TensorAlloy

TensorAlloy is a TensorFlow based machine learning framework for metal alloys. **TensorAlloy** builds direct computation graph from atomic positions to total energy. Thus, atomic forces and the virial stress tensor can be derived by the **AutoGrad** module of TensorFlow directly.

1. Requirements

- Python>=3.7.0
- TensorFlow==1.13.1
- scikit-learn
- scipy
- numpy
- ase>=3.18.0
- matplotlib>=3.0.0
- toml==0.10.0
- joblib
- pandas

`ase`, `toml` and `joblib` can be installed with `pip` easily. Other packages may be installed with Anaconda3. However, the performance of conda-provided tensorflow may not be that good. Natively compiled TensorFlow, with all CPU features (SSE, AVX, etc.) enabled, is highly recommended.

2. Installation

This package can be installed with `pip` or `python`:

```
cd tensoralloy
pip install .
```

or

```
cd tensoralloy
python setup.py install
```

A command-line interface `tensoralloy` will also be installed.

3. Training

3.1 Usage

TOML is the configuration file format used by TensorAlloy. All the necessary keys are included in the two examples. Default keys and values can be found in default.toml.

To run a training experiment, the easiest way is using the command-line program `tensoralloy`. Here are some key commands:

- `tensoralloy build database [extxyz]`: build a database from an `extxyz` file.
- `tensoralloy run [input.toml]`: run an experiment from a `toml` input file.
- `tensoralloy print [logfile]`: read the evaluation results from a logfile and print these results to a csv file.
- `tensoralloy --help`: print the help messages.

3.2 Output

After the training, a binary `pb` file (the trained model) will be exported to the `model_dir` specified in the input toml file. This exported `pb` file can be used by the ASE-style [`TensorAlloyCalculator`].

We also provide three pre-trained models:

- Ni: energy, force
- Mo: energy, force, stress
- MoNi: energy, force, stress

3.3 Examples

We also provide two training examples, QM7 and SNAP/Mo-Ni. For each example, a `run.sh` is provided. Below are detailed walkthroughs.

3.3.1 QM7

This example starts from the raw `extxyz` file. The first step is building the SQLITE3 database with the following command

```
tensoralloy build database qm7.extxyz --energy-unit=eV
```

The details of the `extxyz` format can be found here. The unit of the energies in the extxyz is specified with the flag `--energy-unit`.

The second step is running the experiment with command

```
tensoralloy run input.toml
```

The angular symmetry functions are already enabled in the input file. Feel free to modify `train.batch_size` and `nn.atomic.behler.angular`.

The last step is printing the evaluation results:

```
tensoralloy print train/logfile
```

With the default input settings, the evaluation results may look like:

```
             Energy/mae  Energy/mae/atom  Energy/mse      loss
global_step
5000           0.288981         0.019927    0.179771  0.407640
10000          0.192857         0.013317    0.099330  0.293619
15000          0.167815         0.011654    0.080911  0.261736
20000          0.166467         0.011610    0.079693  0.260290
25000          0.155102         0.010835    0.072407  0.246700
30000          0.147741         0.010316    0.067687  0.236977
35000          0.144419         0.010074    0.064431  0.231164
40000          0.140643         0.009803    0.060714  0.224816
45000          0.137724         0.009585    0.057666  0.219831
50000          0.134791         0.009389    0.054062  0.214697
```

This command will also generate a `summary.csv` in the `model_dir`.

3.3.2 SNAP/Mo-No

This is a binary alloy dataset which can be obtained from GitHub freely. The original JSON files are converted to three ASE Sqlite3 databases and shipped with this program.

In this example, we only use radial symmetry functions.

To launch this experiment, simply run the command `tensoralloy run input.toml`. This will take approx 2 hours on our workstation (just one GTX 1080 Ti).

3.4 TFRecords

The TensorAlloy program uses `tfrecords` binary files to cache VAP arrays. **One must note that changing the numbers of symmetry function hyper-parameters (eta, gamma, etc.) will also change `v2g_map`, so you should re-build the tfrecords files.**

4. Prediction

The ASE-style calculator, `TensorAlloyCalculator`, shall be used to make predicitons.

```
#!coding=utf-8
""" Simple usage. """
from tensoralloy.calculator import TensorAlloyCalculator
from ase.build import bulk
from ase.units import GPa

calc = TensorAlloyCalculator("NiMo.pb")
atoms = bulk("Ni", cubic=True)
atoms.calc = calc
print(atoms.get_total_energy())
print(atoms.get_forces())
print(atoms.get_stress() / GPa)
```

5. Input

In this section we will introduce the options and values of the input toml file.

5.1 Root

- `precision`: `medium` (float32) or `high` (float64). Default is `medium`.
- `seed`: the global seed. Default is 611.

5.2 Dataset

- `dataset.sqlite3`: a string, the ASE Sqlite3 Database to use.
- `dataset.name`: a string, the name of this experiment.
- `dataset.rc`: a float, the cutoff radius.
- `dataset.tfrecords_dir`: a string, the directory to save/load tfrecords files.
- `dataset.test_size`: an integer, the number of examples for evaluation.
- `dataset.serial`: a boolean. Default is `false`. Typically this should be false so that structures in the sqlite3 database can be written to tfrecords in parallel. For some old Linux systems or debugging, this can be set to `true`.

5.3 NN

5.3.1 General options

- `nn.activation`: a string, the activation function to use. Options are: `elu`, `leaky_relu`, `softplus`, `softsign`, `tanh`, `sigmoid`.
- `nn.minimize`: a list of string, the properties to minimize. Some examples: `["energy", "forces"]`, `["energy", "stress"]`
- `nn.export`: a list of string, the properties that the exported model should predict. This option has the same format with `nn.minimize`.

5.3.2 Loss options

- `nn.loss.energy.weight`: a float, the weight of the energy loss. Default is 1.
- `nn.loss.energy.per_atom_loss`: a boolean. If true, per-atom energy loss will be used. Default is false.
- `nn.loss.energy.method`: a string, `"rmse"` or `"logcosh"`.
- `nn.loss.forces.weight`: a float, the weight of the force loss. Default is 1.
- `nn.loss.forces.method`: a string, `"rmse"` or `"logcosh"`.
- `nn.loss.stress.weight`: a float, the weight of the force loss. Default is 1.
- `nn.loss.stress.method`: a string, `"rmse"` or `"logcosh"`.
- `nn.loss.l2.weight`: a float, the weight of the L2 regularization. Default is 1.0. **Note: adaptive optimizers (adam, nadam, etc) are not compatible with L2. If you use any adaptive optimizer, please set this to 0.**
- `nn.loss.l2.decayed`: a boolean. If True, the L2 weight will decay exponentially.
- `nn.loss.l2.decay_rate`: a float controls the decay rate.
- `nn.loss.l2.decay_steps`: an integer.

5.3.3 Neural network options

- `nn.atomic.arch`: a string, the architecture of the atomistic neural networks. There two options in this package: `"AtomicNN"` or `"AtomicResNN"`. The former is the traditional type and later is the modified version proposed by the TensorAlloy paper.
- `nn.atomic.kernel_initializer`: a string, the initialization method.
- `nn.atomic.minmax_scale`: a boolean. If True, the min-max normalization of the input features will be enabled. Default is true.
- `nn.atomic.resnet.fixed_static_energy`: a boolean. If `nn.atomic.arch` is `AtomicResNet` and this is true, the static energy parameters will be fixed.
- `nn.atomic.behler.eta`: a list of float, the eta values for radial symmetry functions.
- `nn.atomic.behler.omega`: a list of float, the omega values for radial symmetry functions.
- `nn.atomic.behler.beta`: a list of float, the beta values for angular symmetry functions.
- `nn.atomic.behler.gamma`: a list of float, the gamma values for angular symmetry functions.
- `nn.atomic.behler.zeta`: a list of float, the zeta values for angular symmetry functions.

- `nn.atomic.behler.angular`: a boolean. If true, angular symmetry functions will be used. Otherwise `gamma`, `zeta` and `beta` will be ignored.

5.3.4 Layers

The following section demonstrates how to setup the hidden layers of the atomistic neural network for element **C**. This block is optional. The default setting is `[64, 32]`.

```
[nn.atomic.layers]
C = [128, 64, 32]
```

5.4 Opt

- `opt.method`: a string, the optimizer to used. Options are: `adam`, `nadam`, `sgd`, `adadelta`, `rmsprop`. Default is `adam`.
- `opt.learning_rate`: a float, the initial learning rate. Default is 0.01.
- `opt.decay_function`: a string, the decay function to use. Options are: `exponential`, `inverse_time`, `natural_exp`. Default is `false` which means learning rate decay is disabled.
- `opt.decay_rate`: a float, the decay rate.
- `opt.decay_steps`: an integer.
- `opt.staircase`: a boolean.

5.5 Train

- `train.reset_global_step`: a boolean. If True, the global step will be set to 0 by force. If you want to continue a previous training, this should be changed to `false`.

- `train.batch_size`: an integer, the batch size. Default is 50.

- `train.shuffle`: a boolean. If True, the input dataset will be shuffled. In most cases this should be true.

- `train.model_dir`: a string, the working directory for this experiment.

- `train.train_steps`: an integer, the maximum training steps.

- `train.eval_steps`: an integer, the intervals between two evaluations.

- `train.summary_steps`: an integer, the intervals between two writing summary operations.

- `train.log_steps`: an integer, the intervals between two logging operations.

- `train.profile_steps`: an integer, the intervals between two performance profiling operations. Set this to 0 to disable profiling.

- `train.ckpt.checkpoint_filename`: a string, the previous checkpoint file to read. Default is `false`.

- `train.ckpt.use_ema_variables`: a boolean. If `train.ckpt.checkpoint_filename` is provided, the moving average variables will be loaded if this is true.

- `train.ckpt.restore_all_variables`: a boolean. It only works when `train.ckpt.checkpoint_filename` is provided. If this is true, all variables (including the optimizer-specific variables) will be loaded. If false, only model variables will be used. Typically, if you want to continue a previous training with exactly the same settings but more training steps, set `train.reset_global_step` to false and this option to true. If you want to use the checkpoint values as initial guesses, this option shall be false.

6. Licenses

TensorAlloy

This TensorAlloy program is licensed under GNU Lesser General Public License v3.0. For more information please read LICENSE.

QM7

The extxyz file shipped with this program is created from the MATLAB data file downloaded from quantum-machine.org. The QM7 dataset is a subset of GDB-13 (a database of nearly 1 billion stable and synthetically accessible organic molecules) composed of all molecules of up to 23 atoms (including 7 heavy atoms C, N, O, and S), totalling 7165 molecules.

For more information, please read:

1. L. C. Blum, J.-L. Reymond, 970 Million Druglike Small Molecules for Virtual Screening in the Chemical Universe Database GDB-13, J. Am. Chem. Soc., 131:8732, 2009
2. M. Rupp, A. Tkatchenko, K.-R. Müller, O. A. von Lilienfeld: Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning, Physical Review Letters, 108(5):058301, 2012

SNAP

The SNAP database and its subsets SNAP-Ni and SNAP-Mo are published by Shyue Ping Ong. The original JSON files can be obtained from GitHub freely. The original dataset is also licensed under BSD-3.