The Journal of Open Source Software

BayesFlow: Amortized Bayesian Workflows With Neural Networks

Stefan T. Radev[1*¶], **Marvin Schmitt**[2*], **Lukas Schumacher**[3], **Lasse Elsemüller**[3], **Valentin Pratz**[4], **Yannik Schälte**[5], **Ullrich Köthe**[4*], and **Paul-Christian Bürkner**[2,6*]

1 Cluster of Excellence STRUCTURES, Heidelberg University, Germany **2** Cluster of Excellence SimTech, University of Stuttgart, Germany **3** Institute for Psychology, Heidelberg University, Germany **4** Visual Learning Lab, Heidelberg University, Germany **5** Life and Medical Sciences Institute, University of Bonn, Germany **6** Department of Statistics, TU Dortmund University, Germany ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05702

Software

- Review
- Repository
- Archive

Editor: Øystein Sørensen

Reviewers:

- @sandeshkatakam
- @LoryPack

Submitted: 27 June 2023
Published: 22 September 2023

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

Summary

Modern Bayesian inference involves a mixture of computational techniques for estimating, validating, and drawing conclusions from probabilistic models as part of principled workflows for data analysis (Bürkner et al., 2022; Gelman et al., 2020; Schad et al., 2021). Typical problems in Bayesian workflows are the approximation of intractable posterior distributions for diverse model types and the comparison of competing models of the same process in terms of their complexity and predictive performance. However, despite their theoretical appeal and utility, the practical execution of Bayesian workflows is often limited by computational bottlenecks: Obtaining even a single posterior may already take a long time, such that repeated estimation for the purpose of model validation or calibration becomes completely infeasible.

`BayesFlow` provides a framework for *simulation-based* training of established neural network architectures, such as transformers (Vaswani et al., 2017) and normalizing flows (Papamakarios et al., 2021), for *amortized* data compression and inference. *Amortized Bayesian inference* (ABI), as implemented in `BayesFlow`, enables users to train custom neural networks on model simulations and re-use these networks for any subsequent application of the models. Since the trained networks can perform inference almost instantaneously (typically well below one second), the upfront neural network training is quickly amortized. For instance, amortized inference allows us to test a model's ability to recover its parameters (Schad et al., 2021) or assess its simulation-based calibration (Säilynoja et al., 2022; Talts et al., 2018) for different data set sizes in a matter of seconds, even though this may require the estimation of thousands of posterior distributions. `BayesFlow` offers a user-friendly API, which encapsulates the details of neural network architectures and training procedures that are less relevant for the practitioner and provides robust default implementations that work well across many applications. At the same time, `BayesFlow` implements a modular software architecture, allowing machine learning scientists to modify every component of the pipeline for custom applications as well as research at the frontier of Bayesian inference.

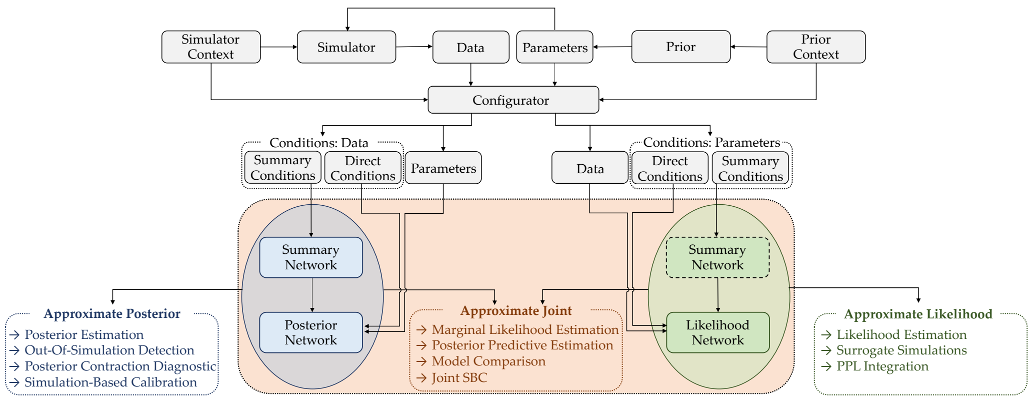

Figure 1: `BayesFlow` defines a formal workflow for data generation, neural approximation, and model criticism.

Statement of Need

`BayesFlow` embodies functionality that is specifically designed for building and validating amortized Bayesian workflows with the help of neural networks. Figure 1 outlines a typical workflow in the context of amortized posterior and likelihood estimation. A simulator coupled with a prior defines a generative Bayesian model. The generative model may depend on various (optional) context variates like varying numbers of observations, design matrices, or positional encodings. The generative scope of the model and the range of context variables determine the *scope of amortization*, that is, over which types of data the neural approximator can be applied without re-training. The neural approximators interact with model outputs (parameters, data) and context variates through a configurator. The configurator is responsible for carrying out transformations (e.g., input normalization, double-to-float conversion, etc.) that are not part of the model but may facilitate neural network training and convergence.

Figure 1 also illustrates an example configuration of four neural networks: 1) a summary network to compress simulation outcomes (individual data points, sets, or time series) into informative embeddings; 2) a posterior network to learn an amortized approximate posterior; and 3) another summary network to compress simulation inputs (parameters) into informative embeddings; and 4) a likelihood network to learn an amortized approximate likelihood. Figure 1 depicts the standalone and joint capabilities of the networks when applied in isolation or in tandem. The input conditions for the posterior and likelihood networks are partitioned by the configurator: Complex ("summary") conditions are processed by the respective summary network into embeddings, while very simple ("direct") conditions can bypass the summary network and flow straight into the neural approximator.

Currently, the software features four key capabilities for enhancing Bayesian workflows, which have been described in the referenced works:

1. **Amortized posterior estimation:** Train a generative network to efficiently infer full posteriors (i.e., solve the inverse problem) for all existing and future data compatible with a simulation model (Radev, Mertens, et al., 2020).
2. **Amortized likelihood estimation:** Train a generative network to efficiently emulate a simulation model (i.e., solve the forward problem) for all possible parameter configurations or interact with external probabilistic programs (Boelts et al., 2022; Radev et al., 2023).
3. **Amortized model comparison:** Train a neural classifier to recognize the "best" model in a set of competing candidates (Elsemüller et al., 2023; Radev, D'Alessandro, et al., 2020; Schmitt et al., 2022) or combine amortized posterior and likelihood estimation to compute Bayesian evidence and out-of-sample predictive performance (Radev et al., 2023).

4. **Model misspecification detection:** Ensure that the resulting posteriors are faithful approximations of the otherwise intractable target posterior, even when simulations do not perfectly represent reality (Radev et al., 2023; Schmitt et al., 2023).

`BayesFlow` has been used for amortized Bayesian inference in various areas of applied research, such as epidemiology (Radev et al., 2021), cognitive modeling (Krause et al., 2022; Schumacher et al., 2023; Sokratous et al., 2023; Wieschen et al., 2020), computational psychiatry (D'Alessandro et al., 2020), neuroscience (Ghaderi-Kangavari et al., 2022), particle physics (Bieringer et al., 2021), agent-based econometrics models (Shiono, 2021), seismic imaging (Siahkoohi et al., 2023), user behavior (Moon et al., 2023), structural health monitoring (Zeng et al., 2023), aerospace (Tsilifis et al., 2022) and wind turbine design (Noever-Castelos et al., 2022), micro-electro-mechanical systems testing (Heringhaus et al., 2022), and fractional Brownian motion (Verdier et al., 2022).

The software is built on top of `TensorFlow` (Abadi et al., 2016) and thereby enables off-the-shelf support for GPU and TPU acceleration. Furthermore, it can seamlessly interact with TensorFlow Probability (Dillon et al., 2017) for flexible latent distributions and a variety of joint priors.

Related Software

When a non-amortized inference procedure does not create a computational bottleneck, approximate Bayesian computation (ABC) might be an appropriate tool. This is the case if a single data set needs to be analyzed, if an infrastructure for parallel computing is readily available, or if repeated re-fits of a model (e.g., cross-validation) are not desired. A variety of mature Python packages for ABC exist, such as PyMC (Salvatier et al., 2016), pyABC (Schälte et al., 2022), ABCpy (Dutta et al., 2021), or ELFI (Lintusaari et al., 2018). In contrast to these packages, `BayesFlow` focuses on amortized inference, but can also interact with ABC samplers (e.g., use BayesFlow to learn informative summary statistics for an ABC analysis).

When it comes to simulation-based inference with neural networks, the `sbi` toolkit enables both likelihood and posterior estimation using different inference algorithms, such as Neural Posterior Estimation (Papamakarios et al., 2021), Sequential Neural Posterior Estimation (Greenberg et al., 2019) and Sequential Neural Likelihood Estimation (Papamakarios et al., 2019). `BayesFlow` and `sbi` can be viewed as complementary toolkits, where `sbi` implements a variety of different approximators for standard modeling scenarios, while `BayesFlow` focuses on amortized workflows with user-friendly default settings and optional customization. The `Swyft` library focuses on Bayesian parameter inference in physics and astronomy. `Swyft` uses a specific type of simulation-based neural inference technique, namely, Truncated Marginal Neural Ratio Estimation (Miller et al., 2021). This method improves on standard Markov chain Monte Carlo (MCMC) methods for ABC by learning the likelihood-to-evidence ratio with neural density estimators. Finally, the `Lampe` library provides implementations for a subset of the methods for posterior estimation in the `sbi` library, aiming to expose all components (e.g., network architectures, optimizers) in order to provide a customizable interface for creating neural approximators. All of these libraries are built on top of `PyTorch`.

Availability, Development, and Documentation

`BayesFlow` is available through PyPI via `pip install bayesflow`, the development version is available via GitHub. GitHub Actions manage continuous integration through automated code testing and documentation. The documentation is hosted at www.bayesflow.org. Currently, `BayesFlow` features seven tutorial notebooks. These notebooks showcase different aspects of the software, ranging from toy examples to applied modeling scenarios, and illustrating both posterior estimation and model comparison workflows.

Acknowledgments

We thank Ulf Mertens, Marco D'Alessandro, René Bucchia, The-Gia Leo Nguyen, Jonas Arruda, Lea Zimmermann, and Leonhard Volz for contributing to the GitHub repository. STR was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2181 - 390900948 (the Heidelberg Cluster of Excellence STRUCTURES), MS and PCB were supported by the Cyber Valley Research Fund (grant number: CyVy-RF-2021-16) and the DFG EXC-2075 - 390740016 (the Stuttgart Cluster of Excellence SimTech). LS and LE were supported by a grant from the DFG (GRK 2277) to the research training group Statistical Modeling in Psychology (SMiP). YS acknowledges support from the Joachim Herz Foundation. UK was supported by the Informatics for Life initiative funded by the Klaus Tschira Foundation. YS and UK were supported by the EMUNE project ("Invertierbare Neuronale Netze für ein verbessertes Verständnis von Infektionskrankheiten", BMBF, 031L0293A-D).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & others. (2016). TensorFlow: A system for large-scale machine learning. *Osdi*, *16*(2016), 265–283.

Bieringer, S., Butter, A., Heimel, T., Höche, S., Köthe, U., Plehn, T., & Radev, S. T. (2021). Measuring QCD splittings with invertible networks. *SciPost Physics*, *10*(6), 126. https://doi.org/10.21468/SciPostPhys.10.6.126

Boelts, J., Lueckmann, J.-M., Gao, R., & Macke, J. H. (2022). Flexible and efficient simulation-based inference for models of decision-making. *Elife*, *11*, e77220.

Bürkner, P.-C., Scholz, M., & Radev, S. T. (2022). Some models are useful, but how do we know which ones? Towards a unified Bayesian model taxonomy. *arXiv Preprint*.

D'Alessandro, M., Radev, S. T., Voss, A., & Lombardi, L. (2020). A Bayesian brain model of adaptive behavior: An application to the wisconsin card sorting task. *PeerJ*, *8*, e10316.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., & Saurous, R. A. (2017). *TensorFlow distributions*. https://arxiv.org/abs/1711.10604

Dutta, R., Schoengens, M., Pacchiardi, L., Ummadisingu, A., Widmer, N., Künzli, P., Onnela, J.-P., & Mira, A. (2021). ABCpy: A high-performance computing perspective to approximate Bayesian computation. *Journal of Statistical Software*, *100*(7), 1–38. https://doi.org/10.18637/jss.v100.i07

Elsemüller, L., Schnuerch, M., Bürkner, P.-C., & Radev, S. T. (2023). A deep learning method for comparing Bayesian hierarchical models. *arXiv Preprint arXiv:2301.11873*.

Gelman, A., Vehtari, A., Simpson, D., Margossian, C. C., Carpenter, B., Yao, Y., Kennedy, L., Gabry, J., Bürkner, P.-C., & Modrák, M. (2020). Bayesian workflow. *arXiv Preprint*.

Ghaderi-Kangavari, A., Rad, J. A., & Nunez, M. D. (2022). *A general integrative neurocognitive modeling framework to jointly describe EEG and decision-making on single trials*. https://doi.org/10.1007/s42113-023-00167-4

Greenberg, D., Nonnenmacher, M., & Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. *International Conference on Machine Learning*, *97*, 2404–2414.

Heringhaus, M. E., Zhang, Y., Zimmermann, A., & Mikelsons, L. (2022). Towards reliable parameter extraction in MEMS final module testing using Bayesian inference. *Sensors*, *22*(14), 5408. https://doi.org/10.3390/s22145408

Krause, M. von, Radev, S. T., & Voss, A. (2022). Mental speed is high until age 60 as revealed by analysis of over a million participants. *Nature Human Behaviour*, *6*(5), 700–708. https://doi.org/10.1038/s41562-021-01282-7

Lintusaari, J., Vuollekoski, H., Kangasrääsiö, A., Skytén, K., Järvenpää, M., Marttinen, P., Gutmann, M. U., Vehtari, A., Corander, J., & Kaski, S. (2018). ELFI: Engine for likelihood-free inference. *Journal of Machine Learning Research*, *19*(16), 1–7. http://jmlr.org/papers/v19/17-374.html

Miller, B. K., Cole, A., Forré, P., Louppe, G., & Weniger, C. (2021). Truncated marginal neural ratio estimation. *Advances in Neural Information Processing Systems*, *34*, 129–143.

Moon, H.-S., Oulasvirta, A., & Lee, B. (2023). Amortized inference with user simulations. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–20.

Noever-Castelos, P., Ardizzone, L., & Balzani, C. (2022). Model updating of wind turbine blade cross sections with invertible neural networks. *Wind Energy*, *25*(3), 573–599.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, *22*(1).

Papamakarios, G., Sterratt, D., & Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *The 22nd International Conference on Artificial Intelligence and Statistics*, 837–848.

Radev, S. T., D'Alessandro, M., Mertens, U. K., Voss, A., Köthe, U., & Bürkner, P.-C. (2020). Amortized Bayesian model comparison with evidential deep learning. *arXiv Preprint*. https://doi.org/10.1109/TNNLS.2021.3124052

Radev, S. T., Graw, F., Chen, S., Mutters, N. T., Eichel, V. M., Bärnighausen, T., & Köthe, U. (2021). OutbreakFlow: Model-based Bayesian inference of disease outbreak dynamics with invertible neural networks and its application to the COVID-19 pandemics in Germany. *PLoS Computational Biology*, *17*(10), e1009472. https://doi.org/10.1371/journal.pcbi.1009472

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., & Köthe, U. (2020). BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2020.3042395

Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., & Bürkner, P.-C. (2023). JANA: Jointly amortized neural approximation of complex Bayesian models. *arXiv Preprint arXiv:2302.09125*.

Säilynoja, T., Bürkner, P.-C., & Vehtari, A. (2022). Graphical test for discrete uniformity and its applications in goodness-of-fit evaluation and multiple sample comparison. *Statistics and Computing*, *32*(2), 32. https://doi.org/10.1007/s11222-022-10090-6

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, *2*, e55. https://doi.org/10.7717/peerj-cs.55

Schad, D. J., Betancourt, M., & Vasishth, S. (2021). Toward a principled Bayesian workflow in cognitive science. *Psychological Methods*, *26*(1), 103.

Schälte, Y., Klinger, E., Alamoudi, E., & Hasenauer, J. (2022). pyABC: Efficient and robust easy-to-use approximate Bayesian computation. *Journal of Open Source Software*, *7*(74), 4304. https://doi.org/10.21105/joss.04304

Schmitt, M., Bürkner, P.-C., Köthe, U., & Radev, S. T. (2023). Detecting model misspecification in amortized Bayesian inference with neural networks. *45th German Conference on Pattern Recognition (GCPR)*.

Schmitt, M., Radev, S. T., & Bürkner, P.-C. (2022). Meta-uncertainty in Bayesian model comparison. *arXiv Preprint arXiv:2210.07278*.

Schumacher, L., Bürkner, P.-C., Voss, A., Köthe, U., & Radev, S. T. (2023). Neural superstatistics for Bayesian estimation of dynamic cognitive models. *Scientific Reports*, *13*(1), 13778. https://doi.org/10.1038/s41598-023-40278-3

Shiono, T. (2021). Estimation of agent-based models using Bayesian deep learning approach of BayesFlow. *Journal of Economic Dynamics and Control*, *125*, 104082. https://doi.org/10.1016/j.jedc.2021.104082

Siahkoohi, A., Rizzuti, G., Orozco, R., & Herrmann, F. J. (2023). Reliable amortized variational inference with physics-based latent distribution correction. *Geophysics*, *88*(3), R297–R322. https://doi.org/10.1190/geo2022-0472.1

Sokratous, K., Fitch, A. K., & Kvam, P. D. (2023). How to ask twenty questions and win: Machine learning tools for assessing preferences from small samples of willingness-to-pay prices. *Journal of Choice Modelling*, *48*, 100418. https://doi.org/10.1016/j.jocm.2023.100418

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. (2018). Validating Bayesian inference algorithms with simulation-based calibration. *arXiv Preprint*.

Tsilifis, P., Ghosh, S., & Andreoli, V. (2022). Inverse design under uncertainty using conditional normalizing flows. *AIAA Scitech 2022 Forum*, 0631. https://doi.org/10.2514/6.2022-0631

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Verdier, H., Laurent, F., Cassé, A., Vestergaard, C. L., & Masson, J.-B. (2022). Variational inference of fractional Brownian motion with linear computational complexity. *Physical Review E*, *106*(5), 055311. https://doi.org/10.1103/PhysRevE.106.055311

Wieschen, E. M., Voss, A., & Radev, S. (2020). Jumping to conclusion? A Lévy flight model of decision making. *The Quantitative Methods for Psychology*, *16*(2), 120–132. https://doi.org/10.20982/tqmp.16.2.p120

Zeng, J., Todd, M. D., & Hu, Z. (2023). Probabilistic damage detection using a new likelihood-free Bayesian inference method. *Journal of Civil Structural Health Monitoring*, *13*(2-3), 319–341. https://doi.org/10.1007/s13349-022-00638-5