

# DeepRiver: A Deep Learning Library for Data Streams

Cedric Kulbach<sup>1</sup>, Lucas Cazonelli<sup>1</sup>, Hoang-Anh Ngo<sup>2</sup>, Max Halford<sup>2</sup>, and Saulo Martiello Mastelini<sup>3</sup>

<sup>1</sup> FZI Research Center for Information Technology, Karlsruhe, Germany <sup>2</sup> AI Institute, University of Waikato, Hamilton, New Zealand <sup>3</sup> University of Sao Paulo, Brazil ¶ Corresponding author

DOI: 10.xxxxxx/draft

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

## Summary

Machine learning algorithms enhance and expedite decision-making processes based on available data. However, as data evolves over time, it becomes crucial to adapt machine learning (ML) systems incrementally to accommodate new data patterns. This adaptation is achieved through online learning or continuous ML technologies. Although deep learning technologies have demonstrated outstanding performance on predefined datasets, their application to online, streaming, and continuous learning scenarios has been limited.

DeepRiver([GitHub - Online-ML/Deep-River — Github.com](#)) is a Python package for deep learning on data streams. Built on top of River([Montiel et al., 2021](#)) and PyTorch([Paszke et al., 2017](#)), it offers a unified API for both supervised and unsupervised learning. Additionally, it provides a suite of tools for preprocessing data streams and evaluating deep learning models.

## Statement of need

In today's rapidly evolving landscape, machine learning (ML) algorithms play a pivotal role in shaping decision-making processes based on available data. The acceleration facilitated by these algorithms necessitates constant adaptation to evolving data structures, as what is valid at one moment may no longer hold in the future. To address this imperative, adopting online learning and continuous ML technologies becomes paramount. While deep learning technologies have demonstrated exceptional performance on static, predefined datasets, their application to dynamic and continuously evolving data streams remains underexplored. The absence of widespread integration of deep learning into online, streaming, and continuous learning scenarios hampers the full potential of these advanced algorithms in real-time decision-making ([Kulbach et al., 2024](#)). The emergence of the DeepRiver([GitHub - Online-ML/Deep-River — Github.com](#)) Python package fills a critical void in the field of deep learning on data streams. Leveraging the capabilities of River([Montiel et al., 2021](#)) and PyTorch([Paszke et al., 2017](#)), DeepRiver offers a unified API for both supervised and unsupervised learning, providing a seamless bridge between cutting-edge deep learning techniques and the challenges posed by dynamic data streams. Moreover, the package equips practitioners with essential tools for data stream preprocessing and the evaluation of deep learning models in dynamic, real-time environments. This was already made use of in the context of Streaming Anomaly Detection ([Cazonelli & Kulbach, 2022](#)). As the demand for effective and efficient adaptation of machine learning systems to evolving data structures continues to grow, the integration of DeepRiver into the landscape becomes crucial. This package stands as a valuable asset, unlocking the potential for deep learning technologies to excel in online, streaming, and continuous learning scenarios. The need for such advancements is evident in the quest to harness the full power of machine learning in dynamically changing environments, ensuring our decision-making processes remain accurate, relevant, and agile in the face of evolving data landscapes.

## Related Work

Online machine learning involves updating models incrementally as new data arrives, rather than retraining models from scratch. Several frameworks and libraries have been developed to support this paradigm:

- scikit-multiflow ([Montiel et al., 2018](#))
  - Python-based Library: Inspired by the Java-based MOA framework, designed for streaming data and online learning in Python.
  - Key Features:
    - \* Supports algorithms like Hoeffding Trees, online bagging, and boosting.
    - \* Includes concept drift detection (e.g., ADWIN, Page-Hinkley) to adapt to changing data distributions.
    - \* Stream generators and evaluators for real-time data simulation and model assessment.
  - Limitations: Focuses mainly on traditional machine learning methods, with limited support for deep learning architectures.
- creme ([Halford et al., 2020](#))
  - Lightweight Online Learning: Specialized in incremental learning where models are updated per instance, leading to efficient, low-latency model training.
  - Provides a unified API with a broad range of online learning algorithms, making it the go-to library for streaming data analysis in Python. Limitations: Primarily supports feature-based models with limited capabilities for deep neural networks.

In 2020, creme merged with scikit-multiflow to create River, combining the strengths of both frameworks.

- Massive Online Analysis (MOA) ([Bifet et al., 2010](#))
  - Java-based Pioneer: One of the earliest frameworks dedicated to stream mining and online learning, widely used in academic research.
  - Key Features:
    - \* Introduced foundational algorithms like Hoeffding Trees, Adaptive Random Forest (ARF), and several drift detection techniques (e.g., DDM, EDDM).
    - \* Excellent scalability for handling high-throughput data streams in real-time.
    - \* Strong focus on concept drift adaptation, making it robust in non-stationary environments.
- capyMOA ([CapyMOA &#X2014; Capymoa.org](#))
  - Python Interface for MOA: capyMOA serves as a bridge between the Java-based MOA framework and Python, allowing users to leverage MOA's powerful streaming algorithms within Python workflows.
  - \* Key Features:
    - Enables access to MOA's core functionalities (e.g., Hoeffding Trees, Adaptive Random Forest) from Python.
    - Facilitates hybrid workflows by integrating MOA's Java algorithms with Python's machine learning libraries.
    - Useful for Python developers looking to use MOA's advanced stream mining capabilities without switching ecosystems.

scikit-multiflow and creme (River) focus on efficient online learning in Python, mainly for traditional machine learning algorithms. MOA offers extensive tools for stream mining but lacks deep learning support and Python compatibility. While capyMOA provides Python accessibility to MOA, capyMOA is limited by the underlying Java infrastructure and lacks a natural integration with PyTorch's deep learning ecosystem.

DeepRiver ([GitHub - Online-ML/Deep-River — Github.com](#)) differentiates itself by integrating deep learning capabilities directly into streaming data workflows, enabling continuous learning for neural network models. This addresses a critical gap left by existing frameworks, which are

93 predominantly focused on non-deep learning models.

## 94 Features

95 DeepRiver([GitHub - Online-MI/Deep-River — Github.com](#)) enables the usage of deep learning  
96 models for data streams. This means that deep learning models need to adapt to changes  
97 within the evolving data stream (Bayram et al., 2022; Lu et al., 2018) i.ex. the number  
98 of classes might change over time. In addition to the integration of PyTorch(Paszke et al.,  
99 2017) into River(Montiel et al., 2021), this package offers additional data stream specific  
100 functionalities such as class incremental learning or specific optimizers for data streams.

## 101 Compatibility

102 DeepRiver([GitHub - Online-MI/Deep-River — Github.com](#)) is built on the unified application  
103 programming interface (API) of River(Montiel et al., 2021) that seamlessly integrates both  
104 supervised and unsupervised learning techniques. Further, it integrates the huge functionality  
105 of PyTorch(Paszke et al., 2017) for deep learning such as using GPU acceleration and a  
106 broad range of architectures. This unified approach simplifies the development process  
107 and facilitates a cohesive workflow for practitioners working with dynamic data streams.  
108 Leveraging the capabilities of the well-established River(Montiel et al., 2021) library and  
109 the powerful PyTorch(Paszke et al., 2017) framework, DeepRiver combines the strengths of  
110 these technologies to deliver a robust and flexible platform for deep learning on data streams.  
111 This foundation ensures reliability, scalability, and compatibility with state-of-the-art machine  
112 learning methodologies. It provides comprehensive [documentation](#) to guide users through  
113 the installation, implementation, and customization processes. Additionally, a supportive  
114 community ensures that users have access to resources, discussions, and assistance, fostering a  
115 collaborative environment for continuous improvement and knowledge sharing.

## 116 Adaptivity

117 DeepRiver([GitHub - Online-MI/Deep-River — Github.com](#)) is specifically designed to cater to  
118 the requirements of online learning scenarios. It enables continuous adaptation to evolving data  
119 by supporting incremental updates and learning from new observations in real time, a critical  
120 feature for applications where data arrives sequentially. Further, it enables the model to adapt  
121 to changes in the number of classes over time for classification tasks. It equips practitioners  
122 with tools for evaluating the performance of deep learning models on data streams. This  
123 feature is crucial for ensuring the reliability and effectiveness of models in real-time applications,  
124 enabling users to monitor and fine-tune their models as the data evolves.

## 125 Architecture

126 The DeepRiver library is structured around various types of estimators for anomaly detection,  
127 classification, and regression. Each category contains several specialized classes that inherit  
128 from more general base classes, forming a hierarchical structure. In anomaly detection, the  
129 base class `AnomalyScaler` has derived classes `AnomalyMeanScaler`, `AnomalyMinMaxScaler`,  
130 and `AnomalyStandardScaler`. Additionally, the `Autoencoder` class, which inherits from  
131 `DeepEstimator`, has a specialized subclass called `ProbabilityWeightedAutoencoder`. The  
132 `RollingAutoencoder` class inherits from `RollingDeepEstimator`.

133 For classification, the base class `Classifier` inherits from `DeepEstimator`. Derived from  
134 `Classifier` are specific classes like `LogisticRegression` and `MultiLayerPerceptron`. The  
135 `RollingClassifier` class inherits from both `RollingDeepEstimator` and `Classifier`.

136 In regression, the base class `Regressor` inherits from `DeepEstimator`. Specific regression  
137 classes like `LinearRegression` and `MultiLayerPerceptron` inherit from `Regressor`. The

MultiTargetRegressor also inherits from DeepEstimator. The RollingRegressor class inherits from both RollingDeepEstimator and Regressor.

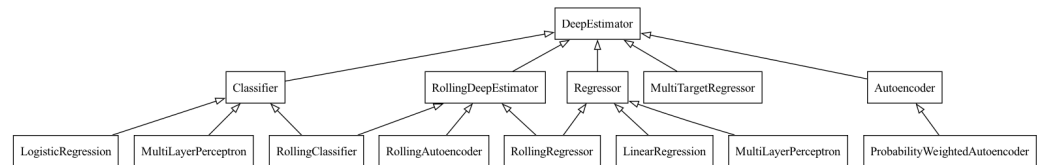


Figure 1: Architecture of DeepRiver

Overall, the library is organized to provide a flexible and hierarchical framework for different types of machine learning tasks, with a clear inheritance structure connecting more specific implementations to their base classes.

## Acknowledgements

This work was carried out with the support of the Research Center for Information Technology in Karlsruhe, Germany.

## References

- Bayram, F., Ahmed, B. S., & Kassler, A. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245, 108632. <https://doi.org/10.1016/j.knosys.2022.108632>
- Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *J. Mach. Learn. Res.*, 11, 1601–1604.
- CapyMOA &#x2014; CapyMOA — capymoa.org. <https://capymoa.org>.
- Cazonelli, L., & Kulbach, C. (2022). Detecting anomalies with autoencoders on data streams. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 258–274. [https://doi.org/10.1007/978-3-031-26387-3\\_16](https://doi.org/10.1007/978-3-031-26387-3_16)
- GitHub - online-ml/deep-river — github.com. <https://github.com/online-ml/deep-river>.
- Halford, M., Bolmier, G., Sourty, R., Vaysse, R., & Zouitine, A. (2020). *creme, a Python library for online machine learning* (Version 0.6.1). <https://github.com/MaxHalford/creme>
- Kulbach, C., Cazonelli, L., Ngo, H.-A., Le-Nguyen, M.-H., & Bifet, A. (2024). *A retrospective of the tutorial on opportunities and challenges of online deep learning*. <https://doi.org/10.48550/arXiv.2405.17222>
- Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363. <https://doi.org/10.1109/TKDE.2018.2876857>
- Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H. M., Read, J., Abdessalem, T., & Bifet, A. (2021). River: Machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110), 1–8. <http://jmlr.org/papers/v22/20-1380.html>
- Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *J. Mach. Learn. Res.*, 19(1), 2915–2914.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. *NIPS-w*.