Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

TorsionNet

Runxuan Jiang RUNXUANJ@UMICH.EDU
Tarun Gogineni TGOG@UMICH.EDU
Josh
Ambuj
Paul
University of Michigan
Ann Arbor, MI 48109, USA

Editor: TBD

Abstract

We propose TorsionNet, an open-source deep reinforcement learning library designed for molecular conformer generation using Python. TorsionNet features several pre-built environments and baseline agents based on state-of-the-art algorithms in the field. The environments and agents are built on a modular interface, allowing users to easily build and test new implementations. Additionally, TorsionNet comes with extensive logging and visualization tools for evaluation of agents and generated conformers, as well as a toolkit for generating and modifying molecules. TorsionNet is well-tested and thoroughly documented, and is available through on PyPi and on Github: https://github.com/ZimmermanGroup/conformer-ml.

Keywords: reinforcement learning, deep learning, deep reinforcement learning, open source, conformer generation, computational chemistry

1. Introduction

There have been many recent developments in the use of deep reinforcement learning applied to computational chemistry tasks (Li et al., 2018; Zhou et al., 2017; Simm et al., 2020). One such task is conformer generation (Gogineni et al., 2020), which involves finding an ensemble of unique low-energy three-dimensional orientations, or conformers, for a given molecule (Ebejer et al., 2012). Efficient and accurate prediction of low-energy conformers is integral to molecular modeling, with wide applications from drug development to 3D QSAR (Cole et al., 2018). However, there currently exists very few open-source software libraries for deep learning applied conformer generation, and the ones that do exist are often not modular enough for further experimentation and modification. Thus, we introduce TorsionNet, a comprehensive and modular Python library for applying deep reinforcement learning to conformer generation, using PyTorch (Paszke et al., 2019) for deep learning and RDKit for chemoinformatic capabilities.

Many libraries already exist that contain benchmarking experiments and baseline implementations for general deep reinforcement learning. For example, OpenAI Gym (Brockman et al., 2016) and bsuite (Osband et al., 2020) both contain implementations of reinforcement learning environments commonly used for benchmarking agents, such as cartpole, mountaincar, and Atari. TorsionNet seeks to fill this role within the conformer generation space by supplying pre-built environments for benchmarking agents, as well as an interface to easily customize environments for further exploration. This allows `TorsionNet`'s environment to be readily modified for other chemoinformatic tasks related to conformer generation, such as protein folding and reaction prediction.

There are also many implementations of baseline deep reinforcement learning agents available, such as Rllib (Liang et al., 2018) and OpenAI baselines (Dhariwal et al., 2017). However, due to the complex nature of molecules which are often represented by graph structures rather than vectors, a large amount of modification and setup work is required to adapt these baseline libraries to work with molecule environments. Within `TorsionNet`, several baseline training algorithms are implemented to accomodate generalized observation spaces, as well as baseline neural networks for molecule inputs built with graph neural network (GNN) components using PyTorch Geometric (Fey and Lenssen, 2019).

Additionally, `TorsionNet` provides extensive logging capabilities and an analysis module for recording and visualizing training results, including conformer-generation specific metrics and visuals.

2. TorsionNet Architecture

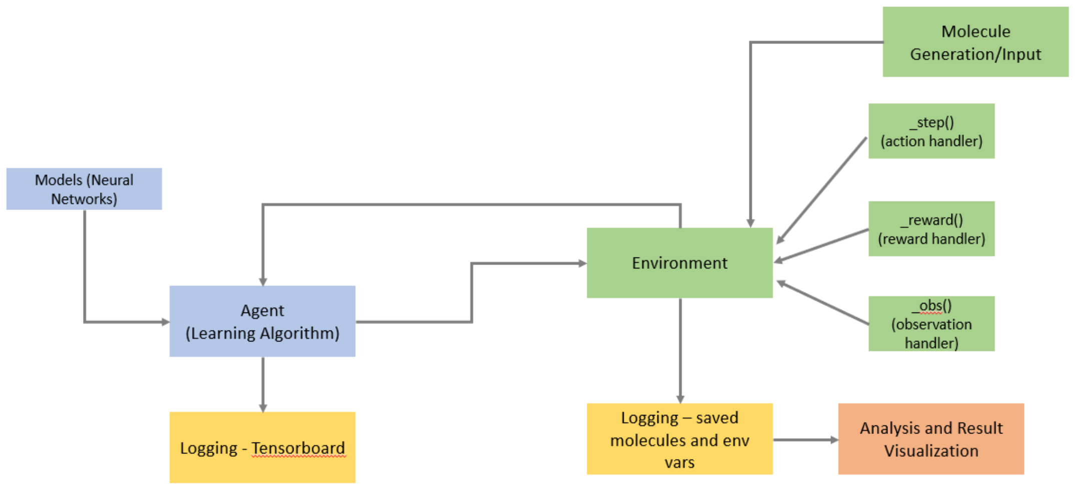
Molecule Generation/Input
_step() (action handler)
Models (Neural Networks)
_reward() (reward handler)
Environment
Agent (Learning Algorithm)
_obs() (observation handler)
Logging - Tensorboard
Logging – saved molecules and env vars
Analysis and Result Visualization

Figure 1: Architecture of `TorsionNet`. Agent components are colored blue, environment components are colored green, and logging/analysis componenets are colored yellow.

2.1 Environments

All `TorsionNet` environments are classes built by overriding methods of the `ConformerEnv` base class. The main methods to be overridden are the `_step(action)` method, which specifies how the environment will modify its state/molecule given an action input, the `_obs()` method, which converts the environment's state into an input for a neural network, and the `_reward()` method, which calculates the reward from the current state of the environment. Other functions such as `_done()`, which determines the end of an episode in the environment, and `reset()` can also be overridden at the user's convenience.

To demonstrate the flexibility of the `ConforomerEnv` interface, `TorsionNet` implements several different mixin classes overriding one of the previously mentioned methods, which can then be mixed and matched to form new environments, some of which are included as pre-built environments in `TorsionNet`, including the environment used in Gogineni et al. (2020).

2.1.1 Molecule Generation

All `TorsionNet` environments are initialized with a `MolConfig` object, which specifies the molecule to be used in the environment as well as any molecule-specific parameters needed by the environment. This allows environments to be compatible with any molecule, including user-generated molecules.

Additionally, `TorsionNet` contains scripts for generating `MolConfig` objects for several simple molecules and molecule benchmarks found in Gogineni et al. (2020), such as branched alkanes, lignin, and more.

`TorsionNet` also includes wrappers for executing multiple environments in parallel.

2.2 Agents

`TorsionNet` contains implementations of several baseline agents, as well as an interface for creating custom agents `BaseAgent`. Agents implement the `step()` method of `BaseAgent`, which correspond to one iteration of interacting with the environment to collect samples and then training on those samples.

Agents are initialized with a `Config` object, which specifies the neural network to be used, as well as training hyperparameters. Agents can also be configured with an evaluation environment, in which the agent will not interact with while training but will be evaluated on the evaluation environment every certain number of steps.

`TorsionNet` implements several baseline built-in agents, including recurrent and non-recurrent implementations of advantage actor critic (A2C) (Wu et al., 2017) and proximal policy optimization (PPO) (Schulman et al., 2017).

2.2.1 Models

`TorsionNet` implements several baseline built-in neural network models, including recurrent and non-recurrent versions of the `TorsionNet` model from (Gogineni et al., 2020) which utilizes a message passing neural network (MPNN) (Gilmer et al., 2017), as well as similar graph networks that utilize graph attention networks (Veličković et al., 2018).

2.3 Logging and Analysis

3. Future Work

pass

4. Conclusions

pass

Acknowledgments

We would like to acknowledge ...

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Jason C. Cole, Oliver Korb, Patrick McCabe, Murray G. Read, and Robin Taylor. Knowledge-based conformer generation using the cambridge structural database. *Journal of Chemical Information and Modeling*, 58(3):615–629, 2018. doi: 10.1021/acs.jcim.7b00697. URL `https://doi.org/10.1021/acs.jcim.7b00697`. PMID: 29425456.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

Jean-Paul Ebejer, Garrett M. Morris, and Charlotte M. Deane. Freely available conformer generation methods: How good are they? *Journal of Chemical Information and Modeling*, 52(5):1146–1158, 2012. doi: 10.1021/ci2004658. URL `https://doi.org/10.1021/ci2004658`. PMID: 22482737.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL `http://proceedings.mlr.press/v70/gilmer17a.html`.

Tarun Gogineni, Ziping Xu, Exequiel Punzalan, Runxuan Jiang, Joshua Kammeraad, Ambuj Tewari, and Paul Zimmerman. Torsionnet: A reinforcement learning approach to sequential conformer search. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20142–20153. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/e904831f48e729f9ad8355a894334700-Paper.pdf`.

Yanjun Li, Hengtong Kang, Ketian Ye, Shuyu Yin, and Xiaolin Li. Foldingzero: Protein folding from scratch in hydrophobic-polar model, 2018.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/liang18b.html`.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygf-kSYwH`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Gregor Simm, Robert Pinsler, and Jose Miguel Hernandez-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8959–8969. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/simm20b.html`.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/361440528766bbaaaa1901845cf4152b-Paper.pdf`.

Zhenpeng Zhou, Xiaocheng Li, and Richard N. Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS Central Science*, 3(12):1337–1344, 2017. doi: 10.1021/acscentsci.7b00492. URL `https://doi.org/10.1021/acscentsci.7b00492`. PMID: 29296675.

Appendix A.