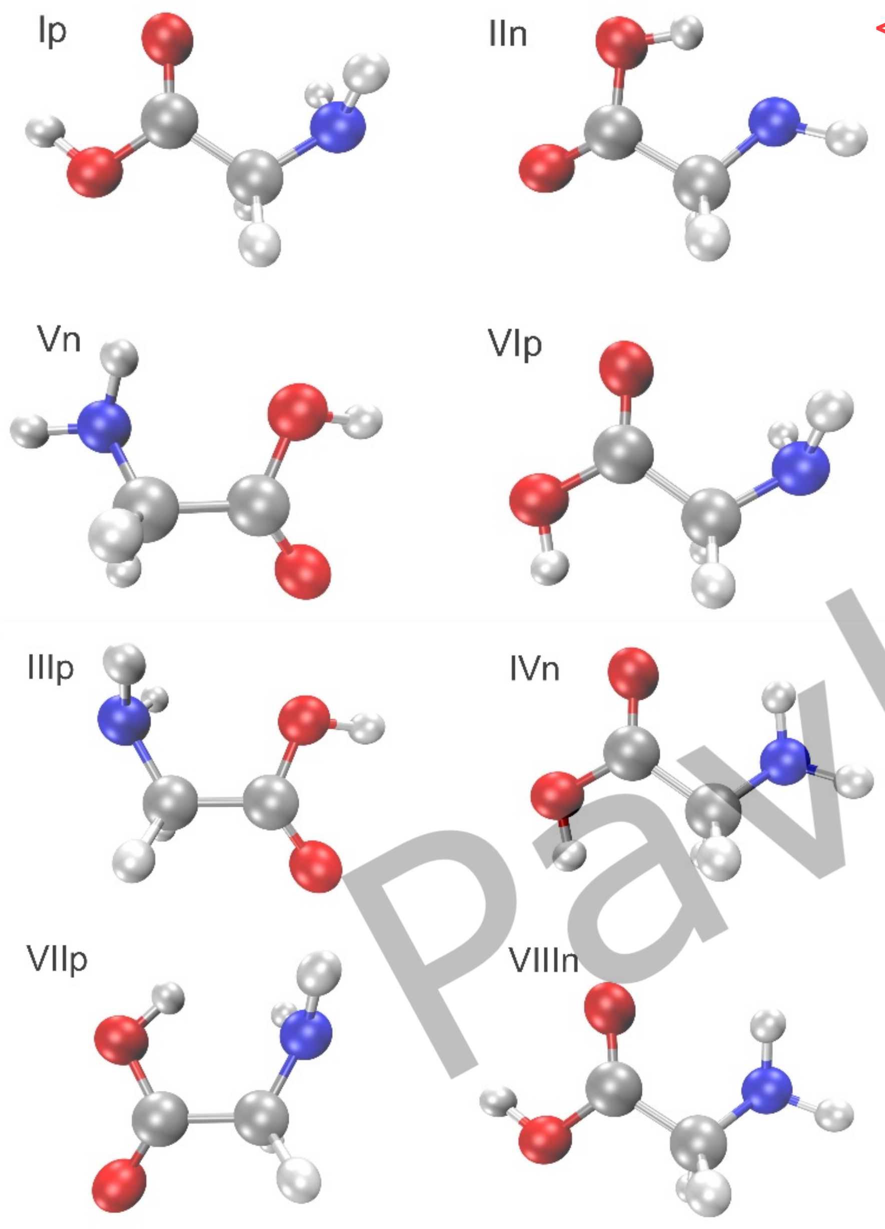
Ethanol spectra



XACS Xiamen Atomistic Computing Suite XACScloud.com

Conformer search through quasi-classical MD

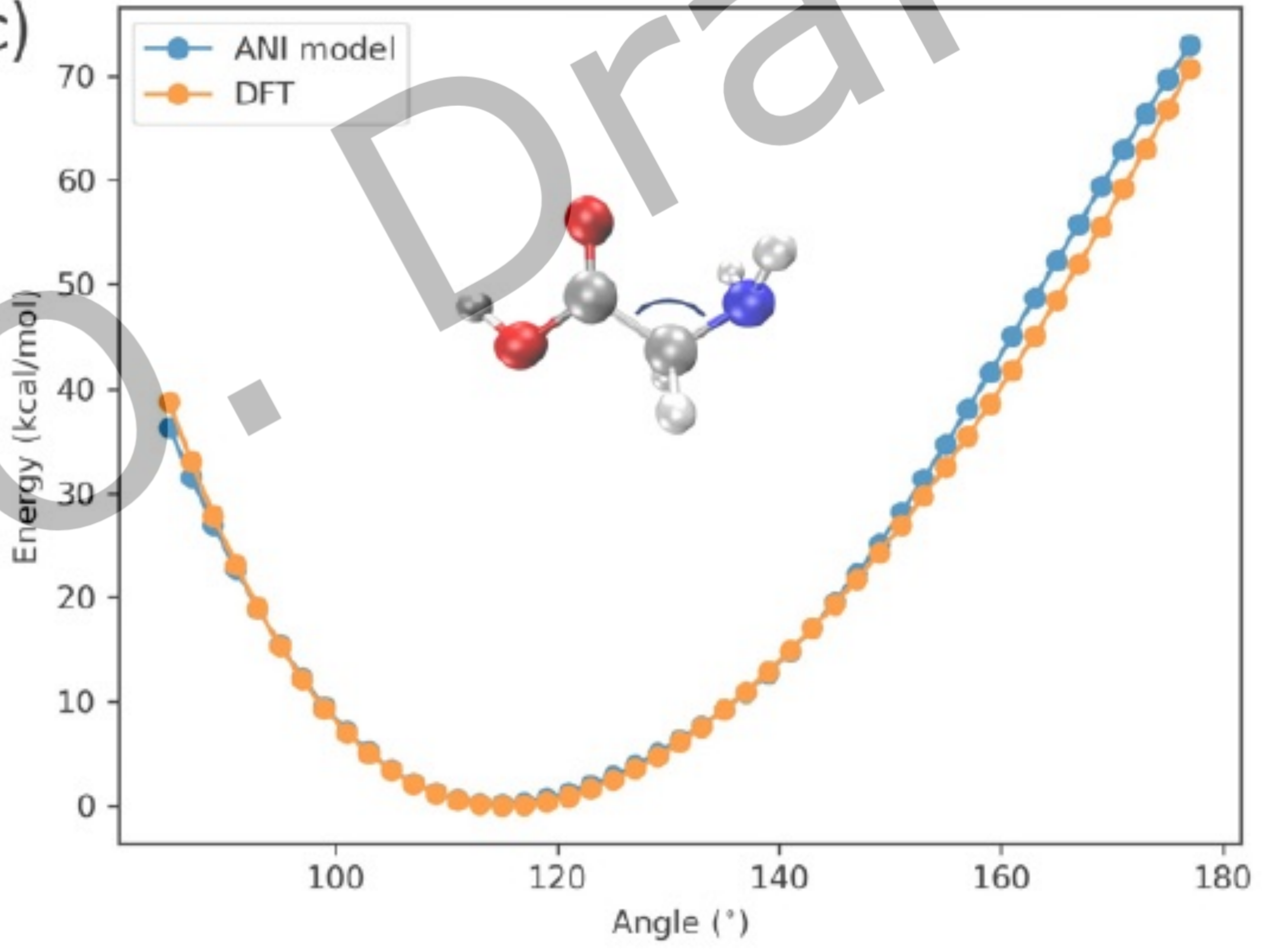
MLatom.com



< 2000 training points with ANI potential

Glycine
8
conformers

(c)

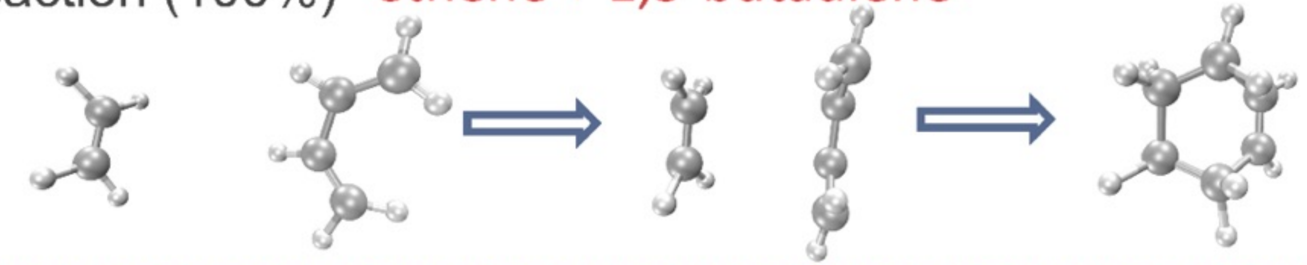


Y.-F. Hou, L. Zhang, Q. Zhang,
F. Ge, P. O. Dral, *arXiv:2404.11811*.

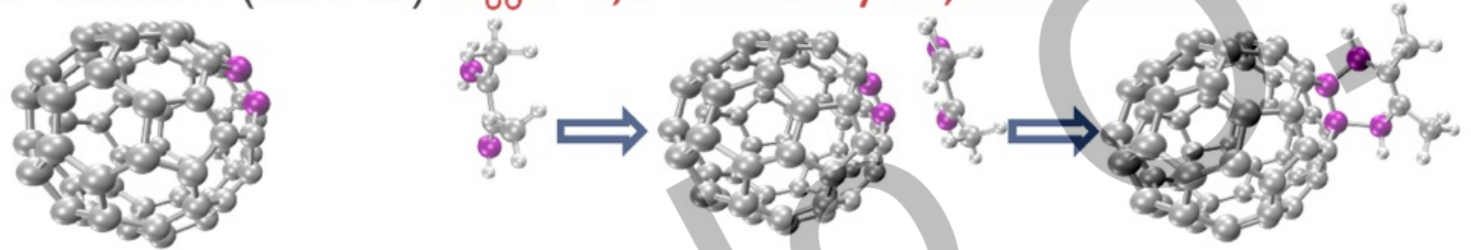
Time-resolved mechanisms

~3000 training points with ANI potential

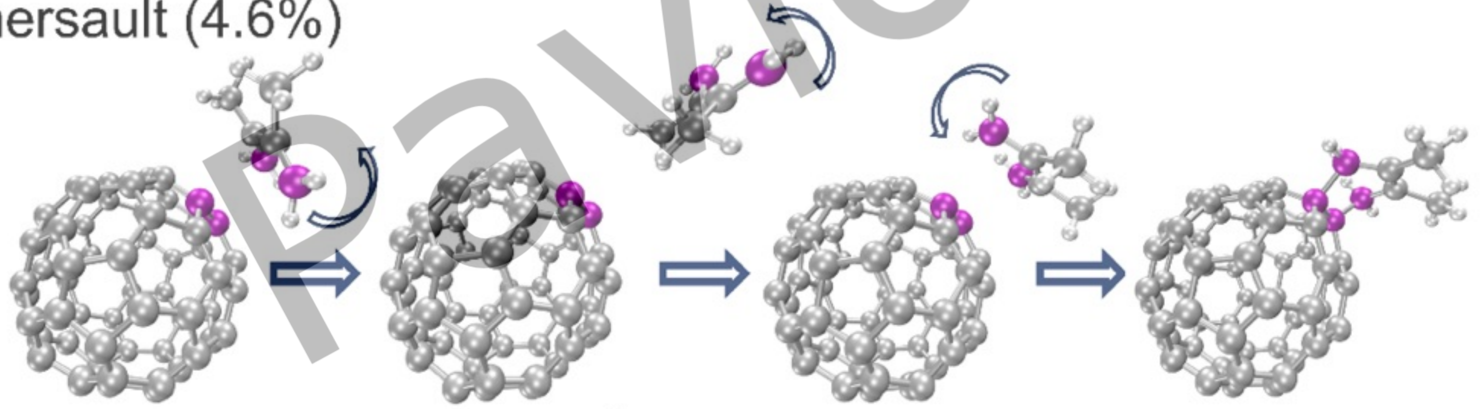
Direct reaction (100%) ethene + 1,3-butadiene

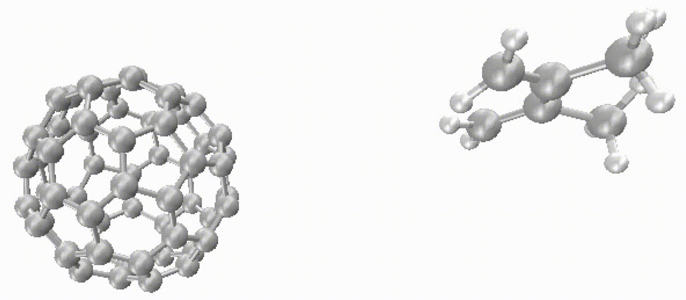


Direct reaction (89.7%) C_{60} + 2,3-dimethyl-1,3-butadiene

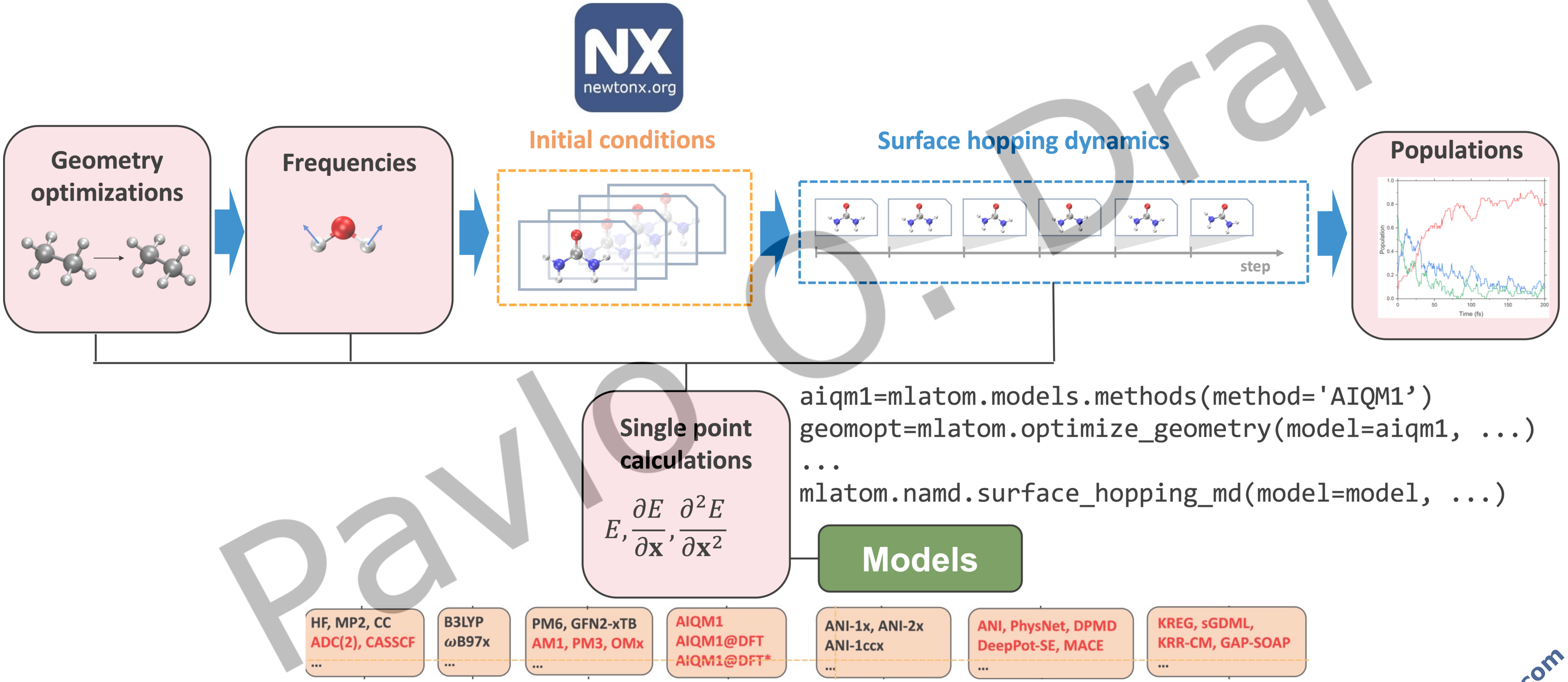


Somersault (4.6%)





Surface-hopping dynamics

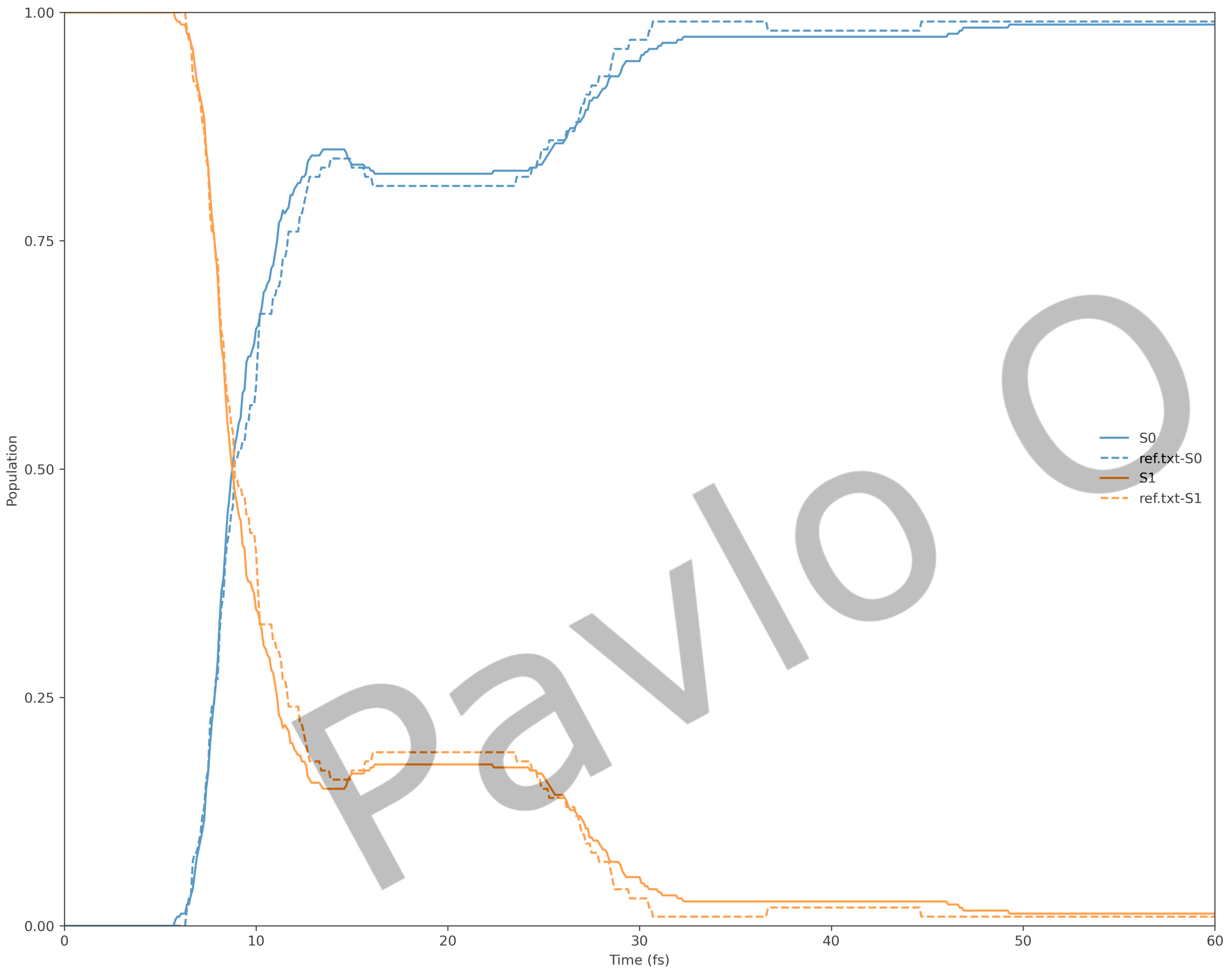


```
aiqm1=mlatom.models.methods(method='AIQM1')
geomopt=mlatom.optimize_geometry(model=aiqm1, ...)
...
mlatom.namd.surface_hopping_md(model=model, ...)
```

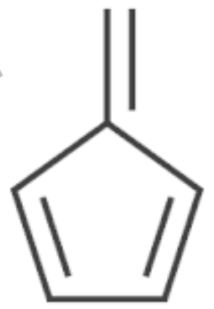
XACS
Xiamen Atomistic Computing Suite
XACScloud.com

Physics-informed active learning

MLatom.com



Fulvene



Wechat

Follow XACS account

Slack

Answering your questions in real time

L. Zhang, Y.-F. Hou, M. Martyka, M. Barbatti, P. O. Dral*, et al. *unpublished*

XACS
Xiamen Atomistic Computing Suite
XACScloud.com

Example AL-1.

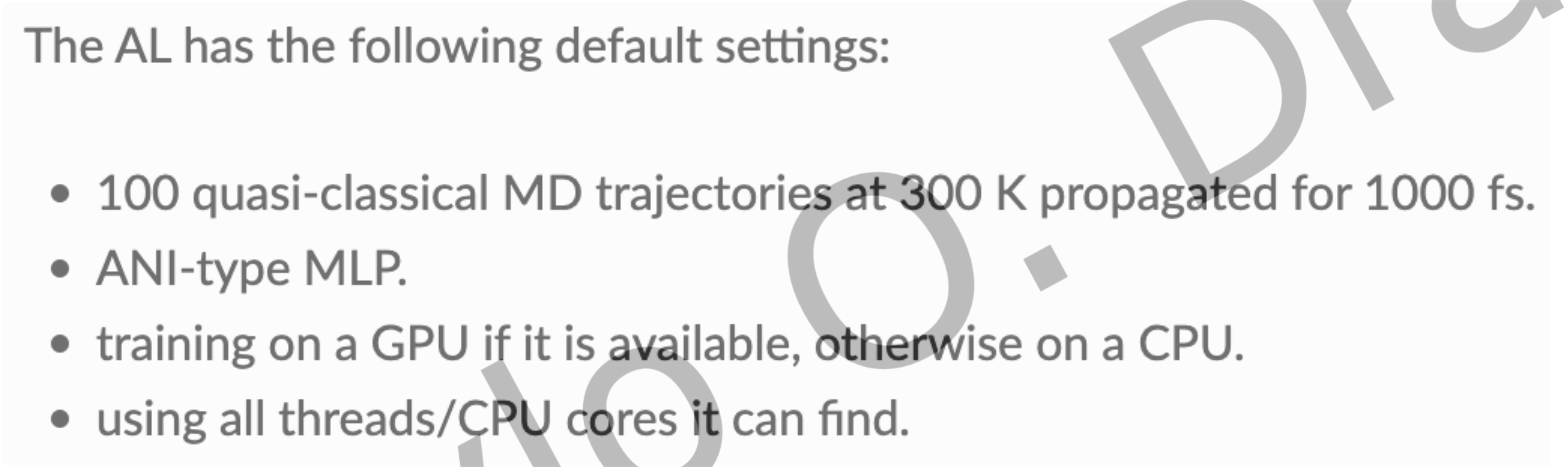
```python
import mlatom as ml

h2 = ml.molecule.from_xyz_string('''2

H 0 0 0
H 0 0 0.7
''')

dft = ml.models.methods(method='B3LYP/6-31G*')

# Optimize geometry and calculate frequency
optmol = ml.optimize_geometry(
    model=dft,
    initial_molecule=h2
    ).optimized_molecule
freq = ml.freq(model=dft,
molecule=optmol)

sampler_kwargs = {
    'initcond_sampler':'wigner',
    'initcond_sampler_kwargs':{
        'molecule':optmol,
        'number_of_initial_conditions':100,
        'initial_temperature':300},
    'maximum_propagation_time':100.0,
    'time_step':0.5,
    'uq_threshold':0.000001,
}

# Start active learning
ml.al(molecule=optmol,
      ml_model='KREG',
      sampler_kwargs=sampler_kwargs,
      min_new_points=5,
      reference_method=dft)
```

MLatom.com

Python: highly flexible, can set better settings

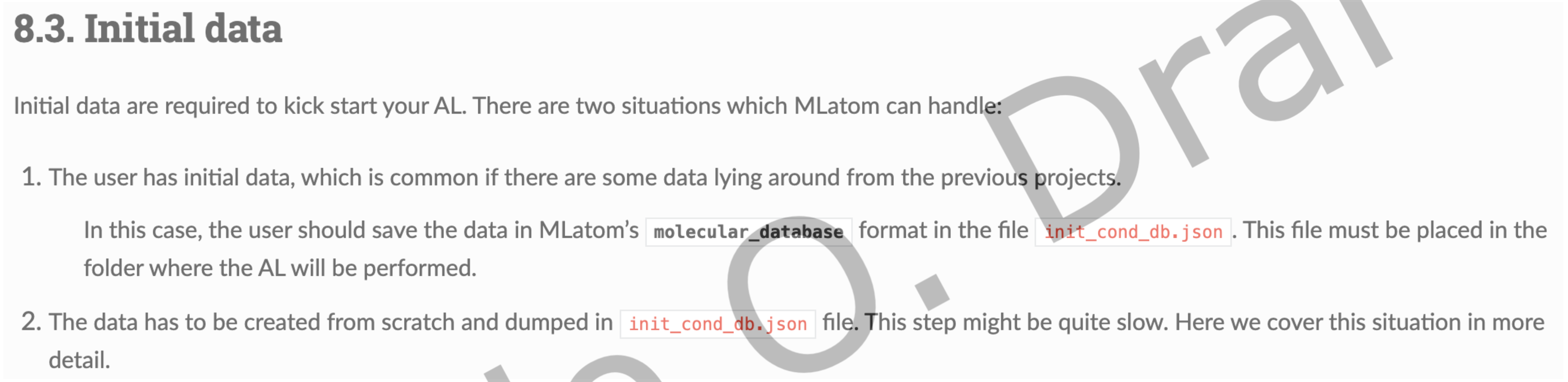
Slow and not flexible...

```
AL # request active learning
B3LYP/6-31G* # reference method to label points
# initial guess for the geometry, does not matter much
xyzfile='2

H 0 0 0
H 0 0 0.7'
```



The AL has the following default settings:

- 100 quasi-classical MD trajectories at 300 K propagated for 1000 fs.
- ANI-type MLP.
- training on a GPU if it is available, otherwise on a CPU.
- using all threads/CPU cores it can find.



8.3. Initial data

Initial data are required to kick start your AL. There are two situations which MLatom can handle:

1. The user has initial data, which is common if there are some data lying around from the previous projects.

   In this case, the user should save the data in MLatom's `molecular_database` format in the file `init_cond_db.json`. This file must be placed in the folder where the AL will be performed.

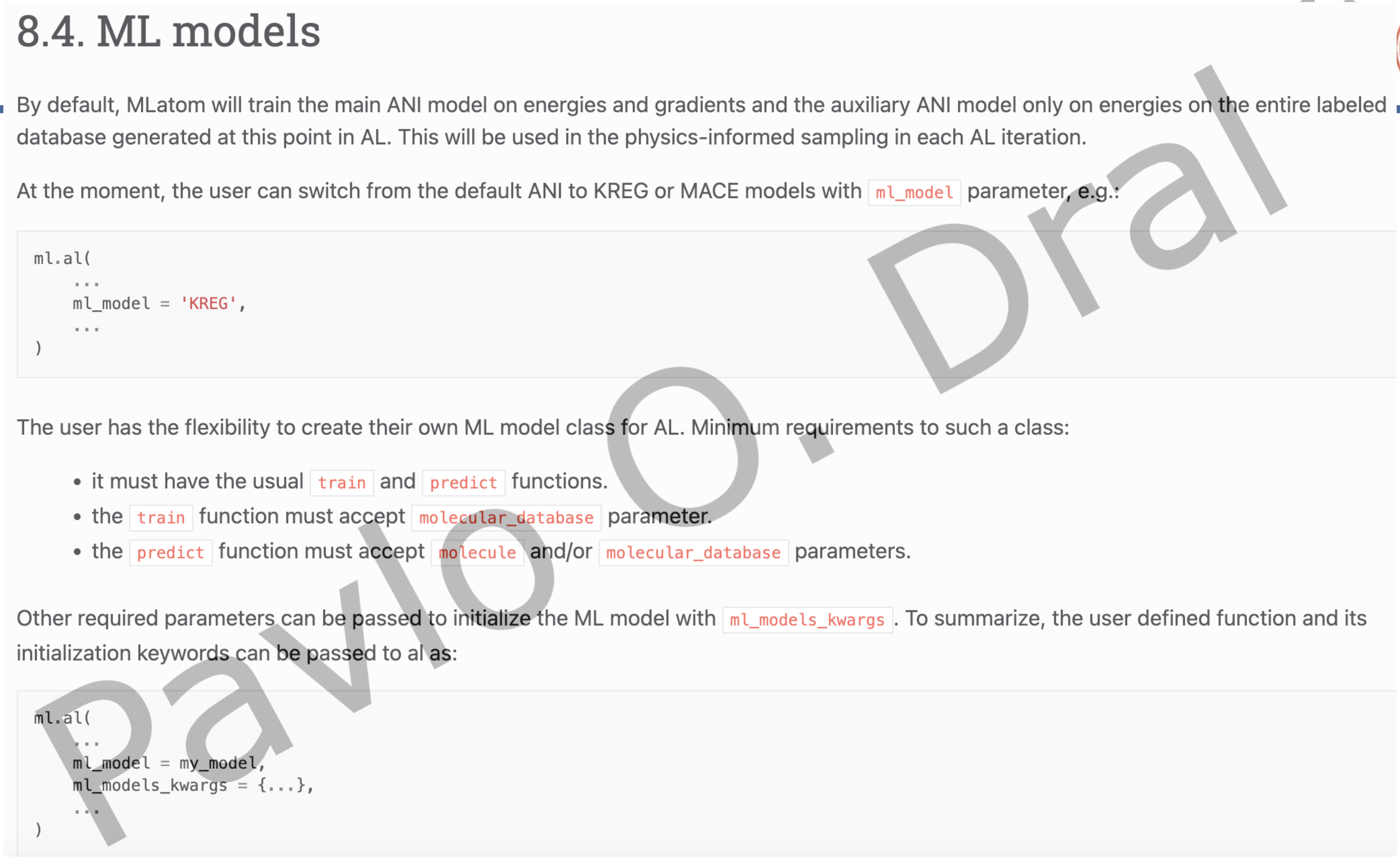
2. The data has to be created from scratch and dumped in `init_cond_db.json` file. This step might be quite slow. Here we cover this situation in more detail.

By default, we use the normal mode (Wigner) sampling to generate the initial data. Also, the reference method used for labeling, is used for geometry optimization and calculation of frequencies and normal modes. The number of initial data points is determined iteratively so that the cross-validation error of ML model would not change by more than 10% after adding additional 50 points based on the learning curve fit (minimum 5 iterations are used for the fit). For judging error, the model is trained only on energies to speed up this step in AL.

The sampling method is defined by parameters `initdata_sampler` and `initdata_sampler_kwargs` which are the most important ones you might want to change (see the manual to see how to change other parameters):

```
ml.al(
    ...
    initdata_sampler=init_dataset_sampler,
    initdata_sampler_kwargs=init_dataset_sampler_kwargs,
    ...
)
```

```
optfreq_method = ml.models.methods(method='ANI-1ccx')
eqmol = ml.optimize_geometry(initial_molecule=h2, model=optfreq_method).optimized_molecule
eqmol = ml.freq(molecule=eqmol, model=optfreq_method)

# or load the molecule with normal modes and frequencies
# eqmol = ml.molecule()
# eqmol.load(filename='eqmol.json', format='json')

init_dataset_sampler = 'wigner'
# or explicitly:
# initial_points_sampler = ml.generate_initial_conditions
init_dataset_sampler_kwargs = {
    #'generation_method': 'Wigner', # required for ml.generate_initial_conditions (although it is default there too)
    'molecule': eqmol,
    'initial_temperature': 500,        # temperature of Wigner sampling in K
    'number_of_initial_conditions': 50 # number of points sampled at one time
}

ml.al(
    ...
    initdata_sampler=init_dataset_sampler,
    initdata_sampler_kwargs=init_dataset_sampler_kwargs,
    ...
)
```



8.4. ML models

By default, MLatom will train the main ANI model on energies and gradients and the auxiliary ANI model only on energies on the entire labeled database generated at this point in AL. This will be used in the physics-informed sampling in each AL iteration.

At the moment, the user can switch from the default ANI to KREG or MACE models with `ml_model` parameter, e.g.:

```
ml.al(
    ...
    ml_model = 'KREG',
    ...
)
```

The user has the flexibility to create their own ML model class for AL. Minimum requirements to such a class:

- it must have the usual `train` and `predict` functions.
- the `train` function must accept `molecular_database` parameter.
- the `predict` function must accept `molecule` and/or `molecular_database` parameters.

Other required parameters can be passed to initialize the ML model with `ml_models_kwargs`. To summarize, the user defined function and its initialization keywords can be passed to al as:

```
ml.al(
    ...
    ml_model = my_model,
    ml_models_kwargs = {...},
    ...
)
```

```
# let's define our MD for sampling

# ... but before we start, MD needs initial conditions which should always be different, hence, we need another sampler for initial conditions!
md_initconds = 'wigner'
# or explicitly:
# md_initconds = ml.generate_initial_conditions
md_initconds_kwargs = {
    'molecule': eqmol, # must contain normal modes and frequencies!
    #'generation_method': 'Wigner',
    'initial_temperature': 300,        # temperature of Wigner sampling in K
    'number_of_initial_conditions': 100 # defines number of trajectories
}

# ... now we can define our MD sampler
al_sampler = 'md'
# faster and default is batch md, when forces of multiple points in parallel in batches:
#al_sampler = 'batch_md'
sampler_kwargs = {
    'initcond_sampler': init_cond_sampler,
    'initcond_sampler_kwargs': md_initconds_kwargs,
    'maximum_propagation_time':5000.0, # Maximum propagation time; unit: fs
    'time_step':0.5,  # MD time step; unit: fs
}

# Finally, let's pass this info to AL

ml.al(
    ...
    sampler=al_sampler,
    sampler_kwargs=sampler_kwargs,
    ...
)
```

Sampling function can also be defined by the user. This function must accept `ml_model` parameter and return `molecule_database` object with the points to label. Generic example:

```
def my_sampler(ml_model=None, kwarg1, kwarg2, ...):
    moldb2label = ml.data.molecular_database()
    #do what you need to collect points
    ...
    moldb2label.append(newmol2label)
    return moldb2label

ml.al(
    ...
    sampler=my_sampler,
    sampler_kwargs={kwarg1_with_value, kwarg2_with_value, ...},
    ...
)
```

In addition, we provide an important option that affects the quality of sampling: `uq_threshold`. Although it is determined automatically, the user can overwrite it with another number - the smaller the stricter and may lead to many more iterations and sampled points.

Example with these two options (0.5 kcal/mol for the UQ threshold):

```
ml.al(
    ...
    sampler='md',
    sampler_kwargs={
    'uq_threshold': 0.5/ml.constants.kcalpermol2Hartree,
    ...},
    ...
)
```

```
ml.al(
    ...
    max_iterations = 5,
    new_points     = 300,
    min_new_points = 10,
    ...
)
```

```python
import mlatom as ml

ethanol = ml.molecule.from_xyz_string('''...
''')
eqmol = ml.optimize_geometry(initial_molecule=ethanol, model=optfreq_method).optimized_molecule
eqmol = ml.freq(molecule=eqmol, model=optfreq_method)

dft = ml.models.methods(method='B3LYP/6-31G*')
init_cond_sampler = 'wigner' # for both initial data and MD
init_dataset_sampler_kwargs = {
    'molecule':eqmol,
    'initial_temperature': 300,
    'number_of_initial_conditions': 50
}
md_initconds_kwargs = {
    'molecule':eqmol,
    'initial_temperature': 300,
    'number_of_initial_conditions': 100
} # We want 100 trajectories at each iteration
# ... now we can define our MD sampler
al_sampler = 'batch_md'
sampler_kwargs = {
    'initcond_sampler': init_cond_sampler,
    'initcond_sampler_kwargs': md_initconds_kwargs,
    'maximum_propagation_time':5000.0, # Maximum propagation time; unit: fs
    'time_step':0.5,  # MD time step; unit: fs
}
ml.al(
    molecule=ethanol,
    reference_method=dft,
    initdata_sampler=init_cond_sampler,
    initdata_sampler_kwargs=init_dataset_sampler_kwargs,
    sampler=al_sampler,
    sampler_kwargs=sampler_kwargs,
    new_points = 300,
)
```

```
...
# copy or give the path to the model from the final iteration
ani = ml.models.ani(model_file='iteration10/mlmodel.pt')
ml.optimize_geometry(molecule=ethanol, model=ani)
...
```