Conformer-RL: A Deep Reinforcement Learning Library for Conformer Generation

Runxuan Jiang[1] runxuanj@umich.edu
Tarun Gogineni[1] tgog@umich.edu
Joshua Kammeraad[2,3] joshkamm@umich.edu
Yifei He[1] heyifei@umich.edu
Ambuj Tewari[1,2] tewaria@umich.edu
Paul Zimmerman[3] paulzim@umich.edu

[1]*Department of EECS, University of Michigan, USA*
[2]*Department of Statistics, University of Michigan, USA*
[3]*Department of Chemistry, University of Michigan, USA*

Editor: TBD

Abstract

We present `Conformer-RL`, an open-source deep reinforcement learning library for molecular conformer generation using Python. The library contains modular interfaces for environments and agents, allowing users to easily build and test new implementations. We also include several pre-built models and agents using state-of-the-art algorithms to serve as performance 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 on PyPi and Github: `https://github.com/ZimmermanGroup/conformer-rl`.

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

1. Introduction

Several recent works have applied 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 has shown 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). Thus, there is a need for a modular and extensible open-source software library for deep reinforcement learning applied to conformer generation. In this paper, 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 (Landrum, 2016) for chemoinformatic capabilities.

Many libraries already exist that contain benchmarking experiments 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 used for benchmarking agents. `Conformer-RL` seeks to fill this role within the conformer generation space by supplying pre-built environments for benchmarking on this task. Additionally, `Conformer-RL` includes an interface to easily customize environments for further exploration within conformer generation and for custom tasks like reaction prediction.

There are also many implementations of baseline deep reinforcement learning agents available, such as RLlib (Liang et al., 2018). However, since the feature rich graphical representation of molecules is often unsupported out of the box in fixed-vector-state-based libraries, a large amount of modification and setup work is required to adapt these baseline libraries to work with molecular 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.

Finally, `Conformer-RL` provides analysis and logging modules for recording and visualizing training results, including conformer-generation specific metrics and visuals.

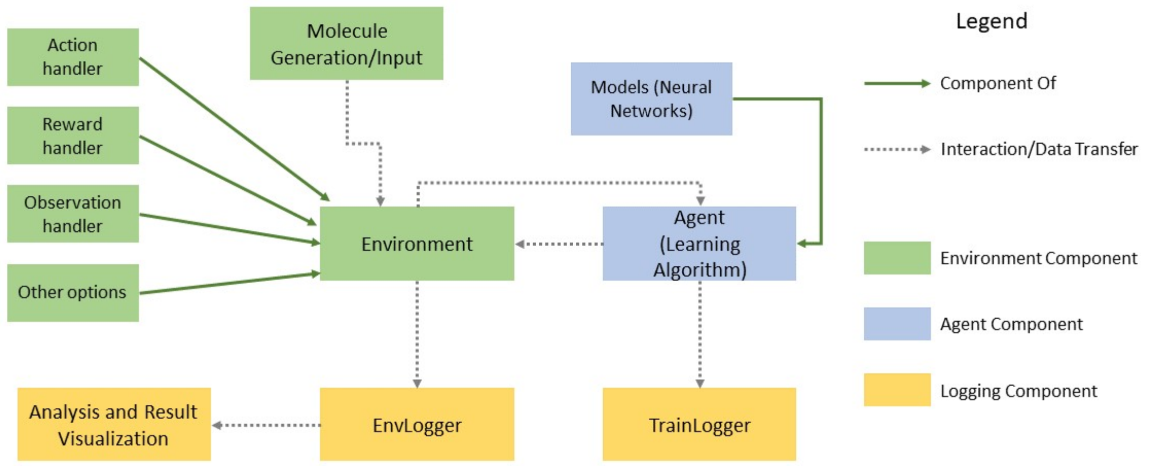


Figure 1: Architecture of `Conformer-RL`.

2. Conformer-RL Architecture

A visual representation of the architecture of `Conformer-RL` can be seen in Figure 1.

2.1 Environments

`Conformer-RL` environments are built on top of the `ConformerEnv` base class. Methods in `ConformerEnv` correspond to different components within the environment, each of which are independent to the behavior of other components. The main components include:

- **Action Handler** determines how the molecule or molecule structure is modified given an incoming action. `Conformer-RL` provides action handlers for setting molecule conformation torsion angles for both discrete and continuous action spaces.

- **Reward Handler** returns a scalar reward based on the current configuration of the molecule. `Conformer-RL` includes several reward implementations derived from the energy of the current molecule conformoration, such as (but not limited to):
  - **Basic Energy Reward** - basic reward that is inversely correlated with the energy of the current conformer.
  - **Pruning Energy Reward** - reward that "prunes" (returns a 0 reward) any conformer generated from an action already seen in the current episode.
  - **Gibbs Score Reward** - the reward used in Gogineni et al. (2020), which prunes conformers based on a TFD (torsional fingerprint deviation) derived metric.
- **Observation Handler** returns a graph representation for the current molecule conformation. `Conformer-RL` contains several methods for extracting features from molecules and converting a conformation into a PyTorch Geometric graph structure, such as:
  - **Node Feature Extractors** - extracts information about atoms in a molecule which can be included in the nodes of the graph representation, such as atom element and three-dimensional coordinates.
  - **Edge Feature Extractors** - extracts information from molecules that can be represented as edges in the graph reperesentation, with options for including bonds between atoms, bond type, Euclidean distances, and more.
  - **Graph Normalizers** - normalizes the graph representation of molecules in terms of translation, rotation, and/or scale.

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.

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 variations 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 be configurable with different molecules, including user-generated ones. The molecular structure is specified as an RDKit molecule object. As a mature cheminformatics library, RDKit offers a standard means of representing and manipulating molecules and can directly interface files from a number of computational chemistry packages including MOL files. RDKit is also at the core of a broad ecosystem of other cheminformatics packages, including Open Babel, which has extensive conversion capabilities for over 100 formats (O'Boyle et al., 2011).

Environments are initialized 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. Molecule generation scripts utilize several libraries depending on the molecule, including stk (Turcani et al., 2021), stko (Steven Bennett, 2021) and Lignin-KMC (Orella et al., 2019), with options for varying molecule size and structure. The currently available conformer generating tools were developed under the assumption of independently varying torsions. This model is sufficient for linear and branched molecules as well as molecules with rigid ring structures. Support for flexible ring structures, in which ring torsions may not vary independently, would be a valuable future contribution.

2.2 Agents and Models

Agents in `Conformer-RL` inherit from the `BaseAgent` class. Each agent is initialized with a `Config` object, which specifies the neural network model to be used, training environment, optimizer function, and other hyperparameters. 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 agents using state of the art algorithms, including advantage actor critic (A2C) (Wu et al., 2017) and proximal policy optimization (PPO) (Schulman et al., 2017). The software also includes baseline implementations of several neural network architectures compatible with molecular inputs, including versions of the model from (Gogineni et al., 2020). The networks are built using graph neural network components using PyTorch Geometric (Fey and Lenssen, 2019) and are compatible with molecules of variable size.

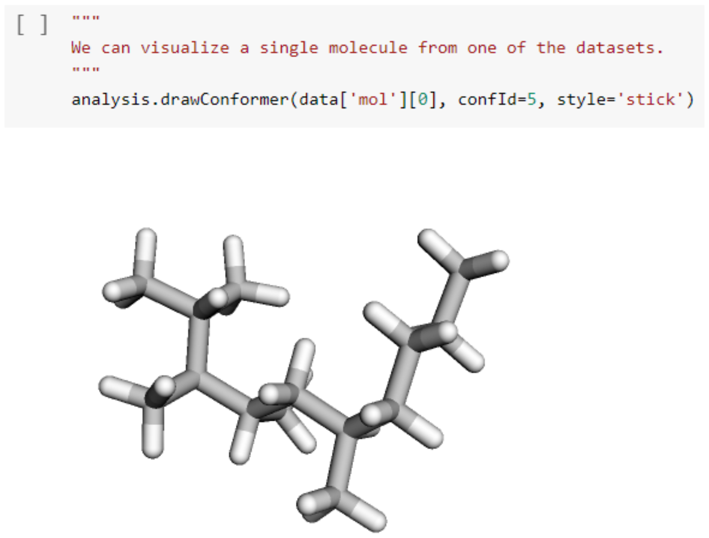
2.3 Logging and Analysis

The `Conformer-RL` library contains two types of loggers. `TrainLogger` logs information from the agent during training, such as total reward per episode, training loss, runtime, etc., and supports logging data directly to TensorBoard (Abadi et al., 2015), where the data can be visualized in real time. `EnvLogger` records environment information across a single environment interaction/episode, such as the conformers generated and conformer energies. `EnvLogger` supports saving the per-episode data and each generated molecule conformer as a MOL file. 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 and visualizing results. The toolkit is designed to be used in a notebook and provides convenience methods for generating figures, charts, and interactive 3D visuals for molecule conformers (Figure 2).

3. Conclusion

`Conformer-RL` is a comprehensive library for training and testing deep reinforcement learning agents in the conformer generation task. `Conformer-RL`'s modular interfaces can increase research reproducibility and stimulate discovery in conformer generation. Full documentation can be found at `https://conformer-rl.readthedocs.io/en/latest/`.



```
[ ] """
    We can visualize a single molecule from one of the datasets.
    """
    analysis.drawConformer(data['mol'][0], confId=5, style='stick')
```

Figure 2: Example of using the toolkit to visualize a generated conformer in Jupyter.

Acknowledgments

We acknowledge the support of NSF via grants DMS-1646108, IIS-2007055, and CHE-1551994.

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.

Jean-Paul Ebejer, Garrett M. Morris, and Charlotte M. Deane. Freely available conformer generation methods: How good are they? *Journal of Chemical Information and Model-*

ing, 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.

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.

Greg Landrum. Rdkit: Open-source cheminformatics software. 2016. URL https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

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.

Noel M. O'Boyle, Michael Banck, Caig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open babel: An open chemical toolbox. *J Cheminform*, 3(33), 2011. doi: https://doi.org/10.1186/1758-2946-3-33. URL https://doi.org/10.1186/1758-2946-3-33.

Michael J. Orella, Terry Z. H. Gani, Josh V. Vermaas, Michael L. Stone, Eric M. Anderson, Gregg T. Beckham, Fikile R. Brushett, and Yuriy Román-Leshkov. Lignin-kmc: A toolkit for simulating lignin biosynthesis. *ACS Sustainable Chemistry & Engineering*, 7(22): 18313–18322, 2019. doi: 10.1021/acssuschemeng.9b03534. URL https://doi.org/10.1021/acssuschemeng.9b03534.

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`.

Lukas Turcani Steven Bennett, Andrew Tarzia. stko. `https://github.com/JelfsMaterialsGroup/stko`, 2021.

Lukas Turcani, Andrew Tarzia, Filip T. Szczypiński, and Kim E. Jelfs. stk: An extendable python framework for automated molecular and supramolecular structure assembly and discovery. *The Journal of Chemical Physics*, 154(21):214102, 2021. doi: 10.1063/5.0049708. URL `https://doi.org/10.1063/5.0049708`.

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.