MLPotential Manual

Amsterdam Modeling Suite 2022.1

www.scm.com

Jan 14, 2022

CONTENTS

1 Theory and usage **1**
- 1.1 Quickstart guide . . . 1
- 1.2 Theory of ML potentials . . . 1
- 1.3 Installation and uninstallation . . . 2
- 1.4 Included (pre-parameterized) models . . . 2
- 1.5 Custom models (custom parameters) . . . 3
- 1.6 Backends . . . 5
- 1.7 CPU and GPU (CUDA), parallelization . . . 5
- 1.8 Troubleshooting . . . 6
- 1.9 Support . . . 6
- 1.10 Technical information . . . 6
- 1.11 References . . . 6

2 AMS driver's tasks and properties **7**
- 2.1 Geometry, System definition . . . 7
- 2.2 Tasks: exploring the PES . . . 7
- 2.3 Properties in the AMS driver . . . 7

3 MLPotential Keywords **9**

Index **11**

CHAPTER

ONE

THEORY AND USAGE

The MLPotential engine in the Amsterdam Modeling Suite can calculate the potential energy surface using several different types of machine learning (ML) potentials.

1.1 Quickstart guide

To set up a simple MLPotential job using the graphical user interface, see the ANI-1ccx Thermochemistry tutorial.

1.2 Theory of ML potentials

With machine learning potentials, it is possible to quickly evaluate the energies and forces in a system with close to first-principles accuracy. Machine learning potentials are fitted (trained, parameterized) to reproduce reference data, typically calculated using an ab initio or DFT method. Machine learning potentials are sometimes referred to as machine learning force fields, or as interatomic potentials based on machine learning.

Several types of machine learning potentials exist, for example neural-network-based methods and kernel-based methods.

Several types of **neural network potentials** exist. It is common for such potentials to calculate the total energy as a sum of atomic contributions. In a **high-dimensional neural network potential** (HDNNP), as proposed by Behler and Parrinello[1], each atomic contribution is calculated by means of a feed-forward neural network, that takes in a representation of the chemical environment around the atom as input. This representation, or atomic environment **descriptor** or **fingerprint**, consists of a vector of rotationally, translationally, and permutationally invariant functions known as **atom-centered symmetry functions** (ACSF).

Graph convolutional neural network potentials (GCNNPs), or **message-passing network neural potentials**, similarly construct the total energy by summing up atomic contribution, but the appropriate representations of local atomic chemical environments are learned from the reference data.

Kernel-based methods make predictions based on how similar a system is to the systems in the training set.

There are also other types of machine learning potentials. For more detailed information, see for example references[2] and[3].

[1] J. Behler, M. Parrinello. Phys. Rev. Lett. 98 (2007) 146401 https://doi.org/10.1103/PhysRevLett.98.146401
[2] J. Behler. J. Chem. Phys. 145 (2016) 170901. https://doi.org/10.1063/1.4966192
[3] T. Mueller, A. Hernandez, C. Wang. J. Chem. Phys. 152 (2020) 050902. https://doi.org/10.1063/1.4966192

1.3 Installation and uninstallation

The Amsterdam Modeling Suite by default does not include the Python packages necessary to run the machine learning potential backends.

If you set up an MLPotential job via the **graphical user interface**, you will be asked to install the packages if they have not been installed already when you save your input. You can also use the package manager. Moreover, the run script will also contain instructions to install the necessary packages if needed, so that you can seamlessly run MLPotential jobs also on remote machines.

A **command-line installation tool** can also be used, for instance to install the torchani backend:

```
"$AMSBIN"/amspackages install torchani
```

The packages are installed into the AMS Python environment, and do not affect any other Python installation on the system. For the installation, an internet connection is required, unless you have configured the package manager for offline use .

To **uninstall** a package, e.g. `torchani`, run:

```
"$AMSBIN"/amspackages remove torchani
```

The package manager automatically installs necessary python dependencies such as PyTorch, but if you require a different version you can use pip:

```
"$AMSBIN"/amspython -m pip install -U torch
```

Note: Packages installed through pip alone by the user will not show up as installed in the package manager, but they will be detected and used if possible.

If you install the a package into your amspython environment, using `amspython -m pip install`, the package manager will not display it in its overview. However, it will allow you to make use of it for running calculations with the ML Potential module. If you want to make sure that the version you installed will be detected, you can use

```
$ "$AMSBIN"/amspackages check --pip torch
05-11 10:47:57 torch is not installed!
05-11 10:47:57 User installed version located through pip: torch==1.8.1
```

We won't be able to guarantee that all versions you install yourself work, but you may for instance be able to use this to install GPU supported versions of Torch that match your specific drivers.

1.4 Included (pre-parameterized) models

A **model** is the combination of a functional form with a set of parameters. Three pre-parameterized models can be selected, ANI-2x, ANI-1ccx, and ANI-1x. The predictions from these models are calculated from **ensembles**, meaning that the final prediction is an average over several independently trained neural networks.

Table 1.1: Pre-parameterized models for the MLPotential engine

	ANI-2x	ANI-1ccx	ANI-1x
Functional form	HDNNP	HDNNP	HDNNP
Ensemble size	8	8	8
Atomic environment descriptor	ACSF	ACSF	ACSF
Supported elements	H, C, N, O, F, S, Cl	H, C, N, O	H, C, N, O
Training set structures	organic molecules	organic molecules	organic molecules
Reference method	ωB97-x/6-31G(d)	DLPNO-CCSD(T)/CBS	ωB97-x/6-31G(d)
Backend	TorchANI	TorchANI	TorchANI
Reference	[4]	[5]	[6]

For the ANI-*x models, the standard deviation for the energy predictions are calculated for the "main" output molecule (e.g., the final point of a geometry optimization). The summary statistics can be found in the `mlpotential.txt` file in the `worker.0` subdirectory of the results directory.

`Model`

Type Multiple Choice

Default value ANI-2x

Options [Custom, ANI-1ccx, ANI-1x, ANI-2x]

Description Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X) ANI-1cxx: based on DLPNO-CCSD(T)/CBS

ANI-1x and ANI-1ccx have been parameterized to give good geometries, vibrational frequencies, and reaction energies for gasphase organic molecules containing H, C, O, and N. ANI-2x can also handle the atoms F, S, and Cl.

Set to Custom to specify the backend and parameter files yourself.

1.5 Custom models (custom parameters)

Set `Model` to **Custom** and specify which backend to use with the `Backend` option. In a typical case, you would have used that backend to train your own machine learning potential.

The backend reads the parameters, and any other necessary information (for example neural network architecture), from either a file or a directory. Specify the `ParameterFile` or `ParameterDir` option accordingly, with a path to the file or directory. Read the backend's documentation to find out which option is appropriate.

Some backends may require that an energy unit (`MLEnergyUnit`) and/or distance unit (`MLDistanceUnit`) be specified. These units correspond to the units used during the training of the machine learning potential.

Example:

```
Engine MLPotential
    Backend SchNetPack
    Model Custom
    ParameterFile ethanol.schnet-model
    MLEnergyUnit kcal/mol
    MLDistanceUnit angstrom
EndEngine
```

[4] C. Devereux et al., J. Chem. Theory Comput. 16 (2020) 4192-4202. https://doi.org/10.1021/acs.jctc.0c00121
[5] J. S. Smith et al., Nat. Commun. 10 (2019) 2903. https://doi.org/10.1038/s41467-019-10827-4
[6] J. S. Smith et al., J. Chem. Phys. 148 (2018) 241733. https://doi.org/10.1063/1.5023802

Backend

Type Multiple Choice

Options [PiNN, SchNetPack, sGDML, TorchANI]

Description The machine learning potential backend.

MLDistanceUnit

Type Multiple Choice

Default value Auto

Options [Auto, angstrom, bohr]

GUI name Internal distance unit

Description Unit of distances expected by the ML backend (not the ASE calculator). The ASE calculator may require this information.

MLEnergyUnit

Type Multiple Choice

Default value Auto

Options [Auto, Hartree, eV, kcal/mol, kJ/mol]

GUI name Internal energy unit

Description Unit of energy output by the ML backend (not the unit output by the ASE calculator). The ASE calculator may require this information.

ParameterDir

Type String

Default value

GUI name Parameter directory

Description Path to a set of parameters for the backend, if it expects to read from a directory.

ParameterFile

Type String

Default value

Description Path to a set of parameters for the backend, if it expects to read from a file.

1.6 Backends

Table 1.2: Backends supported by the MLPotential engine.

	PiNN	SchNetPack	sGDML	TorchANI
Reference	[7]	[8]	[9]	[10]
Methods	HDNNPs, GCN-NPs, ...	HDNNPs, GCN-NPs, ...	GDML, sGDML	[ensembles of] HDNNPs
Pre-built models	none	none	none	ANI-1x, ANI-2x, ANI-1ccx
Parameters from	ParameterDir	ParameterFile	ParameterFile	ParameterFile
Kernel-based	No	No	Yes	No
ML framework	TensorFlow 1.15	PyTorch	none, PyTorch	PyTorch

Note: For **sGDML**, the order of the atoms in the input file **must** match the order of atoms which was used during the fitting of the model.

Note: If you use a custom parameter file with **TorchANI**, the model specified via `ParameterFile filename.pt` is loaded with `torch.load('filename.pt')['model']`, such that a forward call should be accessible via `torch.load('filename.pt')['model']((species, coordinates))`. The energy shifter is not read from custom parameter files, so the absolute predicted energies will be shifted with respect to the reference data, but this does not affect relative energies (e.g., reaction energies).

1.7 CPU and GPU (CUDA), parallelization

By default a calculation will run on the CPU, and use all available CPU power. To limit the number of threads, the `NumThreads` keyword can be used if the backend uses PyTorch as its machine learning framework. Alternatively, you can set the environment variable OMP_NUM_THREADS.

To use a CUDA-enabled GPU, ensure that a CUDA-enabled version of TensorFlow or PyTorch has been installed (see *Installation and uninstallation* (page 2)). Then set `Device` to the device on which you would like to run, for example, **cuda:0**. Calculations are typically much faster on the GPU than on the CPU.

`Device`

Type Multiple Choice

Default value

Options [, cpu, cuda:0, cuda:1]

Description Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

`NumThreads`

Type String

[7] Y. Shao et al., J. Chem. Inf. Model. 60 (2020) 1184-1193. https://doi.org/10.1021/acs.jcim.9b00994
[8] K. T. Schütt et al., J. Chem. Theory Comput. 15 (2019) 448-455. https://doi.org/10.1021/acs.jctc.8b00908
[9] S. Chmiela et al. Comp. Phys. Commun. 240 (2019) 38-45. https://doi.org/10.1016/j.cpc.2019.02.007
[10] X. Gao et al. J. Chem. Inf. Model (2020). https://doi.org/10.1021/acs.jcim.0c00451

Default value

GUI name Number of threads

Description Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

Note: Because the calculation runs in a separate process, the number of threads is controlled by the input keyword NumThreads and *not* by the environment variable NSCM. **We recommend setting NSCM=1** when using the MLPotential engine.

Only single-node calculations are currently supported.

1.8 Troubleshooting

If you run a PyTorch-based backend and receive an error message starting with:

```
sh: line 1: 1351 Illegal instruction: 4 sh
```

you may be attempting to run PyTorch on a rather old cpu. You could try to upgrade PyTorch to a newer version:

```
"$AMSBIN"/amspython -m pip install torch -U -f https://download.pytorch.org/whl/torch_
↪stable.html
```

If this does not help, please contact SCM support.

1.9 Support

SCM does not provide support for parameterization using the MLPotential backends. SCM only provides technical (non-scientific) support for running simulations via the AMS driver.

1.10 Technical information

Each of the supported backends can be used as ASE (Atomic Simulation Environment) calculators. The MLPotential engine is an interface to those ASE calculators. The communication between the AMS driver and the backends is implemented with a named pipe interface. The MLPotential engine launches a python script, `ase_calculators.py`, which initializes the ASE calculator. The exact command that is executed is written as `WorkerCommand` in the output.

1.11 References

CHAPTER

TWO

AMS DRIVER'S TASKS AND PROPERTIES

MLPotential is an engine used by the AMS driver. While the specific options for the MLPotential engine are described in this manual, the definition of the system, the selection of the task and certain (potential-energy-surface-related) properties are documented in the AMS driver's manual.

In this page you will find useful links to the relevant sections of the AMS driver's Manual.

2.1 Geometry, System definition

The definition of the system, i.e. the atom types and atomic coordinates (and optionally, the lattice vectors and atomic masses for isotopes) are part of the AMS driver input. See the System definition section of the AMS manual.

2.2 Tasks: exploring the PES

The job of the AMS driver is to handle all changes in the simulated system's geometry, e.g. during a geometry optimization or molecular dynamics calculation, using energy and forces calculated by the engine.

These are the tasks available in the AMS driver:

- GCMC (Grand Canonical Monte Carlo)
- Geometry Optimization
- IRC (Intrinsic Reaction Coordinate)
- Molecular Dynamics
- NEB (Nudged Elastic Band)
- PESScan (Potential Energy Surface Scan, including linear transit)
- Single Point
- Transition State Search
- Vibrational Analysis

2.3 Properties in the AMS driver

The following properties can be requested to the MLPotential engine in the AMS driver's input:

- Elastic tensor

- Hessian
- Nuclear gradients (forces)
- Normal modes
- PES point character
- Phonons
- Stress tensor
- Thermodynamic properties

CHAPTER

THREE

MLPOTENTIAL KEYWORDS

`Backend`

Type Multiple Choice

Options [PiNN, SchNetPack, sGDML, TorchANI]

Description The machine learning potential backend.

`Device`

Type Multiple Choice

Default value

Options [, cpu, cuda:0, cuda:1]

Description Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

`MLDistanceUnit`

Type Multiple Choice

Default value Auto

Options [Auto, angstrom, bohr]

GUI name Internal distance unit

Description Unit of distances expected by the ML backend (not the ASE calculator). The ASE calculator may require this information.

`MLEnergyUnit`

Type Multiple Choice

Default value Auto

Options [Auto, Hartree, eV, kcal/mol, kJ/mol]

GUI name Internal energy unit

Description Unit of energy output by the ML backend (not the unit output by the ASE calculator). The ASE calculator may require this information.

`Model`

Type Multiple Choice

Default value ANI-2x

Options [Custom, ANI-1ccx, ANI-1x, ANI-2x]

Description Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X) ANI-1cxx: based on DLPNO-CCSD(T)/CBS

ANI-1x and ANI-1ccx have been parameterized to give good geometries, vibrational frequencies, and reaction energies for gasphase organic molecules containing H, C, O, and N. ANI-2x can also handle the atoms F, S, and Cl.

Set to Custom to specify the backend and parameter files yourself.

`NumThreads`

Type String

Default value

GUI name Number of threads

Description Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

`ParameterDir`

Type String

Default value

GUI name Parameter directory

Description Path to a set of parameters for the backend, if it expects to read from a directory.

`ParameterFile`

Type String

Default value

Description Path to a set of parameters for the backend, if it expects to read from a file.

INDEX

A

AMS driver, 6, 7
Atoms, 7

C

Charge, 7
Coordinates, 7

E

Elastic tensor, 7

G

GCMC (*Grand Canonical Monte Carlo*), 7
Geometry, 7
Geometry Optimization, 7

H

Hessian, 7

I

IRC (*Intrinsic Reaction Coordinate*), 7
Isotopes, 7

L

Lattice Vectors, 7
Linear Transit, 7

M

Molecular Dynamics, 7

N

NEB (*Nudged Elastic Band*), 7
Nuclear gradients (*forces*), 7

P

PES, 7
PES point character, 7
PESScan (*Potential Energy Surface Scan*), 7
Phonons, 7
Point Charges, 7
Potential Energy Surface, 7

S

Single Point, 7
Stress tensor, 7

T

Task, 7
Thermodynamic properties, 7
Transition State Search, 7

V

Vibrational Analysis, 7

X

xyz, 7