SCM
Software for Chemistry & Materials

MLPotential Manual

Amsterdam Modeling Suite 2023.1

www.scm.com

Mar 31, 2023

CONTENTS

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 What's new in AMS2023.1?

- New model: M3GNet-UP-2022 based on M3GNet. This is a universal potential (UP) that can be used for the entire periodic table of elements up to, but excluding, Curium (Cm, 96).
- New backend: M3GNet
- PiNN is no longer a backend in MLPotential, but you can use it through Engine ASE.

1.2 Quickstart guide

To set up a simple MLPotential job using the graphical user interface, see the

- ANI-1ccx Thermochemistry tutorial
- M3GNet tutorial

1.3 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).

[1] J. Behler, M. Parrinello. Phys. Rev. Lett. 98 (2007) 146401 https://doi.org/10.1103/PhysRevLett.98.146401

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].

1.4 Installation and uninstallation

The Amsterdam Modeling Suite requires the installation of additional Python packages 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. A **command-line installation tool** can also be used, for instance to install the torchani backend:

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

You can use the command line installer to install these packages on a remote system, so that you can seamlessly run MLPotential jobs also on remote machines.

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 AMS package manager for offline use .

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

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

1.4.1 Installing GPU enabled backends using AMSpackages

New in version AMS2023.101.

Various versions of the ML potential packages are available through the AMSpackages, with different system dependencies such as GPU drivers. The option can be selected under the "ML options" menu in the graphical package manager (SCM -> Packages). You can choose from the following options,

- CPU, will install CPU-only backends, including PyTorch and Tensorflow-CPU.
- GPU (Cuda 11.6), will install GPU enabled backends, including Tensorflow, and a CUDA 11.6 specific version of pyTorch.
- GPU (Cuda 11.7), will install GPU enabled backends, including Tensorflow, but will include CUDA 11.7 enabled pyTorch instead.

The default is CPU. Note that this is the only option available under MacOS.

Using the package manager on the the command line or in shell scripts you can use the `--alt` flag, together with one of the options. On the command line the options are denoted as *mlcpu*, *mlcu116* and *mlcu117* respectively. To install GPU enabled versions of the ML potential backends on the command line, for instance using the CUDA 11.7 enabled version of PyTorch:

[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

```
$ "$AMSBIN"/amspackages --alt mlcu117 install mlpotentials
Going to install packages:
nvidia-cuda-runtime-cu11 v[11.7.99] - build:0
tensorflow v[2.9.1] - build:0
All ML Potential backends v[2.0.0] - build:0
torch v[1.13.1+cu117] - build:0
nvidia-cudnn-cu11 v[8.5.0.96] - build:0
M3GNet ML Backend v[0.2.4] - build:0
sGDML Calculator patch v[0.4.4] - build:0
TorchANI Calculator patch v[2.2] - build:0
SchNetPack ML Backend v[1.0.0] - build:0
nvidia-cuda-nvrtc-cu11 v[11.7.99] - build:0
nvidia-cublas-cu11 v[11.10.3.66] - build:0
ANI Models for TorchANI backend v[2.2] - build:0
TorchANI NN module patch v[2.2] - build:0
TorchANI ML backend v[2.2] - build:0
sGDML ML backend v[0.4.4] - build:0
```

Alternatively, to install a single backend for instance torchani:

```
"$AMSBIN"/amspackages --alt mlcu117 install torchani
```

To change the default value, you can set an environment variable `SCM_AMSPKGS_ALTERNATIVES`. For advanced configuration options of the package installation, see also the package manager instructions.

1.4.2 Installing packages using pip

The package manager installs trusted and tested versions of packages from our website, but if you require a different version you can use pip to install packages from https://pypi.org:

```
"$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 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
```

Not all versions of the packages on PyPI work with our ML potential backends.

1.5 Included (pre-parameterized) models

A **model** is the combination of a functional form with a set of parameters. Four pre-parameterized models can be selected: M3GNet-UP-2022 (**U**niversal **P**otential), ANI-2x, ANI-1ccx, and ANI-1x. The predictions from the ANI-* 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

	M3GNet-UP-2022	ANI-2x	ANI-1ccx	ANI-1x
Functional form	NNP	HDNNP	HDNNP	HDNNP
Ensemble size	1	8	8	8
Atomic environment descriptor	m3gnet	ACSF	ACSF	ACSF
Supported elements	H, He, Li, .., Am	H, C, N, O, F, S, Cl	H, C, N, O	H, C, N, O
Training set structures	materials project	organic molecules	organic molecules	organic molecules
Reference method	PBE	ωB97-x/6-31G(d)	DLPNO-CCSD(T)/CBS	ωB97-x/6-31G(d)
Backend	m3gnet	TorchANI	TorchANI	TorchANI
Reference	[4]	[5]	[6]	[7]

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, M3GNet-UP-2022]

Description Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X) ANI-1cxx: based on DLPNO-CCSD(T)/CBS M3GNet-UP-2022: based on DFT (PBE) data.

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.

M3GNet-UP-2022 is a universal potential (UP) for the entire periodic table and has been primarily trained to crystal data (energies, forces, stresses) from the Materials Project.

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

[4] C. Chen, S. P. Ong. Nature Computational Science 2, 718–728 (2022). arXiv.2202.02450 (https://doi.org/10.48550/arXiv.2202.02450).
[5] C. Devereux et al., J. Chem. Theory Comput. 16 (2020) 4192-4202. https://doi.org/10.1021/acs.jctc.0c00121
[6] J. S. Smith et al., Nat. Commun. 10 (2019) 2903. https://doi.org/10.1038/s41467-019-10827-4
[7] J. S. Smith et al., J. Chem. Phys. 148 (2018) 241733. https://doi.org/10.1063/1.5023802

1.6 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
```

`Backend`

Type Multiple Choice

Options [M3GNet, NequIP, 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.

1.7 Backends

Table 1.2: Backends supported by the MLPotential engine.

	M3GNet	SchNetPack	sGDML	TorchANI
Reference	?	[10]	[11]	[12]
Methods	m3gnet	HDNNPs, GCN-NPs, …	GDML, sGDML	[ensembles of] HDNNPs
Pre-built models	M3GNet-UP-2022	none	none	ANI-1x, ANI-2x, ANI-1ccx
Parameters from	ParameterDir	ParameterFile	ParameterFile	ParameterFile
Kernel-based	No	No	Yes	No
ML framework	TensorFlow 2.9.1	PyTorch	none, PyTorch	PyTorch

Note: Starting with AMS2023, PiNN[9] is only supported as a custom Calculator through Engine ASE[8].

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).

1.8 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.

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

`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

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.

1.9 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.

1.10 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.

1.11 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.

1.12 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.

Note: The MLPotential Engine currently only supports 0D (molecules) and 3D (bulk) systems.

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 [M3GNet, NequIP, 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, M3GNet-UP-2022]

Description Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X) ANI-1cxx: based on DLPNO-CCSD(T)/CBS M3GNet-UP-2022: based on DFT (PBE) data.

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.

M3GNet-UP-2022 is a universal potential (UP) for the entire periodic table and has been primarily trained to crystal data (energies, forces, stresses) from the Materials Project.

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