

# ABLkit: a Python toolkit for abductive learning

Yu-Xuan HUANG<sup>1,2</sup>, Wen-Chao HU<sup>1,2</sup>, En-Hao GAO<sup>1,2</sup>, Yuan JIANG (✉)<sup>1,2</sup>

<sup>1</sup> National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

<sup>2</sup> School of Artificial Intelligence, Nanjing University, Nanjing 210023, China

© Higher Education Press 2024

## 1 Introduction

The integration of machine learning and logical reasoning has long been considered a holy grail problem in artificial intelligence. Recent years have seen considerable attention and significant progress in this field. **AB**ductive **L**earning (ABL) [1,2] is a groundbreaking paradigm that integrates machine learning and logical reasoning in a unified framework. In ABL, the learning model learns to convert data into primitive logic facts, which serve as inputs for logical reasoning. The reasoning part, which employs abductive reasoning, utilizes a given knowledge base to infer the truth value of these logic facts, for updating the learning model. The above process is iterative, enabling learning and reasoning to work together in a balanced loop.

In this paper, we introduce ABLkit, an open-source Python toolkit for ABL. ABLkit provides a comprehensive framework that covers the entire ABL workflow, including data loading, learning model development, reasoning module construction, and bridging between learning and reasoning.

The key features of ABLkit include:

- **High flexibility** ABLkit's flexible architecture ensures compatibility with widely-used learning and reasoning libraries, including scikit-learn [3], PyTorch [4], and SWI-Prolog [5]. This adaptability enables users to leverage diverse machine learning modules and logical reasoning components within their ABL systems.
- **Easy-to-Use interface** ABLkit offers easy-to-use APIs that simplify the development process. Providing data, model, and knowledge, users can build ABL systems efficiently, often with minimal coding effort.
- **Optimized performance** ABLkit incorporates advanced engineering optimizations like caching mechanisms and abstract data interfaces. These enhancements ensure superior performance and accelerated training speed.

The source code is available at the website of [github.com/AbductiveLearning/ABLkit](https://github.com/AbductiveLearning/ABLkit), along with extensive

documentation and examples. For convenient installation, users can directly install it from PyPI by executing the command `pip install ABLkit`.

## 2 The ABLkit toolkit

The overview of ABLkit is shown in Fig. 1, which consists of four modules: Data, Learning, Reasoning, and Bridge.

*Data* module efficiently manages data storage, operations, and evaluations. It includes implementations of abstract data interface, which is responsible for transferring data between various components of ABLkit and ensures a unified and versatile interface. It also contains a series of evaluation metrics for assessing performance.

*Learning* module concentrates on creating, training, and utilizing machine learning models. It provides a standardized wrapper with a unified interface for learning models. This module is fully compatible with popular toolkits like “scikit-learn” or “PyTorch”. Additionally, it provides flexible interfaces, enabling users to freely customize their models according to specific needs and preferences.

*Reasoning* module focuses on constructing knowledge base and performing logical reasoning. Users can conveniently customize a domain knowledge base by overriding the `logic_forward` method, or import knowledge expressed in Prolog language. Similarly, reasoning techniques can be customized. The module's *Reasoner* class then minimizes inconsistency between knowledge and data, using the ZOOpt library [6] or alternative methods.

*Bridge* module integrates the above three parts. Providing data, model, and knowledge, users can execute the ABL process with a single line of code using its `train` method.

When using the ABLkit, it involves the following steps: preparing datasets, building the learning part, building the

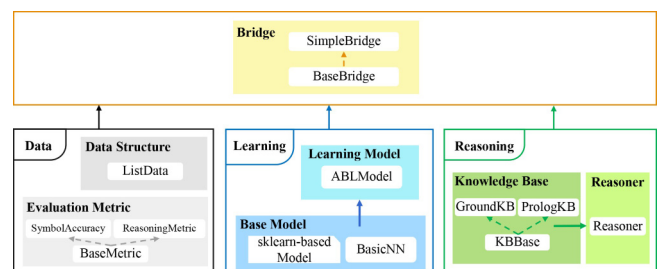


Fig. 1 An overview of ABLkit

Received January 19, 2024; accepted May 13, 2024

E-mail: [jiangy@lamda.nju.edu.cn](mailto:jiangy@lamda.nju.edu.cn)

Special Column—Code & Data

reasoning part, and bridging learning and reasoning. Users first prepare data into the desired format, and construct the learning model with a standard wrapper. Next, the reasoning part is set up by creating a knowledge base and passing it to the *Reasoner*. Finally, the ABL process is executed with the methods of *Bridge*. Users can adjust the hyper-parameters within each module for optimized performance. For advanced customization, they could override the existing methods in the toolkit.

### 3 Experiments

We compare ABLkit with DeepProbLog [7], DeepStochLog [8], and NGS [9] on two benchmark datasets, MNIST Addition [7] and HWF [9]. These selected methods represent the neuro-symbolic approaches but differ from ABL. Specifically, they necessitate the adoption of neural networks as the learning model, and restrict their reasoning modules to specific languages, such as probabilistic logic, stochastic definite clause grammar, or context-free grammar. ABLkit's high flexibility enables it to use any learning model and reasoning module suitable for specific tasks, and enables learning and reasoning to work together in a mutually beneficial way.

The results are shown in Table 1. In the HWF dataset, DeepProbLog fails to converge within a day, resulting in low accuracy. By effectively integrating machine learning and logical reasoning in a balanced loop, coupled with engineering optimizations, ABLkit demonstrates superior performance in terms of predictive accuracy, training time efficiency, and memory usage. Detailed experimental results are available in ABLkit's documentation.

**Table 1** Performance comparison of different methods within a 1-day time limit. “—” means no official implementation for the dataset. The units for time and memory are seconds (s) and megabytes (MB), respectively

Method	Addition			HWF		
	Acc.	Time	Mem.	Acc.	Time	Mem.
DeepProbLog	97.1	2045	3521	31.7	timeout	4315
DeepStochLog	97.5	257	3545	97.9	43098	4355
NGS	—	—	—	98.4	338	3705
ABLkit	<b>98.1</b>	<b>47</b>	<b>2482</b>	<b>99.2</b>	<b>77</b>	<b>3074</b>

### 4 Conclusion

In this paper, we present ABLkit, a Python toolkit for abductive learning. It offers high flexibility, easy-to-use interface, and optimized performance, promising to facilitate both academic research and practical applications in this field.

**Acknowledgements** This research was supported by JiangsuSF (BK20232003). The authors thank Wang-Zhou Dai and Hao-Yuan He for helpful discussions.

**Competing interests** The authors declare that they have no competing interests or financial conflicts to disclose.

### References

1. Zhou Z H. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Sciences*, 2019, 62(7): 76101
2. Zhou Z H, Huang Y X. Abductive learning. In: Hitzler P, Sarker M K, eds. *Neuro-Symbolic Artificial Intelligence: The State of the Art*. Amsterdam: IOS Press, 2022, 353–369
3. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 2011, 12(85): 2825–2830
4. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. PyTorch: an imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems* 32. 2019, 8024–8035
5. Wielemaker J, Schrijvers T, Triska M, Lager T. SWI-prolog. *Theory and Practice of Logic Programming*, 2012, 12(1–2): 67–96
6. Liu Y R, Hu Y Q, Qian H, Qian C, Yu Y. ZOOpt: a toolbox for derivative-free optimization. *Science China Information Sciences*, 2022, 65(10): 207101
7. Manhaeve R, Dumancic S, Kimmig A, Demeester T, De Raedt L. DeepProbLog: neural probabilistic logic programming. In: *Advances in Neural Information Processing Systems* 31. 2018, 3753–3763
8. Winters T, Marra G, Manhaeve R, De Raedt L. DeepStochLog: neural stochastic logic programming. In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. 2022, 10090–10100
9. Li Q, Huang S, Hong Y, Chen Y, Wu Y N, Zhu S C. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, 5884–5894