XACS
Xiamen Atomistic Computing Suite
XACScloud.com

MLatom.com

Advanced course on ML

Pavlo O. Dral

Xiamen University, P.R. China

Visiting Professor in
Nicolaus Copernicus University, Poland

5 July 2024

第3题：您对机器学习的哪些课程比较感兴趣（可多选） [多选题]

选项	小计	比例
主动学习	28	68.29%
迁移学习	26	63.41%
Delta学习	16	39.02%
面跃迁动力学	17	41.46%
准经典动力学	24	58.54%
过渡态及分析	32	78.05%
⊞ 其他 [详细]	2	4.88%
本题有效填写人次	**41**	

4. Active learning (Python)

1. Transfer learning (Python)

5-6 – remaining time

2. MD (quasi-classical)

3. TS

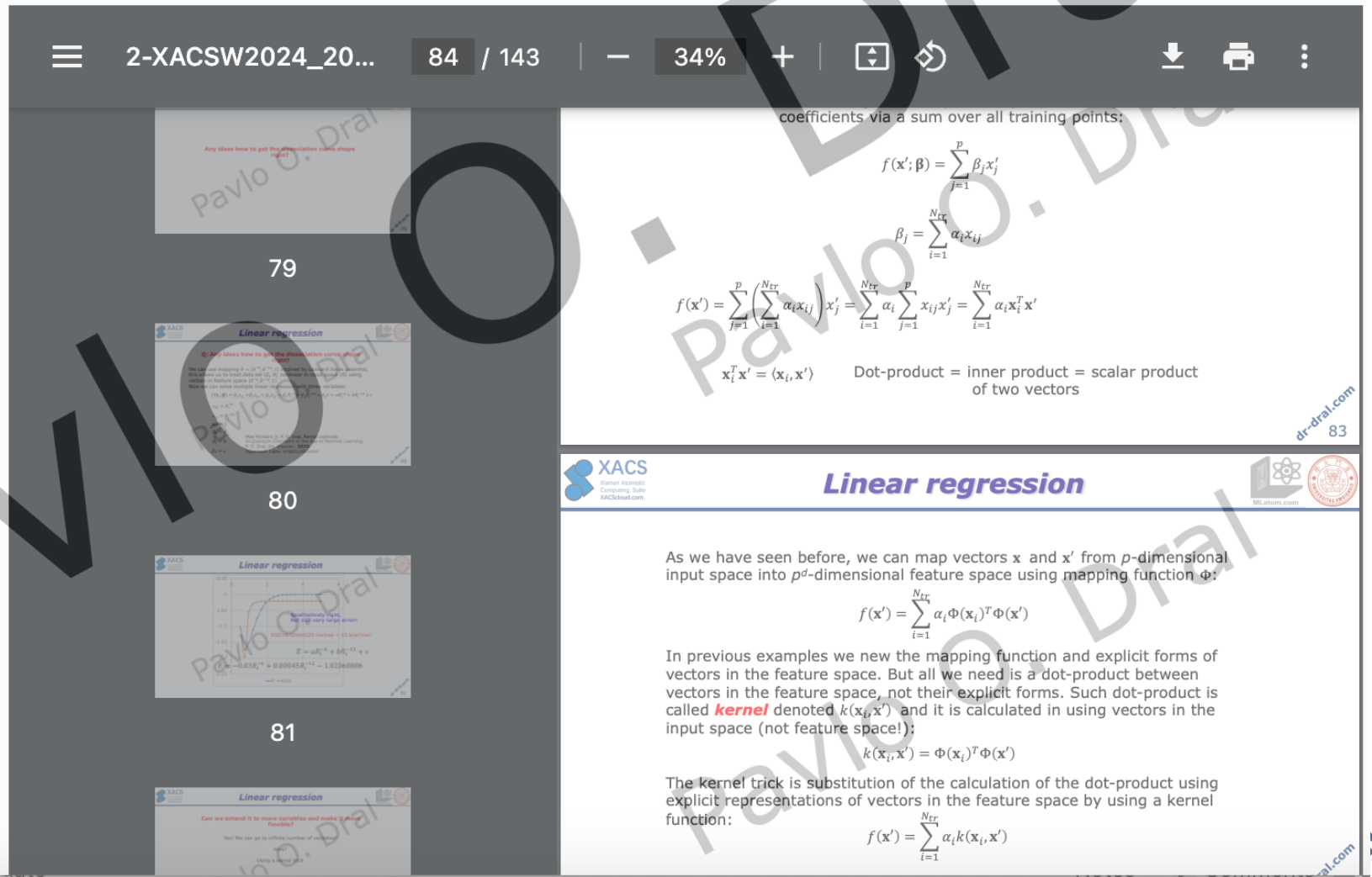
序号	提交答卷时间	答案文本	查看答卷
42	7月4日 14:02	LLM, feature engineering	查看答卷
44	7月4日 14:52	没有ml基础不太懂，所以想学	查看答卷

Slides added to ML basics

1. Machine learning basics

1.1. Slides

Slides:



XACS
Xiamen Atomistic Computing Suite
XACScloud.com

MLatom.com

Transfer learning

Pavlo O. Dral

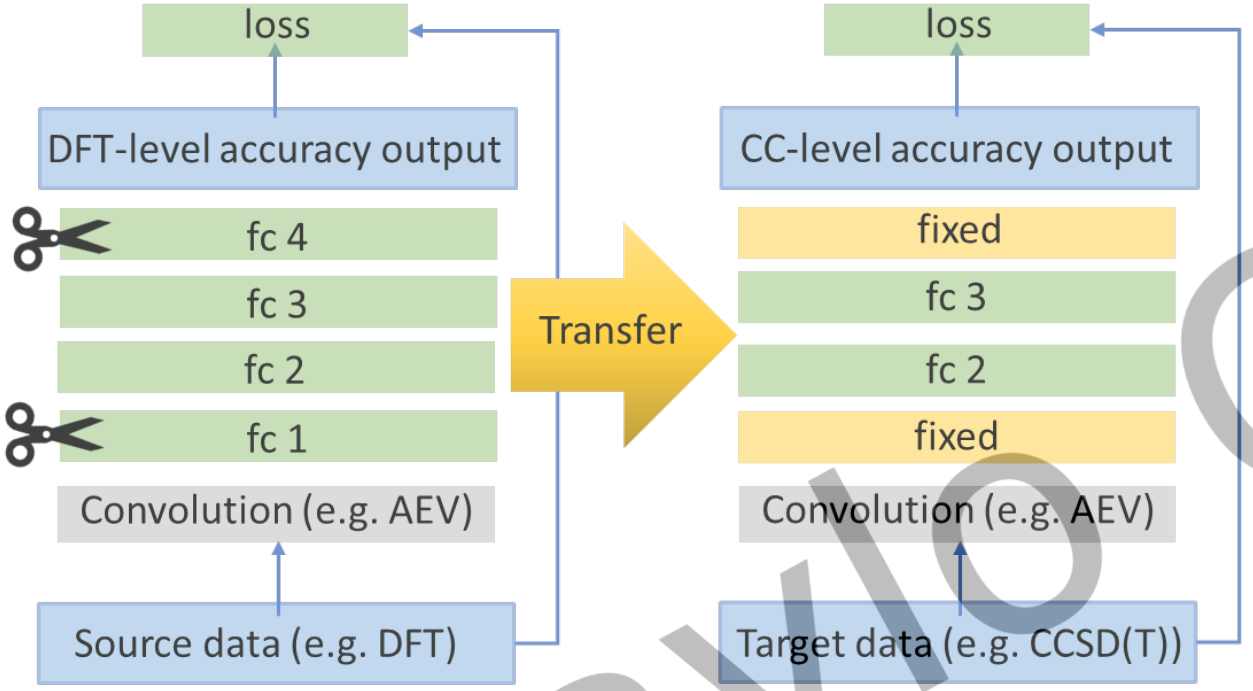
Xiamen University, P.R. China

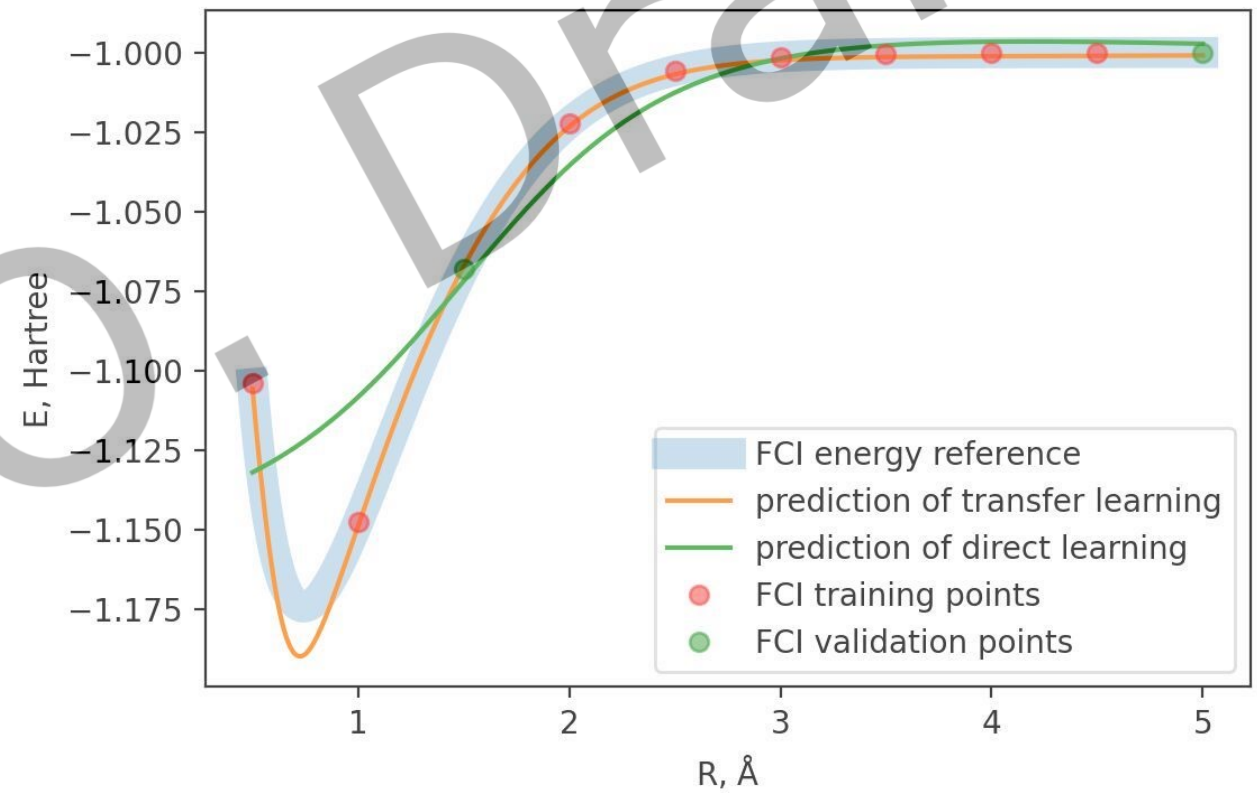
Visiting Professor in
Nicolaus Copernicus University, Poland

5 July 2024

dr-dral.com

Multi-fidelity data: Transfer learning

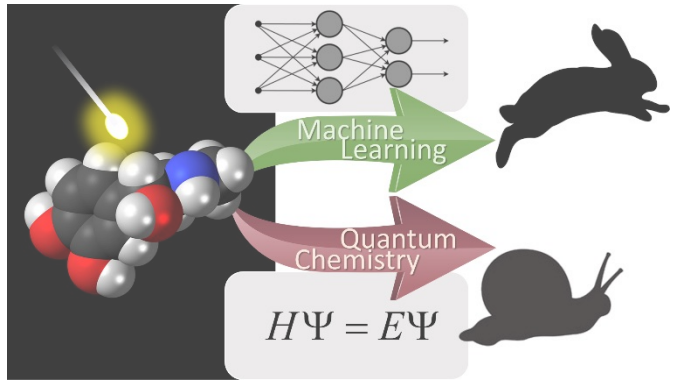




Figures: Pavlo O. Dral, Tetiana Zubatiuk, Bao-Xin Xue, Learning from multiple quantum chemical methods: Δ-learning, transfer learning, co-kriging, and beyond. In *Quantum Chemistry in the Age of Machine Learning*, Pavlo O. Dral, Ed. Elsevier: **2023**. Paperback ISBN: 9780323900492

Machine learning in quantum chemistry

P. O. Dral, M. Barbatti, *Nat. Rev. Chem.* **2021,** *5*, 388



Universal models (no training required):

AIQM1 AIQM1: P. Zheng, R. Zubatyuk, W. Wu, O. Isayev, P. O. Dral, *Nat. Commun.* **2021**, *12*, 7022

ANI-1ccx ANI-1ccx: J. S. Smith, *et al*. *Nat. Commun.* **2019,** *10*, 2903

Perspective: P. O. Dral, *J. Phys. Chem. Lett.* **2020**, *11*, 2336

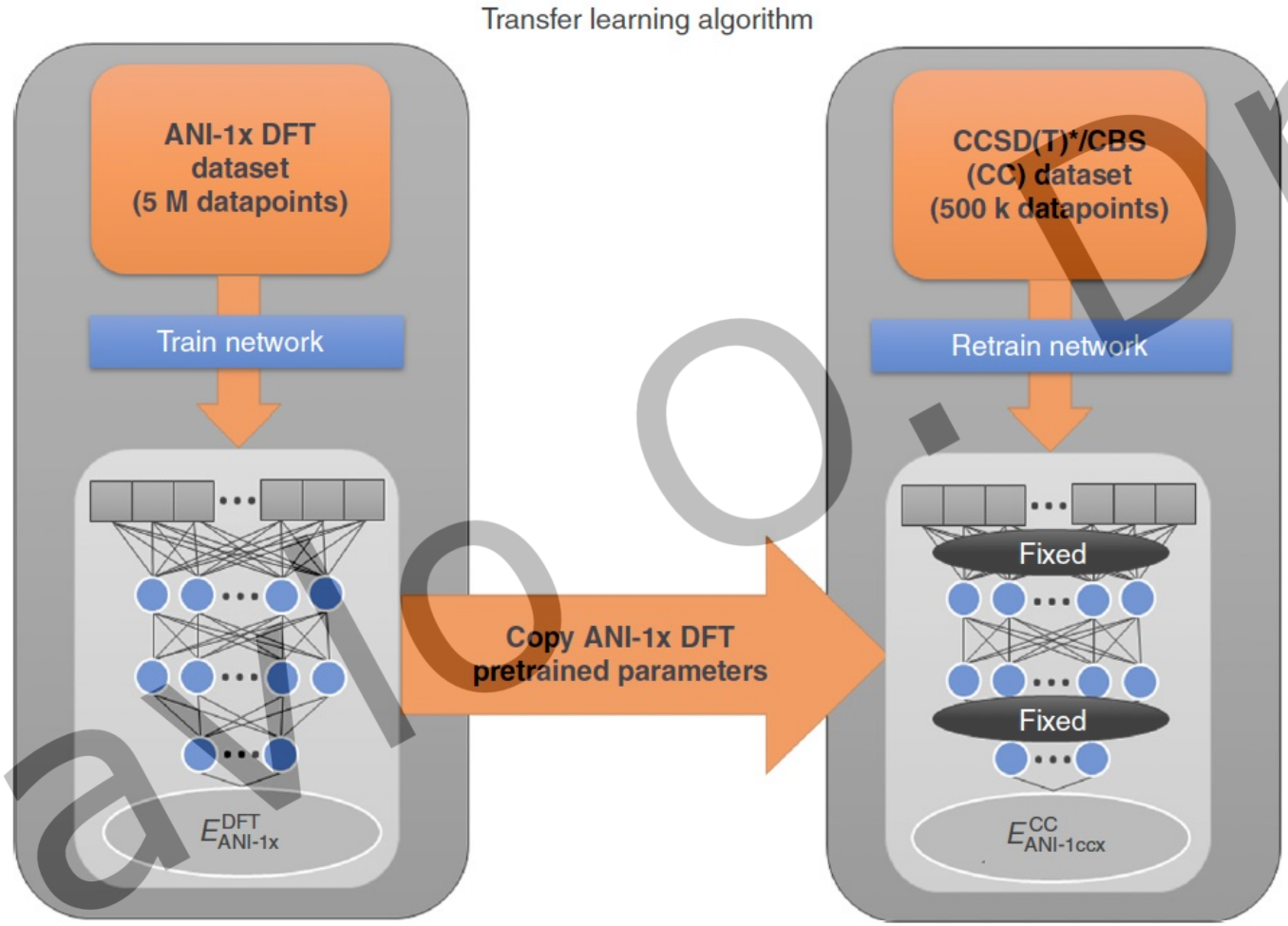
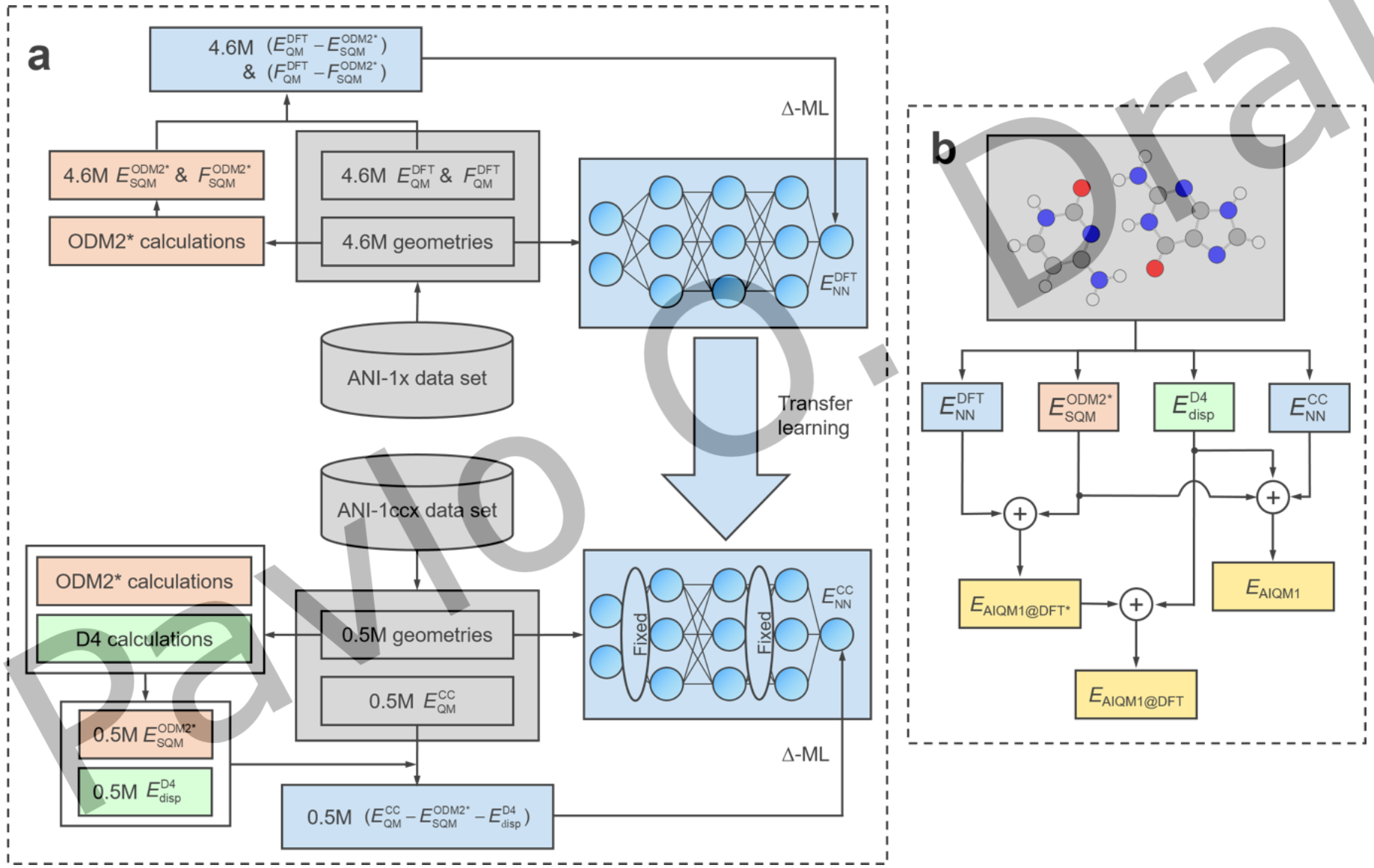


Fig. 4 Diagram of the transfer learning technique evaluated in this work. Transfer learning starts from a pretrained ANI-1x DFT model, then retrains to higher accuracy CCSD(T)*/CBS data with some parameters fixed during training

ANI-1ccx: J. S. Smith, *et al*. *Nat. Commun.* **2019,** *10*, 2903

Design



P. Zheng, R. Zubatyuk, W. Wu, O. Isayev, P. O. Dral, *Nat. Commun.* **2021**, *12*, 7022

5.1. Jupyter notebook

- tutorial notebook ← Download this Jupyter notebook

You can upload it to the XACS cloud, launch Jupyter lab (click 'Connect' after launching), and open it in the Jupyter lab.

5.2. Tutorial

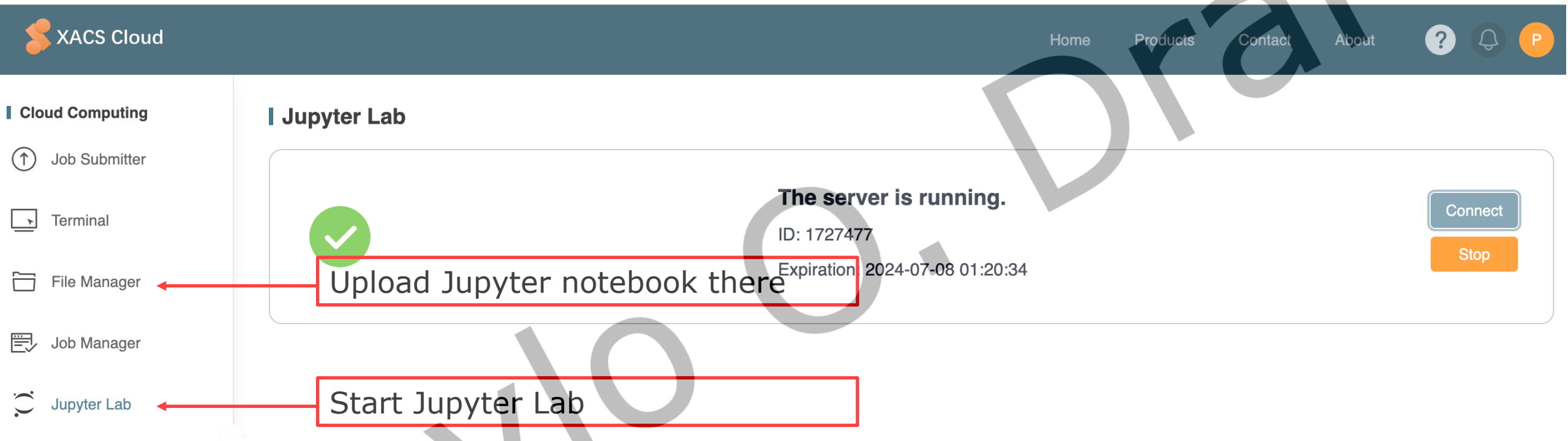
Transfer learning (TL) is an often-used technique in machine learning that helps you train better neural network models. T knowledge (parameters) of the pre-trained model from a different task, by fixing and changing some of its layers and ther There are serval reasons that we might utilize this technique:

- **Better training performance**

  The pre-trained models can give you a good starting point to kick off the training, which helps the convergence ar of the final model.

- **Less data hungry**

  Training from a pre-trained model does not require data as much as training from scratch, since the pre-trained mo information from its training data. This can be critical, especially when the accessibility to the data for the new tas


XACS Cloud
Home
Products
Contact
About
P
Cloud Computing
Job Submitter
Terminal
File Manager
Job Manager
Jupyter Lab
Jupyter Lab
The server is running.
ID: 1727477
Expiration: 2024-07-08 01:20:34
Connect
Stop
Upload Jupyter notebook there
Start Jupyter Lab

tl.ipynb

Code

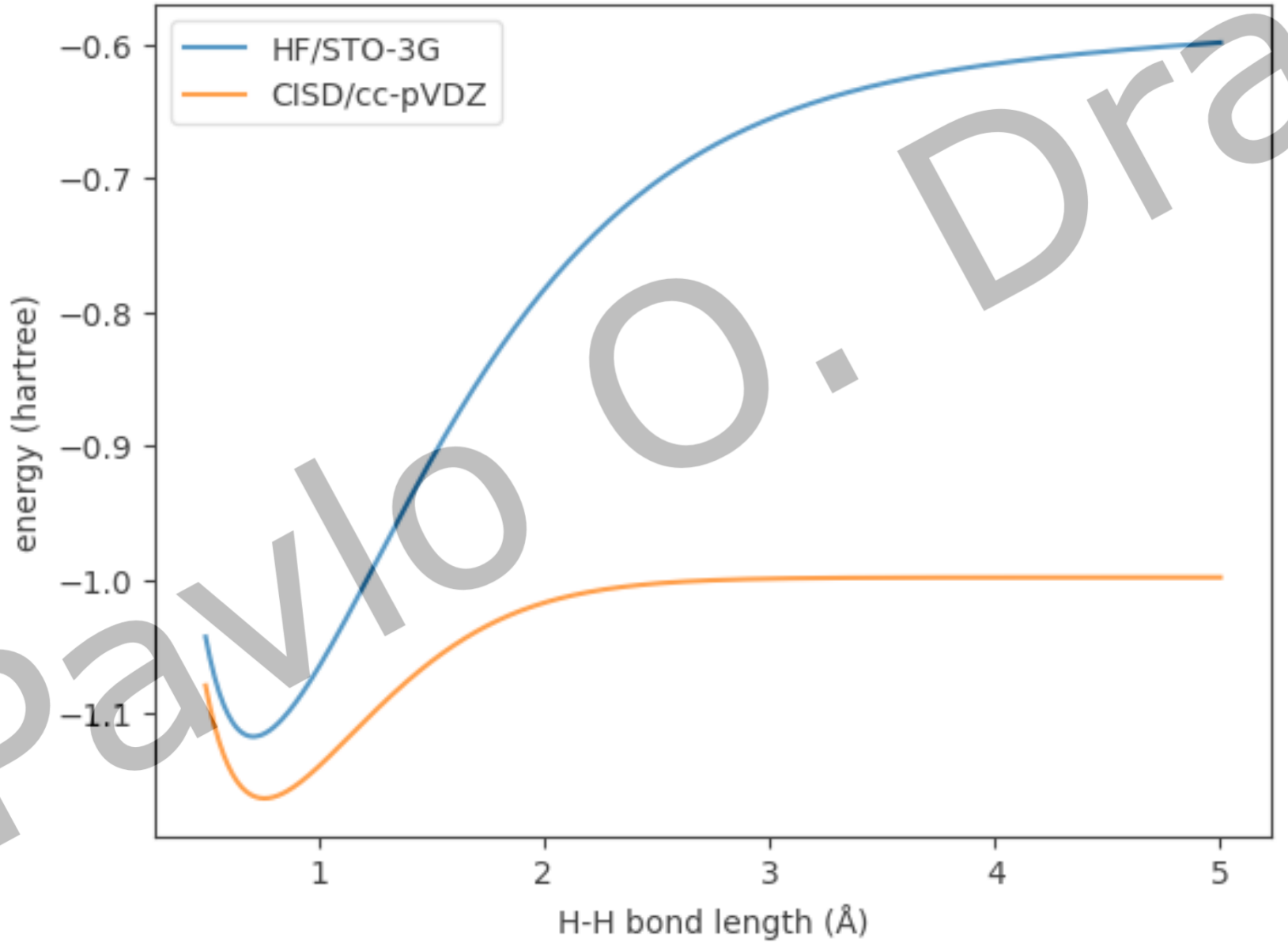
[1]:
```
import mlatom as ml
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# prepare H2 geometries with bond lengths ranging from 0.5 to 5.0 Å
xyz = np.zeros((451, 2, 3))
xyz[:, 1, 2] = np.arange(0.5, 5.01, 0.01)
z = np.ones((451, 2)).astype(int)
molDB = ml.molecular_database.from_numpy(coordinates=xyz, species=z)
```

[3]:
```
# calculate HF energies
hf = ml.models.methods(method='HF/STO-3G', program='PySCF')
hf.predict(molecular_database=molDB, calculate_energy=True)
molDB.add_scalar_properties(molDB.get_properties('energy'), 'HF_energy') # save HF energy with a new name
```

[4]:
```
# calculate CISD energies
cisd = ml.models.methods(method='CISD/cc-pVDZ', program='PySCF')
cisd.predict(molecular_database=molDB, calculate_energy=True)
molDB.add_scalar_properties(molDB.get_properties('energy'), 'CISD_energy')
```

[5]:
```
# plot the energies
plt.plot(xyz[:, 1, 2], molDB.get_properties('HF_energy'), label='HF/STO-3G')
plt.plot(xyz[:, 1, 2], molDB.get_properties('CISD_energy'), label='CISD/cc-pVDZ')
plt.legend()
plt.xlabel('H-H bond length (Å)')
plt.ylabel('energy (hartree)')
plt.show()
```


HF/STO-3G
CISD/cc-pVDZ
energy (hartree)
−0.6
−0.7
−0.8
−0.9
−1.0
−1.1
1
2
3
4
5
H-H bond length (Å)

[6]:
```
# train ANI model with HF energies
! rm ANI-HF.pt &> /dev/null
ani = ml.models.ani(model_file='ANI-HF.pt', verbose=False)
ani.train(molecular_database=molDB, property_to_learn='HF_energy')
# predict with trained ANI model
ani.predict(molecular_database=molDB, property_to_predict='ANI_HF_energy')
```

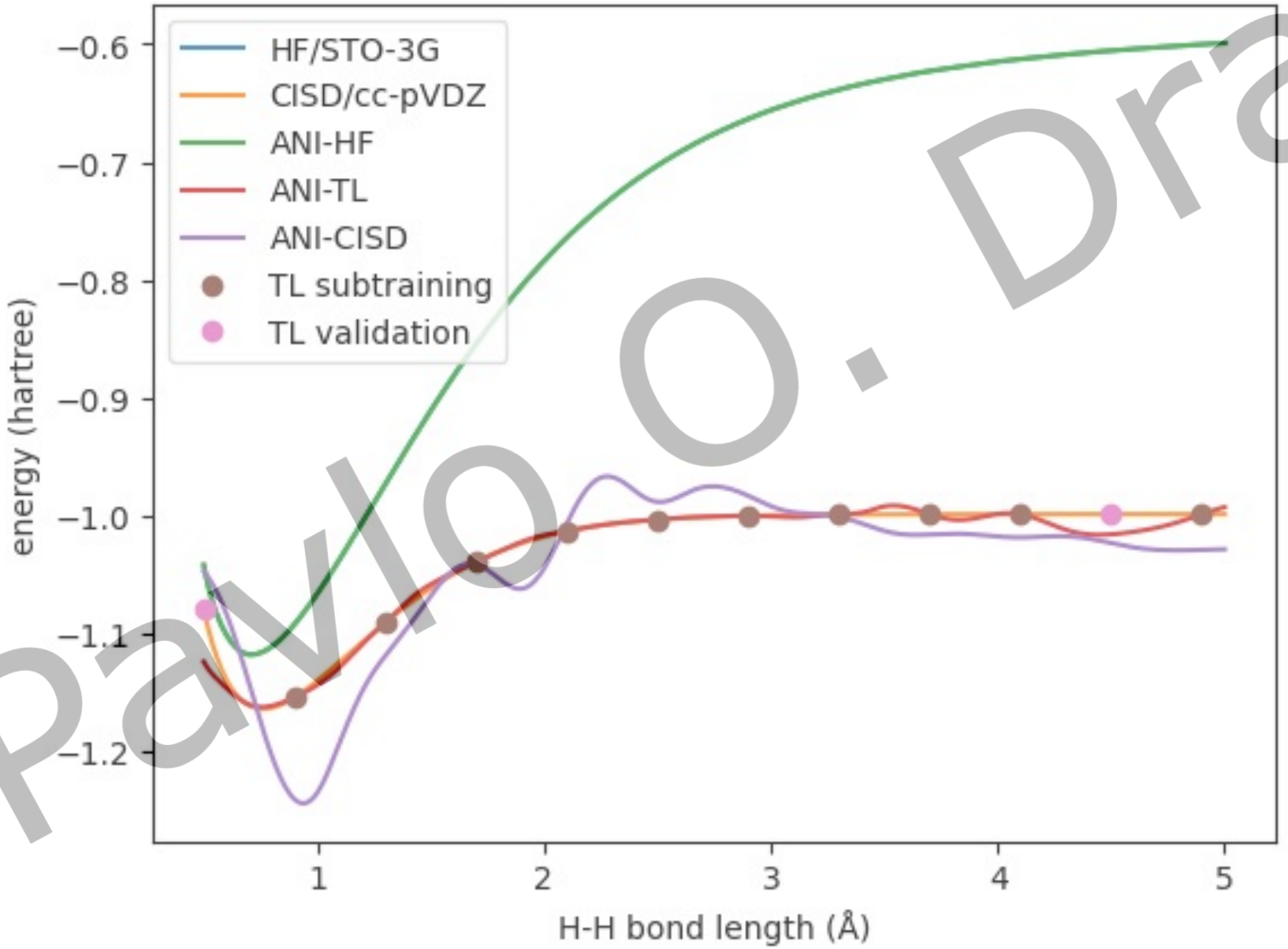
[7]:
```
# show the model structure
ani.model
```

[7]:
```
Sequential(
  (0): AEVComputer()
  (1): ANIModel(
    (H): Sequential(
      (0): Linear(in_features=48, out_features=160, bias=True)
      (1): CELU(alpha=0.1)
      (2): Linear(in_features=160, out_features=128, bias=True)
      (3): CELU(alpha=0.1)
      (4): Linear(in_features=128, out_features=96, bias=True)
      (5): CELU(alpha=0.1)
      (6): Linear(in_features=96, out_features=1, bias=True)
    )
  )
)
```

[8]:
```
# fix some of the layers
! cp ANI-HF.pt ANI-HF-TL.pt
ani = ml.models.ani(model_file='ANI-HF-TL.pt', verbose=False)
ani.fix_layers([[0, 6]])
# transfer leaning with every 40th of the data
step = 40
val = molDB[::step][::10]
sub = ml.molecular_database([mol for mol in molDB[::step] if mol not in val])
ani.energy_shifter.self_energies = None # let the model recalculate the self atomic energies
ani.train(molecular_database=sub, validation_molecular_database=val, property_to_learn='CISD_energy',
          hyperparameters={'learning_rate': 0.0001}, reset_optimizer=True)
# predict with TL model
ani.predict(molecular_database=molDB, property_to_predict='ANI_TL_energy')
```

[9]:
```
# train ANI model with CISD energies directly
! rm ANI_CISD.pt &> /dev/null
ani_cisd = ml.models.ani(model_file='ANI_CISD.pt', verbose=False)
ani_cisd.train(molecular_database=sub, validation_molecular_database=val, property_to_learn='CISD_energy')
# predict with trained ANI model
ani_cisd.predict(molecular_database=molDB, property_to_predict='ANI_CISD_energy')
```

[10]:
```
# plot the energies
plt.plot(xyz[:, 1, 2], molDB.get_properties('HF_energy'), label='HF/STO-3G')
plt.plot(xyz[:, 1, 2], molDB.get_properties('CISD_energy'), label='CISD/cc-pVDZ')
plt.plot(xyz[:, 1, 2], molDB.get_properties('ANI_HF_energy'), label='ANI-HF')
plt.plot(xyz[:, 1, 2], molDB.get_properties('ANI_TL_energy'), label='ANI-TL')
plt.plot(xyz[:, 1, 2], molDB.get_properties('ANI_CISD_energy'), label='ANI-CISD')
plt.plot(sub.xyz_coordinates[:, 1, 2], sub.get_properties('CISD_energy'), 'o', label='TL subtraining')
plt.plot(val.xyz_coordinates[:, 1, 2], val.get_properties('CISD_energy'), 'o', label='TL validation')
plt.legend()
plt.xlabel('H-H bond length (Å)')
plt.ylabel('energy (hartree)')
plt.show()
```


HF/STO-3G
CISD/cc-pVDZ
ANI-HF
ANI-TL
ANI-CISD
TL subtraining
TL validation
energy (hartree)
−0.6
−0.7
−0.8
−0.9
−1.0
−1.1
−1.2
1
2
3
4
5
H-H bond length (Å)