The Journal of Open Source Software

PyMilo: A Python Library for ML I/O

AmirHosein Rostami [1,2¶], **Sepand Haghighi** [1], **Sadra Sabouri** [1,3], and **Alireza Zolanvari** [1]

1 Open Science Lab **2** University of Toronto, Toronto, Canada **3** University of Southern California, Los Angeles, United States ¶ Corresponding author

DOI: 10.21105/joss.08858

Software

- Review
- Repository
- Archive

Editor: Abhishek Tiwari

Reviewers:

- @idoby
- @ag-chirag
- @rakeshcusat

Submitted: 03 July 2025
Published: 20 December 2025

Summary

PyMilo is an open-source Python package that addresses the limitations of existing machine learning (ML) model storage formats by providing a transparent, reliable, end-to-end, and safe method for exporting and deploying trained models. Current tools rely on black-box or executable formats that obscure internal model structures, making them difficult to audit, verify, or safely share. Meanwhile, tensor-centric formats such as Safetensors (Hugging Face, 2025) securely store and transfer numerical tensors but do not capture the internal and structural composition of classical machine-learning models (e.g., scikit-learn pipelines), which remain PyMilo's primary focus. Others apply structural transformations during export that may degrade predictive performance and reduce the model to a limited inference-only interface. In contrast, PyMilo serializes models in a transparent human-readable format that preserves end-to-end model fidelity and enables reliable, safe, and interpretable exchange. Here, transparent refers to the ability to inspect model internals through a human-readable structure without execution, and end-to-end fidelity denotes that a model exported and re-imported with PyMilo retains the exact same signature, functionality, parameters, and internal structure as the original, ensuring complete behavioral and structural equivalence. This package is designed to make the preservation and reuse of trained ML models safer, more interpretable, and easier to manage across different stages of the ML workflow (Figure 1).

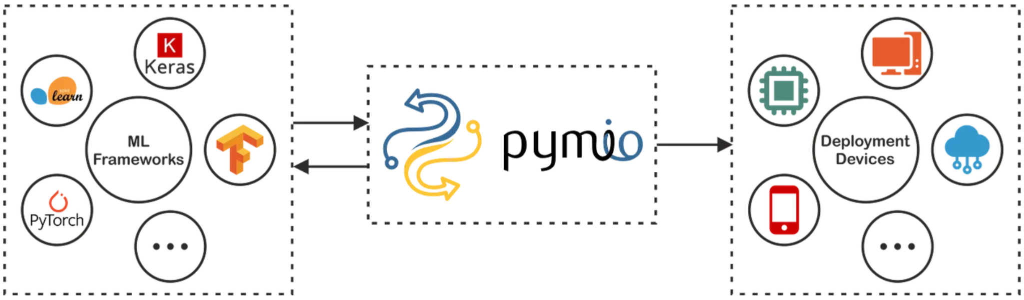

Figure 1: PyMilo is an end-to-end, transparent, and safe solution for transporting models from machine learning frameworks to the target devices. PyMilo preserves the original model's structure while transferring, allowing it to be imported back as the exact same object in its native framework. Currently, PyMilo (v1.4) supports models built with scikit-learn. Support for PyTorch and TensorFlow is planned in upcoming releases.

Statement of Need

Modern machine learning development is largely centered around the Python ecosystem, which has become a dominant platform for building and training models due to its rich libraries and community support (Raschka et al., 2020). However, once a model is trained, sharing or deploying it securely and transparently remains a significant challenge (Davis et al., 2023; Parida et al., 2025). This issue is especially important in high-stakes domains such as healthcare, where ensuring model accountability and integrity is critical (Garbin & Marques, 2022). In such settings, any lack of clarity about a model's internal logic or origin can reduce trust in its predictions. Researchers have increasingly emphasized that greater transparency in AI systems is critical for maintaining user trust and protecting privacy in machine learning applications (Bodimani, 2024).

Despite ongoing concerns around transparency and safety, the dominant approach for exchanging pretrained models remains ad hoc binary serialization, most commonly through Python's `pickle` module or its variant `joblib`. These formats allow developers to store complex model objects with minimal effort, but they were never designed with security or human interpretability in mind (Parida et al., 2025). In fact, loading a pickle file may execute arbitrary code contained within it, a known vulnerability that can be exploited if the file is maliciously crafted (Brownlee, 2018; Python Software Foundation, 2024). While these methods preserve full model fidelity within the Python ecosystem, they pose serious security risks and lack transparency, as the serialized files are opaque binary blobs that cannot be inspected without loading. Furthermore, compatibility is fragile because pickled models often depend on specific library versions, which may hinder long-term reproducibility (Brownlee, 2018).

To improve portability across environments, several standardized model interchange formats have been developed alongside `pickle`. Most notably, Open Neural Network Exchange (ONNX) and Predictive Model Markup Language (PMML) convert trained models into framework-agnostic representations (Bai et al., 2025; Guazzelli et al., 2009), enabling deployment in diverse systems without relying on the original training code. ONNX uses a graph-based structure built from primitive operators (e.g., linear transforms, activations), while PMML provides an XML-based specification for traditional models like decision trees and regressions.

Although these formats enhance security by avoiding executable serialization, they introduce compatibility and fidelity challenges. Exporting complex pipelines to ONNX or PMML often leads to structural approximations, missing metadata, or unsupported components, especially for customized models (Guazzelli et al., 2009). As a result, the exported model may differ in behavior, resulting in performance degradation or loss of accuracy (Jajal et al., 2023). Jajal et al. (2023) found that models exported to ONNX can produce incorrect predictions despite successful conversion, indicating semantic inconsistencies between the original and exported versions. This reflects predictive performance degradation and highlights the risks of silent behavioral drift in deployed systems.

Beyond concerns about end-to-end model preservation, ONNX and PMML also present limitations in transparency, scope, and reversibility. ONNX uses a binary protocol buffer format that is not human-readable, which limits interpretability and makes auditing difficult. PMML, although XML-based and readable, is verbose and narrowly scoped, supporting only a limited subset of scikit-learn models. As noted by Cody et al. (2024), both ONNX and PMML focus on static model specification rather than operational testing or lifecycle validation workflows. Moreover, PMML does not provide a mechanism to restore exported models into Python, making it a one-way format that limits reproducibility across ML workflows.

Other tools have been developed to address specific use cases, though they remain limited in scope. For example, SKOPS improves the safety of scikit-learn model storage by enabling limited inspection of model internals without requiring code execution (skops-dev, 2024). However, it supports only scikit-learn models, lacks compatibility with other frameworks, and does not provide a fully transparent or human-readable structure. TensorFlow.js targets JavaScript environments by converting TensorFlow or Keras models into a JSON configuration file and binary weight files for execution in the browser or Node.js (Smilkov et al., 2019). However, this process has been shown to introduce compatibility issues, performance degradation, and inconsistencies in inference behavior due to backend limitations and environment-specific faults (Quan et al., 2022). Models from other frameworks, such as scikit-learn or PyTorch, must be re-implemented or retrained in TensorFlow to be exported. Additionally, running complex models in JavaScript runtimes introduces memory and performance limitations, often making the deployment of large neural networks prohibitively slow or even infeasible in browser environments (Nerd Corner, 2025).

In summary, current solutions force practitioners into trade-offs between security, transparency, end-to-end fidelity, and performance preservation. The machine learning community still lacks a safe and transparent end-to-end model serialization framework through which users can securely share models, inspect them easily, and accurately reconstruct them for use across diverse frameworks and environments.

PyMilo is proposed to address the above gaps. It is an open-source Python library that provides an end-to-end solution for exporting and importing machine learning models in a safe, non-executable, and human-readable format such as JSON. PyMilo serializes trained models into a transparent format and fully reconstructs them without structural changes, preserving their original functionality and behavior. This process does not affect inference time or performance and imports models on any target device without additional dependencies, enabling seamless execution in inference mode. While PyMilo may import functions from widely used scientific libraries during deserialization to restore model behavior (for example, NumPy or SciPy), the JSON representation itself never contains executable code; any remaining security risk is therefore inherited from these already-trusted dependencies rather than introduced by PyMilo's serialization mechanism. PyMilo benefits a wide range of stakeholders, including machine learning engineers, data scientists, and AI practitioners, by facilitating the development of more transparent and accountable AI systems. Furthermore, researchers working on transparent AI (Räuker et al., 2023), user privacy in ML (Bodimani, 2024), and safe AI (Macrae, 2019) can use PyMilo as a framework that provides transparency and safety in the machine learning environment.

References

Bai, J., Lu, F., Zhang, K., & others. (2025). *ONNX (Open Neural Network Exchange)* (Version 1.18.0). https://github.com/onnx/onnx

Bodimani, M. (2024). Assessing the impact of transparent AI systems in enhancing user trust and privacy. *Journal of Science & Technology*, *5*(1), 50–67. https://thesciencebrigade.com/jst/article/view/68

Brownlee, J. (2018). *Save and load machine learning models in Python with scikit-learn*. https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/.

Cody, T., Li, B., & Beling, P. (2024). On extending the Automatic Test Markup Language (ATML) for machine learning. *2024 IEEE International Systems Conference (SysCon)*, 1–8. https://doi.org/10.1109/SysCon61195.2024.10553464

Davis, J. C., Jajal, P., Jiang, W., Schorlemmer, T. R., Synovic, N., & Thiruvathukal, G. K. (2023). Reusing deep learning models: Challenges and directions in software engineering. *2023 IEEE John Vincent Atanasoff International Symposium on Modern Computing (JVA)*, 17–30. https://doi.org/10.1109/JVA60410.2023.00015

Garbin, C., & Marques, O. (2022). Assessing methods and tools to improve reporting, increase transparency, and reduce failures in machine learning applications in health care. *Radiology: Artificial Intelligence*, *4*(2), e210127. https://doi.org/10.1148/ryai.210127

The Journal of Open Source Software

Guazzelli, A., Zeller, M., Lin, W.-C., & Williams, G. (2009). *PMML: An open standard for sharing models*. https://doi.org/10.32614/RJ-2009-010

Hugging Face. (2025). *Safetensors - ML safer for all* (Version 0.6.2). https://github.com/huggingface/safetensors

Jajal, P., Jiang, W., Tewari, A., Kocinare, E., Woo, J., Sarraf, A., Lu, Y.-H., Thiruvathukal, G. K., & Davis, J. C. (2023). Analysis of failures and risks in deep learning model converters: A case study in the ONNX ecosystem. *arXiv Preprint arXiv:2303.17708*. https://doi.org/10.48550/arXiv.2303.17708

Macrae, C. (2019). Governing the safety of artificial intelligence in healthcare. *BMJ Quality & Safety*, *28*(6), 495–498. https://doi.org/10.1136/bmjqs-2019-009484

Nerd Corner. (2025). *TensorFlow.js vs TensorFlow (Python)*. https://nerd-corner.com/tensorflow-js-vs-tensorflow-python/.

Parida, S. K., Gerostathopoulos, I., & Bogner, J. (2025). How do model export formats impact the development of ML-enabled systems? A case study on model integration. *2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 48–59. https://doi.org/10.1109/CAIN66642.2025.00014

Python Software Foundation. (2024). *pickle — Python object serialization*. https://docs.python.org/3/library/pickle.html#security.

Quan, L., Guo, Q., Xie, X., Chen, S., Li, X., & Liu, Y. (2022). Towards understanding the faults of JavaScript-based deep learning systems. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 1–13. https://doi.org/10.1145/3551349.3560427

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, *11*(4), 193. https://doi.org/10.3390/info11040193

Räuker, T., Ho, A., Casper, S., & Hadfield-Menell, D. (2023). Toward transparent AI: A survey on interpreting the inner structures of deep neural networks. *2023 Ieee Conference on Secure and Trustworthy Machine Learning (Satml)*, 464–483. https://doi.org/10.1109/SaTML54575.2023.00039

skops-dev. (2024). *SKOPS* (Version 0.11.0). https://github.com/skops-dev/skops

Smilkov, D., Thorat, N., Assogba, Y., Nicholson, C., Kreeger, N., Yu, P., Cai, S., Nielsen, E., Soegel, D., Bileschi, S., & others. (2019). TensorFlow.js: Machine learning for the web and beyond. *Proceedings of Machine Learning and Systems*, *1*, 309–321. https://doi.org/10.48550/arXiv.1901.05350