The Journal of Open Source Software

DeepRiver: A Deep Learning Library for Data Streams

Cedric Kulbach [1*¶], **Lucas Cazzonelli** [1*], **Hoang-Anh Ngo** [2*¶], **Max Halford** [3], and **Saulo Martiello Mastelini** [4]

1 FZI Research Center for Information Technology, Karlsruhe, Germany **2** AI Institute, University of Waikato, Hamilton, New Zealand **3** Carbonfact, Paris, France **4** Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.07226

Software

- Review
- Repository
- Archive

Editor: Taher Chegini

Reviewers:

- @musabgultekin
- @atanikan

Submitted: 04 August 2024
Published: 06 January 2025

Summary

Machine learning algorithms enhance decision-making efficiency by leveraging 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 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. These algorithms, while accelerating analysis, require continuous adaptation to dynamic data structures, as patterns may evolve rapidly. 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 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. Such functionality has been applied to Streaming Anomaly Detection (Cazzonelli & 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:
    - Introduces 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 Developers, 2024)
  - 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` 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 predominantly focused on non-deep learning models.

Features

DeepRiver enables the usage of deep learning models for data streams. This means that deep learning models need to adapt to changes within the evolving data stream (Bayram et al., 2022; Lu et al., 2018) e.g. the number of classes might change over time. In addition to the integration of PyTorch (Paszke et al., 2017) into River (Montiel et al., 2021), this package offers additional data stream specific functionalities such as class incremental learning or specific optimizers for data streams.

Compatibility

DeepRiver is built on the unified application programming interface (API) of River(Montiel et al., 2021) that seamlessly integrates both supervised and unsupervised learning techniques. Additionally, it incorporates PyTorch's (Paszke et al., 2017) extensive functionality for deep learning such as using GPU acceleration and a broad range of architectures. This unified approach simplifies the development process and facilitates a cohesive workflow for practitioners working with dynamic data streams. Leveraging the capabilities of the well-established River(Montiel et al., 2021) library and the powerful PyTorch(Paszke et al., 2017) framework, DeepRiver combines the strengths of these technologies to deliver a robust and flexible platform for deep learning on data streams. This foundation ensures reliability, scalability, and compatibility with state-of-the-art machine learning methodologies, with comprehensive documentation guiding users through the installation, implementation, and customization processes. Additionally, a supportive community ensures that all DeepRiver's users have access to resources, discussions, and assistance, fostering a collaborative environment for continuous improvement and knowledge sharing.

Adaptivity

DeepRiver is specifically designed to cater to the requirements of online learning scenarios. It enables continuous adaptation to evolving data by supporting incremental updates and learning from new observations in real time, a critical feature for applications where data arrives sequentially. Moreover, it allows the model to dynamically adjust to changes in the number of classes over time for classification tasks. It equips practitioners with tools for evaluating the performance of deep learning models on data streams. This feature is crucial for ensuring the reliability and effectiveness of models in real-time applications, enabling users to monitor and fine-tune their models as the data evolves.

Architecture

The DeepRiver library is structured around various types of estimators for anomaly detection, classification, and regression. In anomaly detection, the base class AnomalyScaler has derived classes AnomalyMeanScaler, AnomalyMinMaxScaler, and AnomalyStandardScaler. Additionally, the Autoencoder class, which inherits from DeepEstimator, has a specialized subclass called ProbabilityWeightedAutoencoder. The RollingAutoencoder class inherits from RollingDeepEstimator.

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

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

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

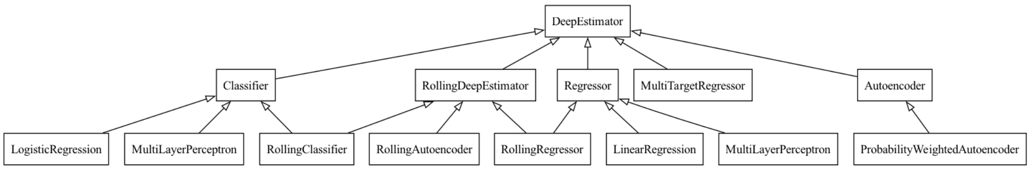

Figure 1: Architecture of DeepRiver

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

Hoang-Anh Ngo received an External Study Awards from the AI Institute, University of Waikato, Hamilton, New Zealand for research on online machine learning under the supervision of Prof. Albert Bifet.

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. *Journal of Machine Learning Research*, *11*(52), 1601–1604. http://jmlr.org/papers/v11/bifet10a.html

CapyMOA Developers. (2024). *CapyMOA — capymoa.org*. https://capymoa.org.

Cazzonelli, 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

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., Cazzonelli, 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. *Journal of Machine Learning Research*, *19*(72), 1–5. http://jmlr.org/papers/v19/18-251.html

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*. https://api.semanticscholar.org/CorpusID:40027675