Journal of Machine Learning Research 23 (2022) 1-8 Submitted 1/21; Revised 5/22; Published 9/22

PyPOTS: 一个用于部分观察时间序列机器学习的 Python 工具包

Wenjie Du WDU@TIME-SERIES.AI
PyPOTS Research

Yiyuan Yang YIYUAN.YANG@CS.OX.AC.UK
PyPOTS Research & University of Oxford

Linglong Qian LINGLONG.QIAN@KCL.AC.UK
PyPOTS Research & King' s College London

Jun Wang JWANGFX@CONNECT.UST.HK
PyPOTS Research & Hong Kong University of Science and Technology

Qingsong Wen QINGSONGEDU@GMAIL.COM
Squirrel Ai Learning

Editor: My editor

Abstract

PyPOTS 是一个开源的 Python 库，专门用于多变量部分观测时间序列中缺失值的数据挖掘和分析。特别地，它提供了对分类为五项任务的各种算法的轻松访问：插补、预测、异常检测、分类和聚类。包含的模型代表了一系列方法论范式，提供了一个统一且文档完善的接口，适用于学术研究和实际应用。其设计理念注重稳健性和可扩展性，在开发 PyPOTS 期间将软件构建的最佳实践作为原则进行实施，例如单元测试、持续集成和持续交付、代码覆盖率、可维护性评估、交互教程和平行化。该工具箱在 PyPI、Anaconda 和 Docker 上可用。PyPOTS 是开源的，并且公开发布在 GitHub https://github.com/WenjieDu/PyPOTS。

Keywords: 时间序列，数据科学，机器学习，神经网络，Python

1 介绍

缺失数据是现实世界时间序列中普遍存在的挑战，由于传感器错误、通信故障和其他意外故障。这一问题导致了部分观测到的时间序列（POTS），这可能会阻碍高级数据分析和建模 Wang et al. (2025) 。因此，在许多领域中有效的 POTS 算法非常有价值，包括健康监测 Silva et al. (2012) 、网络故障检测 Du et al. (2021) 、城市交通预测 Chen and Sun (2021) 以及基因表达分析 Bar-Joseph et al. (2003) 。

为了处理时间序列中的缺失数据，几个库提供了插补工具，如表 1 所示，Python 的填补基准 Khayati et al. (2020)，填充 Law (2021)，自动补全 Kearney (2022) 和填充缺口 Nater

表 1: 可用的部分观测时间序列（POTS）库的比较。✓ 表示已实现，(✓) 表示正在进行中，✗ 表示未实现。

库	建模任务					数量	数量	执行
	补全	预测	检测	分类	聚类	算法	数据集	环境
Mice	✓	✗	✗	✗	✗	1	20	R
ImputeTS	✓	✗	✗	✗	✗	10	6	R
Impyute	✓	✗	✗	✗	✗	10	1	Python
Autoimpute	✓	✗	✗	✗	✗	8	1	Python
ImputeBench	✓	✗	✗	✗	✗	20	10	Python
ImputeGAP	✓	(✓)	✗	✗	✗	34	17	Python
PyPOTS	✓	✓	✓	✓	✓	**52**	**172**	Python

et al. (2025)；以及 R 的老鼠 Van Buuren and Groothuis-Oudshoorn (2011) 和时间序列插补 Moritz and Bartz-Beielstein (2017)。虽然两步走的方法，即先填充缺失值然后进行下游建模，可能是有效的，但它可能在复杂任务中表现不佳。端到端方法通常直接操作 POTS 并能获得更好的结果。PyPOTS 通过支持模块化管道和集成模型 Du et al. (2024) 来容纳这两种策略。

尽管部分观测时间序列建模非常重要，但在 Python 这样庞大的社区中，仍没有专门的库来支持 POTS 的各种数据挖掘任务。PyPOTS 作为一个全面的 Python 工具箱被开发出来以填补这一空白。它为 POTS 上的各种任务提供了端到端的解决方案，超越了仅限插补的方法，使带有缺失数据的时间序列分析更加可靠。

PyPOTS 相比现有软件包具有以下明显优势：**1).** 它包含了 52 种算法和 172 个数据集，涵盖了部分观测时间序列的五个建模任务；**2).** 它提供了一个统一的接口、详细的文档以及所有算法的交互式教程，以方便开发和使用；**3).** 它利用了几项自动化服务来衡量、跟踪并确保库的质量，包括跨平台持续集成，覆盖所有算法的单元测试、代码覆盖率测量及可维护性评估；**4).** 尽可能地运用优化工具以提升库的扩展性。PyPOTS 能够在计算资源有限的情况下对大规模数据集进行模型训练，在多个 GPU 设备上并行运行一个模型，并使所有算法实现一次训练到处可运行。

2 项目集中度

稳健性保证。 GitHub 动作被用来自动进行单元测试，涉及多种版本的 Python 和不同的操作系统，包括 *Linux*（*Ubuntu* 发行版）、*macOS* 和窗口。存在 CI（持续集成）工作流程，

能够每天自动运行所有测试，并在提出拉取请求时确保代码库中的所有内容都是良好的。当 GitHub 上发布带有新功能的版本时，CD（持续交付）工作流程将自动运行构建管道，在 PyPI（Python 包索引）、Anaconda 和 Docker 上发布新版本，以便无缝地交付给用户。

代码质量保证。 PyPOTS 项目遵循 PEP 8 Python 代码样式指南。代码覆盖率发布在并由*覆盖 alls* 跟踪，一个基于网络的代码覆盖率服务。代码可维护性则通过 *SonarQube 云*进行评估，这是一个用于软件质量保证的自动化工具。PyPOTS 目前整体代码覆盖率为 85%，可维护性为 95%（被评为等级 **A**）。这些代码标准和测量是为了确保代码质量，并作为来自开源社区的所有拉取请求的安全保障。

文档和教程。 综合文档是使用斯芬克斯开发的，托管在阅读文档上，并可在 https://docs.pypots.com 获取。它保持与 NumPy Harris et al. (2020) 相同的文档字符串和渲染风格。文档包含详细的安装说明、快速入门示例、详细 API 参考以及常见问题列表。除了文档外，PyPOTS 的交互式教程以 Jupyter Notebook 的形式发布在 GitHub 仓库 https://github.com/WenjieDu/Brew-POTS 中，并提供了一个简化的教程谷歌 *Colab* 供用户即时运行和探索。

开源社区。 作为一个实用的机器学习库，PyPOTS 在科研界得到了广泛的应用，并被认为是 PyTorch 生态系统的一部分。PyPOTS 的代码托管在 GitHub 上以完全开源并鼓励社区合作。我们已经在即时消息平台（如 Slack 和微信）上建立了社群组，以便社群成员能够及时讨论和提供反馈。十几个贡献者正在帮助开发该框架，其他人则通过报告错误和请求功能来做出贡献。

3 设计与实现

基础框架。 PyPOTS 建立在像 NumPy Harris et al. (2020)，Scikit-learn Pedregosa et al. (2011)，SciPy Virtanen et al. (2020) 和 PyTorch Paszke et al. (2019) 这样的通用库之上，用于建模和计算，并使用熊猫以及 H5py Collette et al. (2017) 来处理数据。受 Scikit-learn Buitinck et al. (2013) 的 API 设计启发，它采用了一种统一且面向任务的接口用于插补、预测、异常检测、分类和聚类。拟合 () 处理训练过程并选择最佳模型检查点。填充 ()，预测 ()，检测 ()，分类 () 和聚类 ()，对应每个任务，运行推理并返回结果。与依赖通用预测 () 方法的传统库不同，PyPOTS 使用特定于任务的方法来明确表示模型的能力。其模块化设计利用继承和多态性，便于新模型的轻松集成。

可扩展性增强。 为了增强 PyPOTS 的可扩展性，设计并实现了三个关键特性：1). 数据懒加载：工业时间序列数据集通常庞大且内存密集。为了解决这个问题，PyPOTS 除了支持小型数据集的完全加载外，还提供了一种延迟加载策略。预处理的数据可以存储在 HDF5 文件中，并且只需根据需要将必要的数据批次读入内存，在训练和推理过程中显著减少了内存使用；2). 多设备并行加速：利用 PyTorch，PyPOTS 实现了无缝的 GPU 加速。对于大型数据集和

计算密集型模型，用户只需指定设备索引（例如，`["cuda:0", "cuda:1"]`），就可以轻松扩展到多个 GPU，使并行训练变得高效且易于访问；3). 统一模型序列化接口：PyPOTS 提供了一种一致的、与模型无关且与设备无关的接口来保存和加载模型。这便于将已训练好的模型在不同设备和环境中进行迁移，支持“一次训练，随处运行”的范式，并提高了部署灵活性。

超参数优化。 深度学习模型严重依赖超参数的选择，这显著影响性能 Feurer and Hutter (2019)。鉴于大多数 PyPOTS 算法是神经网络，超参数优化至关重要。这一需求因各领域时间序列数据集的多样性而加剧，在这种情况下，一个数据集的最佳设置可能在另一个数据集上表现不佳。为解决此问题，PyPOTS 集成了 MicrosoftNNI Microsoft (2021)，这是一个用于高效超参数调优的 AutoML 工具包。用户只需准备两个配置文件：一个定义超参数搜索空间，另一个定义非神经接口的调优设置。一旦启动，PyPOTS 将与 NNI 通信，并提供一个 Web 界面来监控调优过程并查看结果，这些结果可以根据性能指标进行排序以识别最佳配置。

一个展示。 代码片段展示了如何训练 BRITS 模型来分类 PhysioNet-2012 数据集 Goldberger et al. (2000)。有了准备好的数据集，用户只需要编写几行包含 PyPOTS 的代码即可训练模型并在新数据上生成结果。

```
# Our ecosystem kit will download and preprocess the dataset
from benchpots.datasets import preprocess_physionet2012
data = preprocess_physionet2012(subset='set-a',rate=0.1)
train_set = {"X": data["train_X"], "y": data["train_y"]}
val_set = {"X": data["val_X"], "y": data["val_y"]}
test_set, test_labels = {"X": data["test_X"]}, data["test_y"]

# Initialize and train the BRITS model
from pypots.classification import BRITS
from pypots.nn.functional import calc_binary_classification_metrics
model = BRITS(n_steps=data["n_steps"],
              n_features=data["n_features"],
              n_classes=data["n_classes"],
              rnn_hidden_size=256,
              epochs=5)
model.fit(train_set, val_set)

# Make predictions and evaluate
preds = model.classify(test_set)
metrics = calc_binary_classification_metrics(preds, test_labels)
```

4 结论

本文介绍了一个全面的 Python 工具包 PyPOTS，用于建模部分观测的时间序列。它包含 52 种算法并支持五项任务：插补、分类、聚类、异常检测和预测。致力于成为一个方便实用的库，PyPOTS 未来还有很长的路要走。在撰写本文时，PyPOTS 中大多数算法都是基于神经网络的最先进技术方法，可以取得出色的结果，但由于缺乏可解释性，它们并不适用于一些敏感领域，例如金融和营销。我们计划包含更多模型，特别是具有可解释性的模型，如概率方法和图模型。此外，我们将更加关注时空数据，这也是时间序列的一种常见类型，并实现与其兼容的模型。

参考文献

Ziv Bar-Joseph, Georg K Gerber, David K Gifford, Tommi S Jaakkola, and Itamar Simon. Continuous representations of time-series gene expression data. *Journal of Computational Biology*, 10(3-4):341–356, 2003.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

Xinyu Chen and Lijun Sun. Bayesian Temporal Factorization for Multidimensional Time Series Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2021.3066551. URL http://arxiv.org/abs/1910.06366.

Andrew Collette, James Tocknell, Thomas A Caswell, Darren Dale, Ulrik Kofoed Pedersen, Aleksandar Jelenak, Andrea Bedini, Martin Raspaud, Jialin Lei, Laurence Hole, Matthieu Brucher, Martin Teichmann, Ghislain Antony Vaillant, Jakirkham, Konrad Hinsen, Pierre De Buyl, Axel Huebl, Florian Rathgeber, Toon Verstraelen, Spaghetti Sort, Simon Gregor Ebner, Smutch, Matthew Zwier, Antony Lee, Matthew Brett, Joseph Kleinhenz, Jonah Bernhard, John Tyree, Antoine Pitrou, and Andy Salnikov. H5py/h5py: 2.7.0, March 2017. URL https://doi.org/10.5281/zenodo.594310.

Wenjie Du, David Cote, Chris Barber, and Yan Liu. Forecasting loss of signal in optical networks with machine learning. *J. Opt. Commun. Netw.*, 13(10):E109–E121, Oct 2021. doi: 10.1364/JOCN.423667. URL http://opg.optica.org/jocn/abstract.cfm?URI=jocn-13-10-E109.

Wenjie Du, Jun Wang, Linglong Qian, Yiyuan Yang, Zina Ibrahim, Fanxing Liu, Zepu Wang, Haoxin Liu, Zhiyuan Zhao, Yingjie Zhou, et al. Tsi-bench: Benchmarking time series imputation. *arXiv preprint arXiv:2406.12747*, 2024.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33, 2019.

A. Goldberger, L. Amaral, L. Glass, Jeffrey M. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, C. Peng, and H. Stanley. Physiobank, physiotoolkit, and physionet: components

of a new research resource for complex physiologic signals. *Circulation*, 101 23:E215–20, 2000.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Joe Kearney. Autoimpute. `https://github.com/kearnz/autoimpute`, 2022.

Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux. Mind the gap: an experimental evaluation of imputation of missing values techniques in time series. In *VLDB*, 2020.

Elton Law. Impyute. `https://github.com/eltonlaw/impyute`, 2021.

Microsoft. NNI: Neural Network Intelligence, 1 2021. URL `https://github.com/microsoft/nni`.

Steffen Moritz and Thomas Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *The R Journal*, 9(1):207–218, 2017. doi: 10.32614/RJ-2017-009.

Quentin Nater, Mourad Khayati, and Jacques Pasquier. Imputegap: A comprehensive library for time series imputation. 2025. URL `https://arxiv.org/abs/2503.15250`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. *Computing in cardiology*, 39:245, 2012.

Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Jun Wang, Wenjie Du, Yiyuan Yang, Linglong Qian, Wei Cao, Keli Zhang, Wenjia Wang, Yuxuan Liang, and Qingsong Wen. Deep learning for multivariate time series imputation: A survey. In *the 34th International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.