aenet_tutorial_beginners_guide

March 25, 2021

This notebook is a companion to Miksch, Morawietz, Kaestner, Urban, Artrith, to be published (2021).

The tutorial demonstrates the construction of an artificial neural network (ANN) interatomic potential for water clusters with six water molecules, using the data set from [1].

[1] A. M. Cooper, J. Kästner, A. Urban, N. Artrith, npj Comput. Mater. 6, 54 (2020); Data: https://archive.materialscloud.org/record/2020.0037/v1

1 Prerequisites - installation of ænet and ASE

We will use the atomic energy network (ænet) package (http://ann.atomistic.net) artificial neural network (ANN) potential package

```
N. Artrith and A. Urban, Comput. Mater. Sci. 114 (2016) 135-150.
N. Artrith, A. Urban, and G. Ceder, Phys. Rev. B 96 (2017) 014112.
```

and the atomic simulation environment (ASE) package (https://wiki.fysik.dtu.dk/ase) for structure manipulation and to perform MD simulations

```
Larsen et al., J. Phys.: Condens. Matter Vol. 29 273002, 2017
```

Also *clone* the companion GitHub repository with files for this tutorial (https://github.com/atomisticnet/MLP-beginners-guide).

[1]:
```
try:
  import aenet
  print("successfully imported aenet")
except ImportError:
  !! git clone https://github.com/atomisticnet/aenet.git
  !! cd aenet/lib && make
  !! cd aenet/src && make -f makefiles/Makefile.gfortran_serial
  !! cd aenet/src && make -f makefiles/Makefile.gfortran_serial lib
  !! cd aenet/python3 && python3 setup.py build_ext --inplace
  !! cd aenet/python3 && pip install -e . --user
```

```
  print("completed installation of aenet")
```

```
completed installation of aenet
```

[2]:
```
try:
  import ase
  print("successfully imported ase")
except ImportError:
  !! pip install ase --user --upgrade
  print("completed installing ASE")
```

```
completed installing ASE
```

Get additional materials for this tutorial from GitHub (https://github.com/atomisticnet/MLP-beginners-guide).

[3]:
```
import os
if not os.path.exists('MLP-beginners-guide'):
  !! git clone https://github.com/atomisticnet/MLP-beginners-guide.git
else:
  print("Tutorial files are already installed.")
```

Now: Notebook has to be restarted to activate the packages that we just installed.

[1]:
```
try:
  import aenet
  import ase
  print("Both aenet and ase could be imported. You are all set for the tutorial.
 ↪")
except ImportError:
  print("The notebook needs to be restarted. Run this cell again once the␣
 ↪restart is done.")
  import os
  os.kill(os.getpid(), 9)
```

```
Both aenet and ase could be imported. You are all set for the tutorial.
```

2 Python imports

We need to import Python packages/libaries that we will use below.

[2]:
```
import numpy as np
import pandas as pd
import re
from IPython.display import Image
%matplotlib inline
import matplotlib.pyplot as plt
```

```
plt.rcParams.update({"font.size": 15})
plt.close("all")
```

3 01-generate: Generating the reference data set

The first step is the transformation of the atomic structures in the reference data set into a feature-vector representation of local atomic environments. This is done with the ænet tool `generate.x`.

We enter a subdirectory in which we will generate the reference data set.

[3]:
```
%cd /content/MLP-beginners-guide/tutorial/01_generate/
%ls
```

```
/content/MLP-beginners-guide/tutorial/01_generate
generate.in        O.fingerprint.stp  sort-list.dat
H.fingerprint.stp  output/            Stuttgart-HCP-generate.in
```

`generate.x` is a command-line tool that reads input parameters from text files. The file `generate.in` is the pricipal input file containing information about the chemical species and the structures in the reference data set:

- The name of the *training set* output file to be generated,
- Atomic energies for each species that are subtracted from the total energy before the potential is trained,
- The file names of the *fingerprint* set-ups for each species that define how the local atomic environment is described, and
- The paths of the atomic structure files in the reference data set.

For more details, see also the aenet documentation: http://ann.atomistic.net/documentation/

[4]:
```
! head -n 15 generate.in
```

```
OUTPUT H2O.train

TYPES
2
H 0.000000000
O 0.000000000

SETUPS
H    H.fingerprint.stp
O    O.fingerprint.stp

FILES
1886
../water-clusters-BLYP-D3/train_2000_xsf/train0500_1.xsf
../water-clusters-BLYP-D3/train_2000_xsf/train0500_10.xsf
```

Let's take a look at the *fingerprint* set-up file for oxygen. The one for hydrogen is similar. This example uses

- A BP2011 descriptor [1] with 30 symmetry functions of type 2 and 4,
- The radial cutoff distance is set to 12 Bohr (6.35 Å), and
- The minimal distance between atoms is set to be 0.75 Bohr (0.4 Å).

[1] J. Behler, J. Chem. Phys. 134, 074106 (2011).

Note: The water reference data uses the units Hartree and Bohr for energies and lenghts. aenet ANN potentials are agnostic to the choice of units, but the aenet will (incorrectly) report electron-volts (eV) and Angstrom (Å) units in its output files.

[5]:
```
! cat O.fingerprint.stp
```

```
DESCR
  Structural fingerprint setup for O in bulk TiO2.
  Ref.: N. Artrith and A. Urban, Comp. Mater. Sci. (2016)
END DESCR

ATOM O

ENV  2
H
O

RMIN 0.75d0

SYMMFUNC type=Behler2011
30
G=2 type2=H   eta=0.001  Rs=0.0000  Rc=12.0000
G=2 type2=O   eta=0.001  Rs=0.0000  Rc=12.0000
G=2 type2=H   eta=0.010  Rs=0.0000  Rc=12.0000
G=2 type2=O   eta=0.010  Rs=0.0000  Rc=12.0000
G=2 type2=H   eta=0.030  Rs=0.0000  Rc=12.0000
G=2 type2=O   eta=0.030  Rs=0.0000  Rc=12.0000
G=2 type2=H   eta=0.090  Rs=0.0000  Rc=12.0000
G=2 type2=O   eta=0.090  Rs=0.0000  Rc=12.0000
G=2 type2=H   eta=0.150  Rs=0.9000  Rc=12.0000
G=2 type2=O   eta=0.150  Rs=4.0000  Rc=12.0000
G=2 type2=H   eta=0.300  Rs=0.9000  Rc=12.0000
G=2 type2=O   eta=0.300  Rs=4.0000  Rc=12.0000
G=2 type2=H   eta=0.600  Rs=0.9000  Rc=12.0000
G=2 type2=O   eta=0.600  Rs=4.0000  Rc=12.0000
G=2 type2=H   eta=1.500  Rs=0.9000  Rc=12.0000
G=2 type2=O   eta=1.500  Rs=4.0000  Rc=12.0000
G=4 type2=H  type3=O    eta=0.001 lambda= -1.0  zeta= 4.0  Rc=12.0000
G=4 type2=O  type3=O    eta=0.001 lambda= -1.0  zeta= 4.0  Rc=12.0000
G=4 type2=H  type3=O    eta=0.001 lambda=  1.0  zeta= 4.0  Rc=12.0000
```

```
G=4 type2=O  type3=O    eta=0.001 lambda=  1.0  zeta= 4.0  Rc=12.0000
G=4 type2=H  type3=H    eta=0.010 lambda= -1.0  zeta= 4.0  Rc=12.0000
G=4 type2=H  type3=H    eta=0.010 lambda=  1.0  zeta= 4.0  Rc=12.0000
G=4 type2=H  type3=H    eta=0.030 lambda= -1.0  zeta= 1.0  Rc=12.0000
G=4 type2=O  type3=O    eta=0.030 lambda= -1.0  zeta= 1.0  Rc=12.0000
G=4 type2=O  type3=O    eta=0.030 lambda= -1.0  zeta= 1.0  Rc=12.0000
G=4 type2=H  type3=H    eta=0.030 lambda=  1.0  zeta= 1.0  Rc=12.0000
G=4 type2=H  type3=O    eta=0.030 lambda=  1.0  zeta= 1.0  Rc=12.0000
G=4 type2=O  type3=O    eta=0.030 lambda=  1.0  zeta= 1.0  Rc=12.0000
G=4 type2=H  type3=O    eta=0.070 lambda= -1.0  zeta= 1.0  Rc=12.0000
G=4 type2=H  type3=O    eta=0.070 lambda=  1.0  zeta= 1.0  Rc=12.0000
```

In this example, everything is already prepared to start with the reference data set generation. So, we can simply run `generate.x` and wait until all structures have been processed. We collect the output that `generate.x` prints to the screen in a file named `generate.out`. Note that this can take a minute:

[6]:
```
! /content/aenet/bin/generate.x-2.0.4-gfortran_serial generate.in > generate.out
print("Reference data set generation completed.")
%ls
```

```
Reference data set generation completed.
generate.in   H2O.train          O.fingerprint.stp  sort-list.dat
generate.out  H.fingerprint.stp  output/            Stuttgart-
HCP-generate.in
```

The training-set file `H2O.train` has been generated. **Note that training-set files are binary files and cannot be opened with a text editor.**

Now, we are ready to train our H2O cluster potential. Let's return to the main directory.

[7]:
```
%cd /content/
```

```
/content
```

4 02-train: Training an ANN potential

In this section, we will use the reference data set generated in the previous section to train an artificial neural network (ANN) potential.

Potential training is also done with a command-line tool, which is named `train.x`.

[8]:
```
%cd /content/MLP-beginners-guide/tutorial/02_train/
%ls
```

```
/content/MLP-beginners-guide/tutorial/02_train
train.in
```

An input file for `train.x`, named `train.in`, has already been prepared for us. This file contains the following instructions

- The path to the reference data set file (`H2O.train`),
- The fraction of the data set that should be set aside as validation set,
- The number of training iterations (epochs) to perform,
- The training method and its parameters, and
- The *architecture* of the atomic-energy ANNs for each species.

For more details, see also the aenet documentation: http://ann.atomistic.net/documentation/

[9]:
```
%cat train.in
```

```
TRAININGSET H2O.train
TESTPERCENT 10
ITERATIONS  100

METHOD
bfgs

SAVE_ENERGIES

NETWORKS
! atom   network         hidden
! types  file-name       layers  nodes:activation
  O      O.10t-10t.nn    2       10:tanh 10:tanh
  H      H.10t-10t.nn    2       10:tanh 10:tanh
```

Before we can start with training, the reference data set file needs to be made available. We will create a symbolic link to it in the present directory.

[10]:
```
! ln -s /content/MLP-beginners-guide/tutorial/01_generate/H2O.train
%ls
```

```
H2O.train@  train.in
```

Now, we perform the training. Note that this can take a few minutes. We can observe the training iterations.

[11]:
```
! /content/aenet/bin/train.x-2.0.4-gfortran_serial train.in | tee train.out
!! tar cfvz nn.tar.gz *.nn-* && rm -f *.nn-*
%ls
```

```
 ===============================================================================
                       Training process started.
 ===============================================================================

                            2021-03-26  01:11:28
```

```
Copyright (C) 2015-2018 Nongnuch Artrith and Alexander Urban

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
Mozilla Public License, v. 2.0, for more details.

------------------------------------------------------------------------
                      Training set normalization
------------------------------------------------------------------------

The training set will be normalized now.  Depending on its size this
process can take a while.  The normalized data set will be written to
another file. Load that file in future to avoid this step.

Name of the new training set file: H2O.train.scaled

The network output energy will be normalized to the interval [-1,1].
  Energy scaling factor: f = 392.531310
  Atomic energy shift  : s = -25.483040

Training set normalization done.

------------------------------------------------------------------------
                                Networks
------------------------------------------------------------------------

Creating a new H network

Number of layers :   4

Number of nodes (without bias)
and activation type per layer :

      1 :    27
      2 :    10  hyperbolic tangent (tanh)
      3 :    10  hyperbolic tangent (tanh)
      4 :     1  linear function (linear)

Required memory (words) :      48919 (    382.18 KB)

Total number of weights (incl. bias) :      401

Creating a new O network

Number of layers :   4

Number of nodes (without bias)
```

```
and activation type per layer :

      1 :    30
      2 :    10  hyperbolic tangent (tanh)
      3 :    10  hyperbolic tangent (tanh)
      4 :     1  linear function (linear)

Required memory (words) :      65443 (    511.27 KB)

Total number of weights (incl. bias) :      431

-----------------------------------------------------------------------
                          Storing networks
-----------------------------------------------------------------------

Saving the H network to file : H.10t-10t.nn
Saving the O network to file : O.10t-10t.nn

-----------------------------------------------------------------------
                        Training set info.
-----------------------------------------------------------------------

Training set file                   : H2O.train.scaled
Number of structures in the data set: 1886

Atomic species in training set      : 2
  Species : H O

Average energy (eV/atom) : -0.375722
Minimum energy (eV/atom) : -1.000000
Maximum energy (eV/atom) : 1.000000

The input and output values have been normalized to [-1.0, 1.0].
Structures outside of this interval will not be used for training.
  Energy scaling factor: 392.531310
  Atomic energy shift  : -25.483040

-----------------------------------------------------------------------
                          Training details
-----------------------------------------------------------------------

Training method         : Limited Memory BFGS

Number of iterations    : 100

Training structures     : 1698
Testing  structures     : 188
```

```
Testing set :

     22       23       34       79      106      107      145      146
    153      154      158      165      172      173      175      186
    189      192      200      201      216      221      222      225
    228      239      242      248      258      263      278      304
    309      369      371      384      399      407      439      446
    448      449      455      480      489      492      493      499
    515      520      523      526      529      530      540      542
    556      557      558      580      591      621      627      642
    705      709      720      741      764      779      799      800
    807      813      835      836      837      843      857      869
    870      871      875      876      879      887      889      898
    905      907      912      916      917      920      951      963
    970      974      988     1003     1010     1032     1050     1075
   1091     1101     1116     1133     1178     1190     1192     1200
   1201     1202     1210     1211     1218     1220     1227     1241
   1254     1258     1264     1269     1293     1304     1305     1313
   1317     1344     1369     1396     1408     1411     1422     1427
   1434     1439     1444     1458     1472     1473     1488     1489
   1498     1515     1519     1537     1547     1551     1556     1561
   1566     1572     1584     1591     1603     1616     1628     1643
   1663     1668     1672     1694     1707     1714     1730     1733
   1739     1742     1745     1752     1772     1774     1779     1789
   1797     1802     1806     1836     1838     1870     1881     1882
   1883     1884     1885     1886

-------------------------------------------------------------------------
                            Training process
-------------------------------------------------------------------------

Weight optimization for 100 epochs using the Limited Memory BFGS method.

Sampling type               : sequential

       |------------TRAIN-----------|  |------------TEST------------|
epoch             MAE          <RMSE>             MAE          <RMSE>
    0    1.033603E-03    1.148417E-03    1.018974E-03    1.122986E-03 <
    1    1.033603E-03    1.148417E-03    1.018974E-03    1.122986E-03 <
    2    2.166109E-03    2.246213E-03    2.196121E-03    2.276394E-03 <
    3    4.565986E-04    6.007754E-04    4.632075E-04    6.097483E-04 <
    4    4.511391E-04    5.939909E-04    4.577195E-04    6.023597E-04 <
    5    4.255899E-04    5.602556E-04    4.319588E-04    5.642586E-04 <
    6    3.902296E-04    5.160038E-04    3.933712E-04    5.186351E-04 <
    7    3.368786E-04    4.488755E-04    3.308854E-04    4.502780E-04 <
    8    3.211041E-04    4.258351E-04    3.236831E-04    4.268577E-04 <
    9    3.136198E-04    4.168514E-04    3.166364E-04    4.184181E-04 <
   10    2.955220E-04    3.892955E-04    2.983596E-04    3.880847E-04 <
```

11	2.887204E-04	3.891358E-04	2.989816E-04	3.927048E-04	<
12	2.829289E-04	3.775348E-04	2.900341E-04	3.794371E-04	<
13	2.802000E-04	3.676670E-04	2.815336E-04	3.669067E-04	<
14	2.755465E-04	3.612800E-04	2.769950E-04	3.605580E-04	<
15	2.647098E-04	3.466123E-04	2.640697E-04	3.479910E-04	<
16	2.595012E-04	3.394793E-04	2.580787E-04	3.414323E-04	<
17	2.370662E-04	3.100946E-04	2.339838E-04	3.123945E-04	<
18	2.337471E-04	3.008693E-04	2.395406E-04	3.055868E-04	<
19	2.268971E-04	2.934815E-04	2.330907E-04	2.982068E-04	<
20	2.248515E-04	2.895063E-04	2.276577E-04	2.921501E-04	<
21	2.223819E-04	2.856329E-04	2.241942E-04	2.870008E-04	<
22	2.173449E-04	2.772653E-04	2.174006E-04	2.757714E-04	<
23	2.083253E-04	2.669012E-04	2.110166E-04	2.682129E-04	<
24	1.998556E-04	2.564870E-04	2.058861E-04	2.647704E-04	<
25	1.959174E-04	2.534612E-04	2.132867E-04	2.686368E-04	<
26	1.936206E-04	2.499747E-04	2.048505E-04	2.635938E-04	<
27	1.937651E-04	2.492560E-04	2.055867E-04	2.625658E-04	<
28	1.928636E-04	2.488269E-04	2.067811E-04	2.647462E-04	<
29	1.925338E-04	2.483930E-04	2.068436E-04	2.643365E-04	<
30	1.914316E-04	2.467108E-04	2.065054E-04	2.631613E-04	<
31	1.869370E-04	2.421160E-04	2.030784E-04	2.606717E-04	<
32	1.829402E-04	2.381880E-04	2.025181E-04	2.614156E-04	<
33	1.804692E-04	2.339805E-04	2.007122E-04	2.572027E-04	<
34	1.795198E-04	2.324865E-04	2.012405E-04	2.569332E-04	<
35	1.792674E-04	2.316891E-04	2.002535E-04	2.551111E-04	<
36	1.781684E-04	2.296759E-04	1.993036E-04	2.530169E-04	<
37	1.751723E-04	2.245031E-04	1.958489E-04	2.485075E-04	<
38	1.722181E-04	2.206138E-04	1.925807E-04	2.467608E-04	<
39	1.703024E-04	2.182084E-04	1.916979E-04	2.470471E-04	<
40	1.691496E-04	2.177356E-04	1.901338E-04	2.484444E-04	<
41	1.690010E-04	2.171057E-04	1.891774E-04	2.464770E-04	<
42	1.688544E-04	2.167164E-04	1.889427E-04	2.455960E-04	<
43	1.683767E-04	2.158630E-04	1.878563E-04	2.441562E-04	<
44	1.675097E-04	2.146245E-04	1.863411E-04	2.423612E-04	<
45	1.665121E-04	2.130464E-04	1.841892E-04	2.403688E-04	<
46	1.658652E-04	2.119906E-04	1.833203E-04	2.391548E-04	<
47	1.648537E-04	2.102443E-04	1.818766E-04	2.367171E-04	<
48	1.642254E-04	2.091858E-04	1.810193E-04	2.357356E-04	<
49	1.637187E-04	2.079052E-04	1.805250E-04	2.342208E-04	<
50	1.630771E-04	2.070710E-04	1.796467E-04	2.329858E-04	<
51	1.605315E-04	2.046810E-04	1.761535E-04	2.282891E-04	<
52	1.590510E-04	2.034341E-04	1.754205E-04	2.286828E-04	<
53	1.578123E-04	2.023113E-04	1.735749E-04	2.256973E-04	<
54	1.573606E-04	2.018453E-04	1.731150E-04	2.245282E-04	<
55	1.570008E-04	2.007736E-04	1.720384E-04	2.223401E-04	<
56	1.558888E-04	1.997868E-04	1.688461E-04	2.185228E-04	<
57	1.550335E-04	1.981869E-04	1.679077E-04	2.177343E-04	<
58	1.538111E-04	1.961046E-04	1.651505E-04	2.153109E-04	<

```
   59    1.513926E-04    1.930885E-04    1.620274E-04    2.134456E-04 <
   60    1.493786E-04    1.902116E-04    1.603575E-04    2.133748E-04 <
   61    1.477363E-04    1.882616E-04    1.594608E-04    2.128924E-04 <
   62    1.472945E-04    1.871904E-04    1.595254E-04    2.123202E-04 <
   63    1.469744E-04    1.868992E-04    1.588538E-04    2.116729E-04 <
   64    1.467176E-04    1.866993E-04    1.588095E-04    2.118256E-04 <
   65    1.464792E-04    1.865064E-04    1.584639E-04    2.114123E-04 <
   66    1.461396E-04    1.861042E-04    1.579258E-04    2.107437E-04 <
   67    1.454881E-04    1.852273E-04    1.570508E-04    2.092243E-04 <
   68    1.438568E-04    1.830345E-04    1.555225E-04    2.056587E-04 <
   69    1.422773E-04    1.809235E-04    1.552719E-04    2.033153E-04 <
   70    1.408547E-04    1.797084E-04    1.518029E-04    1.991498E-04 <
   71    1.399832E-04    1.785291E-04    1.532917E-04    1.997879E-04 <
   72    1.396206E-04    1.775502E-04    1.542046E-04    1.975011E-04 <
   73    1.387268E-04    1.767161E-04    1.543735E-04    1.981247E-04 <
   74    1.380552E-04    1.760245E-04    1.537950E-04    1.974635E-04 <
   75    1.366021E-04    1.745122E-04    1.542513E-04    1.984114E-04 <
   76    1.357735E-04    1.729659E-04    1.495163E-04    1.925038E-04 <
   77    1.338512E-04    1.712325E-04    1.482238E-04    1.906963E-04 <
   78    1.327490E-04    1.702392E-04    1.476157E-04    1.893032E-04 <
   79    1.321737E-04    1.692463E-04    1.468382E-04    1.879966E-04 <
   80    1.313195E-04    1.683857E-04    1.450348E-04    1.859832E-04 <
   81    1.305454E-04    1.673256E-04    1.427661E-04    1.835142E-04 <
   82    1.296041E-04    1.663495E-04    1.427473E-04    1.821443E-04 <
   83    1.288414E-04    1.653386E-04    1.417771E-04    1.813121E-04 <
   84    1.279385E-04    1.636387E-04    1.396086E-04    1.810077E-04 <
   85    1.271749E-04    1.627539E-04    1.388339E-04    1.807320E-04 <
   86    1.261391E-04    1.612894E-04    1.378697E-04    1.804301E-04 <
   87    1.260048E-04    1.611350E-04    1.375876E-04    1.799634E-04 <
   88    1.256679E-04    1.605907E-04    1.372713E-04    1.795006E-04 <
   89    1.247128E-04    1.593181E-04    1.369849E-04    1.786199E-04 <
   90    1.239220E-04    1.584206E-04    1.362135E-04    1.775382E-04 <
   91    1.231302E-04    1.575818E-04    1.349549E-04    1.756548E-04 <
   92    1.227922E-04    1.568080E-04    1.333947E-04    1.744178E-04 <
   93    1.221576E-04    1.560244E-04    1.319948E-04    1.729666E-04 <
   94    1.206247E-04    1.537319E-04    1.293860E-04    1.707281E-04 <
   95    1.193319E-04    1.517209E-04    1.254854E-04    1.679133E-04 <
   96    1.190118E-04    1.508329E-04    1.250284E-04    1.676690E-04 <
   97    1.189787E-04    1.504166E-04    1.244464E-04    1.669604E-04 <
   98    1.183739E-04    1.495744E-04    1.240743E-04    1.665498E-04 <
   99    1.177252E-04    1.489369E-04    1.229352E-04    1.645564E-04 <
  100    1.165869E-04    1.478403E-04    1.207679E-04    1.610750E-04 <

Training finished.

-------------------------------------------------------------------
                        Storing final energies
-------------------------------------------------------------------
```

```
Energies of training structures : energies.train.PROCESS
Energies of testing structures  : energies.test.PROCESS
(Manually concatenate the files from different processes.)

Final MAE of training set  =      0.1 meV/atom
Final MAE of testing set   =      0.1 meV/atom

Final RMSE of training set =      0.1 meV/atom
Final RMSE of testing set  =      0.2 meV/atom

------------------------------------------------------------------------
                          Storing networks
------------------------------------------------------------------------

Saving the H network to file : H.10t-10t.nn
Saving the O network to file : O.10t-10t.nn


                       2021-03-26  01:14:53

========================================================================
                    Neural Network training done.
========================================================================
energies.test.0   H2O.train@        O.10t-10t.nn  train.restart
energies.train.0  H2O.train.scaled  train.in       train.rngstate
H.10t-10t.nn      nn.tar.gz         train.out
```

Once training has completed, the ANN potentials for oxygen and hydrogen are available in `O.10t-10t.nn` and `H.10t-10t.nn`. **Note that ANN potential files are binary and cannot be open in a text editor.**

[12]:
```
%cd /content/
```

```
/content
```

5 Analyzing the training results

Was the training successful? One measure is the performance of the trained potential on the validation set, which is reported for each training epoch on lines that like the following:

```
N  MAE_train  RMSE_train  MAE_val  RMSE_val  <
```

`N` is the training iteration (or *epoch*), `MAE_train` and `RMSE_train` are the *mean absolute error* and *root mean squared error* on the training set, and `MAE_val` and `RMSE_val` are the same for the validation set.

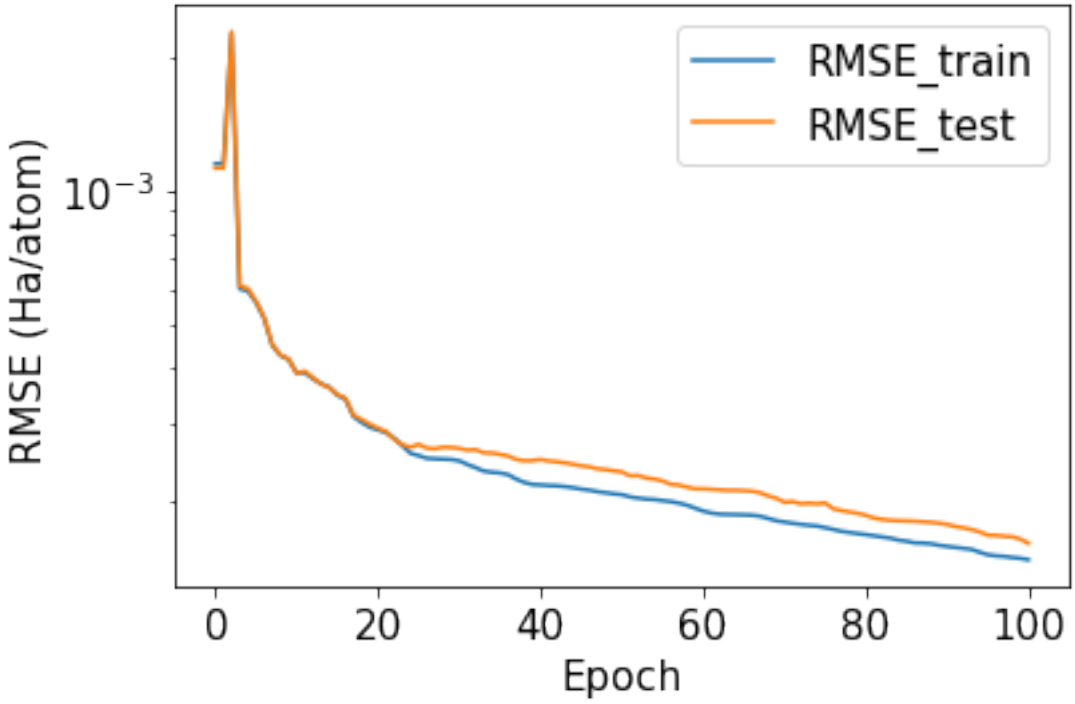
Note: As explained in the companion manuscript, the performance on the validation set is not necessarily a good estimate for the true accuracy of the ANN potential. A more robust accuracy

estimate should be obtained from testing on an independently generated test set.

Plotting the RMSE of both training and validation set together can be helpful to detect overfitting.

Note: As explained above, the energy unit in the water reference data set is Hartree, even though aenet reports electronvolts.

[13]:
```
errors = []
with open("/content/MLP-beginners-guide/tutorial/02_train/train.out") as fp:
  for line in fp:
    if re.match("^ *[0-9].*<$", line):
      errors.append([float(a) for a in line.split()[1:-1]])
errors = np.array(errors)
errors = pd.DataFrame(
    data=errors,
    columns=['MAE_train', 'RMSE_train', 'MAE_test', 'RMSE_test'])
ax = errors[['RMSE_train', 'RMSE_test']].plot(logy=True)
ax.set_xlabel("Epoch"); ax.set_ylabel("RMSE (Ha/atom)")
plt.show()
```

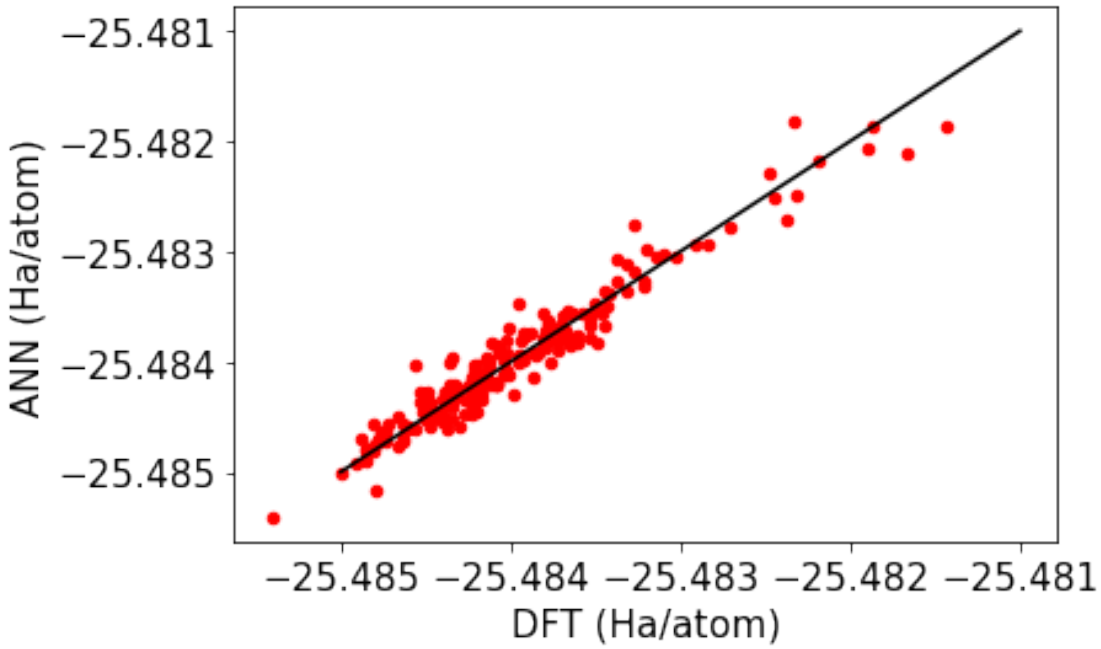


As seen in the figure, the training-set RMSE (`RMSE_train`) continues to decrease until the final training iteration. The validation set RMSE (`RMSE_test`) also decreases but exhibits more noise.

5.1 Comparison of training and validation errors

`train.x` can export the errors for all training samples in the training and validation sets. Here, we visualize the results for one example (set003).

[14]:
```
test = "/content/MLP-beginners-guide/tutorial/02_train/energies.test.0"
test_errors = np.loadtxt(test, skiprows=1, usecols=(3,4))
limits = np.linspace(-25.485, -25.481)
plt.plot(limits, limits, color="black")
plt.ticklabel_format(useOffset=False)
plt.scatter(test_errors[:,0], test_errors[:,1], color="red", s=20,␣
 ↪label="validation")
plt.xlabel('DFT (Ha/atom)')
plt.ylabel('ANN (Ha/atom)')
plt.show()
```



6 04-anetLib-ann-md: The ænet-Python interface

Trained ænet ANN potentials can be used with a number of standard simulations packages. Here we will take a look at the ænet Python interface, which enables compatibility with the Atomic Simulation Environment (ase, see above).

[15]:
```
import aenet
from aenet.ase_calculator import ANNCalculator
```

```
import ase
import ase.spacegroup
import ase.visualize
```

We create a new directory and copy the trained potentials into it.

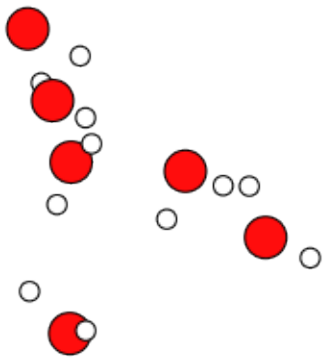
[16]:
```
! mkdir -p /content/MLP-beginners-guide/tutorial/03_md
%cd /content/MLP-beginners-guide/tutorial/03_md
%ls
```

```
/content/MLP-beginners-guide/tutorial/03_md
```

Next, we create an ase structure object from one of the water cluster structures.

[17]:
```
atoms = ase.io.read("/content/MLP-beginners-guide/tutorial/
 ↪water-clusters-BLYP-D3/train_0500_xsf/train0500_1.xsf")
ase.io.write('6h2o.traj', atoms)
ase.io.write('6h2o.png', atoms)
Image("6h2o.png")
#ase.visualize.view(atoms, viewer='x3d')
```

[17]:



The ænet Python package provides an ASE *Calculator* class that can load ænet ANN

potentials. In this case, we use an ANN potential that was trained with the same input file parameters as above but for more iterations.

[18]:
```
pot = {"H": "/content/MLP-beginners-guide/tutorial/02_train/H.10t-10t.nn",
       "O": "/content/MLP-beginners-guide/tutorial/02_train/O.10t-10t.nn"}
calc = ANNCalculator(pot)
```

This calculator can then be used to predict energies (and atomic forces) for a given atomic structure.

[19]:
```
atoms.set_calculator(calc)
E = atoms.get_potential_energy()
print("Total energy = {} Ha".format(E))
```

```
Total energy = -458.68794485712954 Ha
```

[20]:
```
%cd /content
```

```
/content
```

7 Python/ASE Example: Molecular Dynamics Simulation

In this example, we use the ANNCalculator and ASE to perform a molecular dynamics (MD) simulation with constant energy (NVE ensemble).

Note: This is an educational example. MD simulations with ASE in a Google Colab notebook are *not* very efficient. For production simulations, an actual workstation is recommended. Furthermore, ænet also provides a C-compatible library (*ænetLib*) that can be interfaced with other MD software [1].

[1] N. Artrith and A. Urban, Comput. Mater. Sci. 114 (2016) 135-150.

[21]:
```
%cd /content/MLP-beginners-guide/tutorial/03_md
```

```
/content/MLP-beginners-guide/tutorial/03_md
```

The next cell runs the MD simulation. Note that the instructions are purely based on ASE and are not specific to ænet.

The MD simulation is performed in the microcanonical NVE ensemble, i.e., the cell volume and total energy are kept constant. We use a time step of 0.25 femtoseconds for the integration of the integration of motion with the *velocity Verlet* algorithm. Atomic momenta are initialized according to a Maxwell-Boltzmann distribution at a temperature of 300 K.

Note: It can be required to execute the following cell twice.

[23]:
```
from ase.md.velocitydistribution import MaxwellBoltzmannDistribution
from ase.md.verlet import VelocityVerlet
from ase.units import kB, fs
```

```
import aenet

pot = {"H": "/content/MLP-beginners-guide/tutorial/02_train/H.10t-10t.nn",
       "O": "/content/MLP-beginners-guide/tutorial/02_train/O.10t-10t.nn"}

calc = ANNCalculator(pot)

dt = 0.25  # 0.25 fs time step
T = 300.0  # 300 K temperature

md_atoms = ase.io.read("6h2o.traj")
md_atoms.set_calculator(calc)
MaxwellBoltzmannDistribution(md_atoms, T*kB)
dyn = VelocityVerlet(md_atoms, dt*fs)

md_step = 0

def printenergy(atoms=md_atoms, i=[0.0, dt/1000]):
  """ Print status at each MD step. """
  E_pot = atoms.get_potential_energy()/len(atoms)
  E_kin = atoms.get_kinetic_energy()/len(atoms)
  E_tot = E_pot + E_kin
  T = E_kin/(1.5*kB)
  print("{:6.3f}  {:6.1f}  {:15.3f}  {:15.3f}  {:15.3f}".format(
        i[0], T, E_pot, E_kin, E_tot))
  i[0] += i[1]

print("t (ps)   T (K)  E_pot (Ha/atom)  E_kin (Ha/atom)  E_tot (Ha/atom)")
dyn.attach(printenergy, interval=1)
printenergy()
dyn.run(100)
```

```
/root/.local/lib/python3.7/site-packages/ase/md/md.py:48: FutureWarning: Specify
the temperature in K using the 'temperature_K' argument
  warnings.warn(FutureWarning(w))
```

t (ps)	T (K)	E_pot (Ha/atom)	E_kin (Ha/atom)	E_tot (Ha/atom)
0.000	285.4	-25.483	0.037	-25.446
0.000	285.4	-25.483	0.037	-25.446
0.001	285.4	-25.483	0.037	-25.446
0.001	285.3	-25.483	0.037	-25.446
0.001	285.2	-25.483	0.037	-25.446
0.001	285.1	-25.483	0.037	-25.446
0.002	284.9	-25.483	0.037	-25.446
0.002	284.7	-25.483	0.037	-25.446
0.002	284.5	-25.483	0.037	-25.446
0.002	284.2	-25.483	0.037	-25.446
0.003	283.9	-25.482	0.037	-25.446

0.003	283.5	-25.482	0.037	-25.446
0.003	283.2	-25.482	0.037	-25.446
0.003	282.8	-25.482	0.037	-25.446
0.004	282.4	-25.482	0.037	-25.446
0.004	282.0	-25.482	0.036	-25.446
0.004	281.6	-25.482	0.036	-25.446
0.004	281.1	-25.482	0.036	-25.446
0.005	280.7	-25.482	0.036	-25.446
0.005	280.3	-25.482	0.036	-25.446
0.005	279.8	-25.482	0.036	-25.446
0.005	279.3	-25.482	0.036	-25.446
0.006	278.7	-25.482	0.036	-25.446
0.006	278.2	-25.482	0.036	-25.446
0.006	277.6	-25.482	0.036	-25.446
0.006	277.1	-25.482	0.036	-25.446
0.007	276.5	-25.482	0.036	-25.446
0.007	276.0	-25.481	0.036	-25.446
0.007	275.4	-25.481	0.036	-25.446
0.007	274.9	-25.481	0.036	-25.446
0.008	274.4	-25.481	0.035	-25.446
0.008	273.9	-25.481	0.035	-25.446
0.008	273.4	-25.481	0.035	-25.446
0.008	272.9	-25.481	0.035	-25.446
0.009	272.4	-25.481	0.035	-25.446
0.009	272.0	-25.481	0.035	-25.446
0.009	271.5	-25.481	0.035	-25.446
0.009	271.0	-25.481	0.035	-25.446
0.010	270.5	-25.481	0.035	-25.446
0.010	270.0	-25.481	0.035	-25.446
0.010	269.4	-25.481	0.035	-25.446
0.010	268.9	-25.481	0.035	-25.446
0.011	268.4	-25.480	0.035	-25.446
0.011	267.9	-25.480	0.035	-25.446
0.011	267.3	-25.480	0.035	-25.446
0.011	266.8	-25.480	0.034	-25.446
0.012	266.3	-25.480	0.034	-25.446
0.012	265.8	-25.480	0.034	-25.446
0.012	265.3	-25.480	0.034	-25.446
0.012	264.8	-25.480	0.034	-25.446
0.013	264.4	-25.480	0.034	-25.446
0.013	264.0	-25.480	0.034	-25.446
0.013	263.5	-25.480	0.034	-25.446
0.013	263.1	-25.480	0.034	-25.446
0.014	262.7	-25.480	0.034	-25.446
0.014	262.4	-25.480	0.034	-25.446
0.014	262.0	-25.480	0.034	-25.446
0.014	261.6	-25.480	0.034	-25.446
0.015	261.3	-25.480	0.034	-25.446

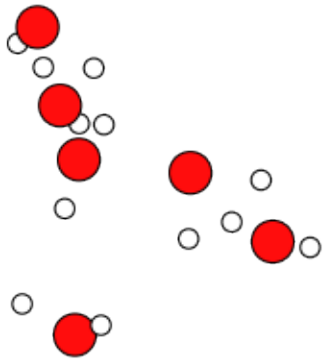
```
0.015   260.9         -25.479          0.034         -25.446
0.015   260.6         -25.479          0.034         -25.446
0.015   260.3         -25.479          0.034         -25.446
0.016   260.0         -25.479          0.034         -25.446
0.016   259.7         -25.479          0.034         -25.446
0.016   259.4         -25.479          0.034         -25.446
0.016   259.1         -25.479          0.033         -25.446
0.017   258.7         -25.479          0.033         -25.446
0.017   258.4         -25.479          0.033         -25.446
0.017   258.1         -25.479          0.033         -25.446
0.017   257.8         -25.479          0.033         -25.446
0.018   257.5         -25.479          0.033         -25.446
0.018   257.2         -25.479          0.033         -25.446
0.018   256.9         -25.479          0.033         -25.446
0.018   256.6         -25.479          0.033         -25.446
0.019   256.3         -25.479          0.033         -25.446
0.019   255.9         -25.479          0.033         -25.446
0.019   255.6         -25.479          0.033         -25.446
0.019   255.3         -25.479          0.033         -25.446
0.020   255.0         -25.479          0.033         -25.446
0.020   254.7         -25.479          0.033         -25.446
0.020   254.3         -25.479          0.033         -25.446
0.020   254.0         -25.479          0.033         -25.446
0.021   253.7         -25.479          0.033         -25.446
0.021   253.4         -25.479          0.033         -25.446
0.021   253.1         -25.478          0.033         -25.446
0.021   252.8         -25.478          0.033         -25.446
0.022   252.6         -25.478          0.033         -25.446
0.022   252.3         -25.478          0.033         -25.446
0.022   252.0         -25.478          0.033         -25.446
0.022   251.8         -25.478          0.033         -25.446
0.023   251.6         -25.478          0.033         -25.446
0.023   251.4         -25.478          0.032         -25.446
0.023   251.2         -25.478          0.032         -25.446
0.023   251.0         -25.478          0.032         -25.446
0.024   250.8         -25.478          0.032         -25.446
0.024   250.7         -25.478          0.032         -25.446
0.024   250.5         -25.478          0.032         -25.446
0.024   250.3         -25.478          0.032         -25.446
0.025   250.2         -25.478          0.032         -25.446
0.025   250.1         -25.478          0.032         -25.446
0.025   249.9         -25.478          0.032         -25.446
0.025   249.8         -25.478          0.032         -25.446
```

[23]: True

We can now take a look at the final structure.

[27]:
```
ase.io.write('6h2o-md.png', md_atoms)
Image("6h2o-md.png")
#ase.visualize.view(md_atoms, viewer='x3d')
```

[27]:



[25]:
```
%cd /content/
```

```
/content
```

8 Caution

This tutorial is purely an educational example, and the constructed ANN potential cannot be expected to be robust. Longer training (more iterations) and extensive testing would be required.