The Journal of Open Source Software

Balsa: A Fast C++ Random Forest Classifier with Commandline and Python Interface

Tobias Borsdorff [1¶], **Denis de Leeuw Duarte** [2], **Joris van Zwieten** [2], **Soumyajit Mandal** [1¶], **and Jochen Landgraf** [1]

1 SRON Netherlands Institute for Space Research, The Netherlands **2** Jigsaw B.V., The Netherlands ¶ 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Published: unpublished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).

DRAFT

Summary

A Random Forest classifier is a widely used machine learning method. It builds on the strengths of decision trees (Pedregosa et al., 2011) which are simple, intuitive models to classify input data into subsets based on the values of specific features. They form hierarchical structures with decision nodes leading to prediction (Breiman, 2001). Combining the output of multiple decision trees improves its predictive accuracy, reduces the risk of overfitting, and effectively identifies outliers within data sets. Balsa is a highly efficient C++ implementation of the Random Forest classifier concept, built with a focus on runtime and memory performance as key design priorities. Balsa provides multithreaded and distributed training capabilities, allowing users to scale machine learning processes across multiple computing cores or distributed systems by training separate Random Forests and combining them at the end. It supports both binary classification tasks and multi-label classification, expanding its use for a variety of machine learning challenges. Furthermore, Balsa includes comprehensive tools to assess the importance of features, prediction performance metrics, and statistical analysis, allowing users to gain insight into their data and model performance. To ensure ease of use, Balsa uses a compact and storage-efficient binary format to save trained Random Forests. This allows models to be quickly reloaded for future predictions. Designed to fit seamlessly into existing C++ development workflows, Balsa can be easily integrated into custom machine learning pipelines. Alternatively, users can employ Balsa through a command line interface or via a versatile Python interface, the latter can be easily installed with pip. This design provides flexibility for users working in diverse programming environments and across a wide range of machine learning use cases. Balsa performance, flexibility, and ease of integration make it a reliable choice for researchers and developers who require fast, scalable, and efficient Random Forest implementation for a variety of machine-learning tasks. The Balsa package is an open-source software hosted on GitHub and is accompanied by a comprehensive user guide. This guide includes detailed installation instructions and multiple examples that demonstrate various use cases (T. Borsdorff et al., 2024).

Statement of need Balsa has been developed by SRON Netherlands Institute for

Space Research and Jigsaw B.V., The Netherlands,in support of the operational processing of Copernicus Sentinel-5 Precursor (S5P) methane data (Lorente et al., 2021, 2023). The processing requires strict clearing of the measurement data regarding cloud interference, leading to a classification problem (Tobias Borsdorff, Martinez-Velarte, et al., 2024). During the beta phase of software development, we utilized the scikit-learn (sklearn) implementation (Pedregosa et al., 2011) of the Random Forest classifier concept. Integrating into an operational processing framework required a C++ implementation with improved run-time and memory efficiency. This led to the development of Balsa, which is designed to build on the sklearn implementation, addressing the performance demands of the satellite mission. Balsa is fully operational within ESA's data processing framework (Tobias Borsdorff, Mandal, et al., 2024a, 2024b). Future applications of Balsa will focus on the key challenge of managing multiple Random Forests simultaneously in memory to optimize efficiency. In addition, Balsa will play a crucial role in supporting the upcoming near real-time S5P methane product, which requires fast and efficient data classification to meet strict processing time constraints. Although Balsa was developed for S5P methane processing, it is designed to be a Random Forest classifier independent of any specific application. It serves as a universal machine learning toolbox that can be applied in a variety of use cases beyond the S5P data processing. Its flexibility, high performance, and ease of integration make it an invaluable tool for any application that requires efficient and scalable Random Forest-based machine learning.

We conducted a performance analysis and comparison of the Balsa implementation against the Python-based SKLearn library and the C++-based Ranger implementation. This comparison, detailed in the Balsa ATBD (Tobias Borsdorff, Mandal, et al., 2024a), focuses on both memory usage and runtime during the training and prediction phases of the Random Forest Classifier (RFC). We found that the accuracy of the prediction across all three implementations (Balsa, SKLearn, and Ranger) is essentially the same, ensuring that any observed differences in performance are due to optimizations in memory usage and runtime, rather than model accuracy. Figure Figure 1 illustrates that Balsa outperforms both SKLearn and Ranger in terms of memory usage during both the training and prediction phases. Specifically, Balsa consistently shows a lower memory footprint making it particularly advantageous for handling larger datasets. Figure Figure 1 presents the total runtime for RFC training and prediction. As shown, Balsa exhibits a comparable runtime to SKLearn and Ranger during training. However, Balsa excels in the prediction phase, where it delivers superior performance in terms of wall-clock time. This is particularly significant, as prediction speed is the final product in many machine learning workflows, making Balsa a promising choice for operational integration.

DRAFT

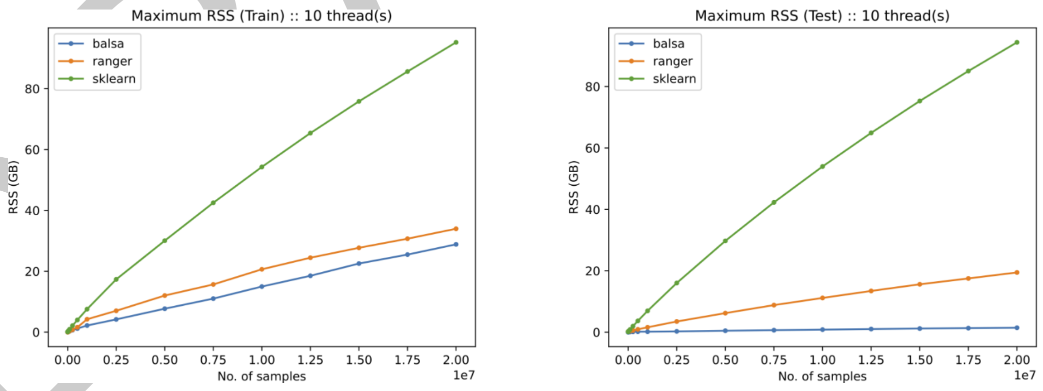

Figure 1: Memory usage during RFC training (left) and prediction (right) for SKLearn (green), Ranger (orange), and Balsa (blue) as a function of dataset size.

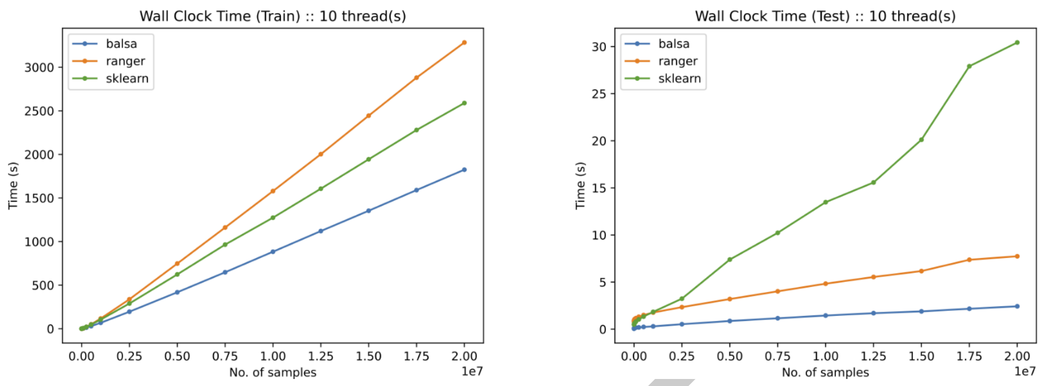

Figure 2: Runtime of RFC training (left) and prediction (right) as a function of dataset size.

Acknowledgements

Balsa development was funded by the European Space Agency.

References

Borsdorff, T., Leeuw Duarte, D. de, Zwieten, J. E. van, & Landgraf, J. (2024). Balsa: A fast c++ random forest classifier. In *GitHub repository*. GitHub. https://github.com/borsdorff/Balsa

Borsdorff, Tobias, Mandal, S., Martinez Velarte, M., Barr, A., & Landgraf, J. (2024a). *Algorithm theoretical baseline document for sentinel-5 precursor: TROPOMI CH4 random forest cloud filter*. https://doi.org/10.5281/zenodo.14186320

Borsdorff, Tobias, Mandal, S., Martinez Velarte, M., Barr, A., & Landgraf, J. (2024b). *Product user manual for the random forest classifier (RFC) c++ implementation to be used for TROPOMI CH4* (Version 1.0.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.14186406

Borsdorff, Tobias, Martinez-Velarte, M. C., Sneep, M., Linden, M. ter, & Landgraf, J. (2024). Random forest classifier for cloud clearing of the operational TROPOMI XCH4 product. *Remote Sensing*, *16*(7). https://doi.org/10.3390/rs16071208

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Lorente, A., Borsdorff, T., Butz, A., Hasekamp, O., Brugh, J. aan de, Schneider, A., Wu, L., Hase, F., Kivi, R., Wunch, D., Pollard, D. F., Shiomi, K., Deutscher, N. M., Velazco, V. A., Roehl, C. M., Wennberg, P. O., Warneke, T., & Landgraf, J. (2021). Methane retrieved from TROPOMI: Improvement of the data product and validation of the first 2 years of measurements. *Atmospheric Measurement Techniques*, *14*(1), 665–684. https://doi.org/10.5194/amt-14-665-2021

Lorente, A., Borsdorff, T., Martinez-Velarte, M. C., & Landgraf, J. (2023). Accounting for surface reflectance spectral features in TROPOMI methane retrievals. *Atmospheric Measurement Techniques*, *16*(6), 1597–1608. https://doi.org/10.5194/amt-16-1597-2023

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, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

DRAFT