Conformer-RL

Runxuan Jiang runxuanj@umich.edu
Tarun Gogineni tgog@umich.edu
Joshua Kammeraad joshkamm@umich.edu
Yife He
Ambuj Tewari tewaria@umich.edu
Paul Zimmerman paulzim@umich.edu
University of Michigan
Ann Arbor, MI 48109, USA

Editor: TBD

Abstract

We present `Conformer-RL`, an open-source deep reinforcement learning library designed for molecular conformer generation using Python. The library contains modular interfaces for environments and agents, allowing users to easily build and test new implementations. Also included are several pre-built pre-built environments and agents based on state-of-the-art algorithms in the field to serve as baselines. Additionally, `Conformer-RL` comes with extensive logging and visualization tools for evaluation of agents and generated conformers, as well as a toolkit for generating and modifying molecules. `Conformer-RL` 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

Recently, much progress has been made in the application of deep reinforcement learning to tasks in computational chemistry (Li et al., 2018; Zhou et al., 2017; Simm et al., 2020). One task where deep reinforcement learning showed promising results 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 reinforcement learning applied conformer generation, and the ones that do exist are often not modular enough for further experimentation and modification. To address this issue, we introduce `Conformer-RL`, 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, mountain-car, and Atari. `Conformer-RL` seeks to fill this role within the conformer generation space by supplying pre-built environments for benchmarking agents. Additionally, `Conformer-RL` includes an environment interface to easily customize environments for further exploration. This allows `Conformer-RL`'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 `Conformer-RL`, we include a general agent base class for building agents compatible with conformer generation environments, as well as several baseline reinforcement learning algorithms. Also included are baseline implementations neural network architectures for molecule inputs built with graph neural network (GNN) components using PyTorch Geometric (Fey and Lenssen, 2019).

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

2. Conformer-RL Architecture

A visual representation of the architecture can be seen in figure 1.

2.1 Environments

`Conformer-RL` environments are classes built on top of the `ConformerEnv` base class. The methods in `ConformerEnv` are implemented in a way so that each method corresponds to a different component within the environment, and is independent to the behavior of the other components. The main components include:

- **Action Handler** is implemented by overriding the `_step(action)` method. The action handler determines how the molecule is modified given an incoming action. For example, one of the pre-built action handlers takes in an array of length equal to the number of torsions in the molecule, and sets the torsion angle of each torsion to the corresponding array element in degrees.

- **Reward Handler** is implemented by overriding the `_reward()` method. The reward handler returns a reward based on the current configuration of the molecule. An example is the Gibbs score from Gogineni et al. (2020).

- **Observation Handler** is implemented by overriding the `_obs()` method. The observation handler returns an observation object representing the current configuration of the molecule and is a compatible input for the neural network of the agent. An example is representing the molecule as a PyTorch Geometric graph where the nodes contain the three-dimensional coordinates for each atom, and the edges represents bonds between atoms.

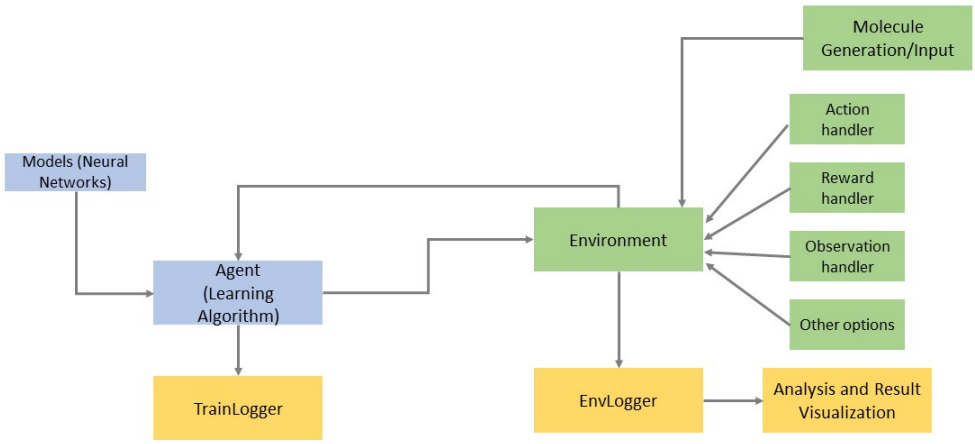

Figure 1: Architecture of `Conformer-RL`. Agent components are colored blue, environment components are colored green, and logging/analysis componenets are colored yellow.

Other class methods of `ConformerEnv` can also be overridden at the user's convenience. For example, if the user would like to keep track of the number of conformers generated with an energy below a certain threshold for each episode, they may add a member variable `self.confs_below_threshold = 0` to the `reset()` method, and then update `self.confs_below_threshold` within the action handler.

For each environment component, `Conformer-RL` implements several pre-built options in the form of mixin classes. Due to the flexibility of the design of `ConformerEnv`, different handlers can be mixed and matched to form unique environments, and new environments for tasks related to conformer generation, such as protein folding and chemical reaction optimization, can be easily built by implementing custom versions of the components.

`Conformer-RL` also includes wrappers for executing multiple environments in parallel.

2.1.1 Molecule Generation

`Conformer-RL` environments are designed to configurable with different molecules, including user-generated ones. Environments are initizlied with a `MolConfig` object, which specifies the RDKit molecule to be used in the environment and any molecule-specific parameters.

For convenience, `Conformer-RL` 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.

2.2 Agents and Models

Agents in `Conformer-RL` are classes inheriting from the `BaseAgent` base class. Agents are initialized with a `Config` object, which specifies the neural network model to be used, training environment, optimize function, and other hyperparameters. Agents can also be configured with an evaluation environment, in which the agent will only evaluate on but not train on. The base class also handles logging. Custom agents can be created by overriding the `step()` method of `BaseAgent`, which corresponds to one iteration of interacting with the environment to collect samples and then training on those samples.

`Conformer-RL` implements several pre-built 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). The pre-built agents also serve as examples for building custom agents, as they are implemented on top of the `BaseAgent` base class.

Implementations of several neural network architectures, including recurrent and non-recurrent versions of the `Conformer-RL` 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

The `Conformer-RL` library contains two types of loggers:

- `TrainLogger` is designed to record information from the agent during training, such as (but not limited to) total reward per episode, training loss, runtime, and memory used. `TrainLogger` supports logging data directly to TensorBoard (Abadi et al., 2015), where the data can be visualized in real time and downloaded if needed.
- `EnvLogger` is designed to record environment information across a single episode, such as (but not limited to) the conformers generated, and the energy of each generated conformer. `EnvLogger` supports saving the per-episode data as a `.pickle` file, as well as saving each generated molecule conformer separately as a `.mol` file. The saved files can then be loaded into a notebook and visualized by `Conformer-RL`'s analysis toolkit.

Both loggers are integrated into the `BaseAgent` and `ConformerEnv` interfaces, and are readily accessible when writing new agents or environments.

`Conformer-RL` contains an analysis toolkit for calculating conformer-related metrics and visualizing results. The toolkit is designed to be used in a notebook and uses seaborn (Waskom, 2021) to generate figures and molecule visualization tools to generate interactive three-dimensional visuals for molecule conformers.

3. Usage and Documentation

`Conformer-RL` provides a detailed API reference for all functions, classes, and non-private class methods in the library. We also provide eight example scripts for training pre-built agents on pre-built environments, and a jupyter notebook showing example usage of the analysis toolkit on logged data. The quick start section of the documentation covers how to run the training scripts, switch between prebuilt environments and agents, and tuning

hyperparameters. Finally, we provide a guide on how to build a custom environment from scratch, and train an agent on the environment.

4. Conclusion

`Conformer-RL` is a comprehensive library with tools for each step of the pipeline for training and testing deep reinforcement learning agents in the conformer generation task, including generating molecules and building environments, building and training agents, and logging/visualizing results. `Conformer-RL`'s modular interfaces and accessible APIs can increase research reproducibility and stimulate discovery in conformer generation.

Acknowledgments

We thank the NSF grant *RTG: Understanding dynamic big data with complex structure (DMS 1646108).*

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

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.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

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.