Trieste: Efficiently Exploring The Depths of Black-box Functions with TensorFlow

**Victor Picheny, Joel Berkeley, Hrvoje Stojic, Uri Granta,
Sebastian W. Ober, Artem Artemev, Khurram Ghani, Alexander Goodall,
Andrei Paleyes, Sattar Vakili, Sergio Pascual-Diaz, Stratis Markou**
Secondmind, Cambridge, UK
`victor@secondmind.ai`

Henry B. Moss
University of Cambridge & Lancaster University, UK

Jixiang Qing, Nasrulloh R.B.S Loka, Ivo Couckuyt
Ghent University - imec, Ghent, Belgium

Abstract

We present TRIESTE, an open-source Python package for Bayesian optimization and active learning benefiting from the scalability and efficiency of TENSORFLOW. Our library enables the plug-and-play of popular TENSORFLOW-based models within sequential decision-making loops, e.g. Gaussian processes from GPFLOW or GPFLUX, or neural networks from KERAS. This modular mindset is central to the package and extends to our acquisition functions and the internal dynamics of the decision-making loop, both of which can be tailored and extended by researchers or engineers when tackling custom use cases. TRIESTE is a research-friendly and production-ready toolkit backed by a comprehensive test suite, extensive documentation, and available at `https://github.com/secondmind-labs/trieste`.

1 Introduction

TENSORFLOW is one of Python's primary machine learning frameworks, offering both flexibility and scalability through its support for auto-differentiation and GPU-based computation. Yet, TENSORFLOW 2 does not have a library for Bayesian Optimization (BO) — an increasingly popular method for black-box optimization under heavily constrained optimization budgets [see 45, for an introduction]. This lack of support is likely due to the inherently sequential and evolving nature of active learning loops, which makes BO implementations prone to trigger expensive retracing of the computational graphs, as used by TENSORFLOW to accelerate numerical calculations. However, once special care is taken to avoid unnecessary retracing, a TENSORFLOW-based BO library would allow users not only to leverage versatile and powerful (probabilistic) TENSORFLOW modeling libraries (e.g. KERAS, GPFLOW, GPFLUX), but also to benefit from BO-specific perks like the freedom to define acquisition functions without specifying their gradients, easily parallelized optimization of acquisition functions, and in-the-loop monitoring of models and convergence statistics (e.g. TENSORBOARD).

In this paper, we present TRIESTE[1], a highly modular, flexible and general-purpose BO library designed to enable users working within TENSORFLOW ecosystems to:

[1]Trieste was the first crewed vessel to reach the bottom of the Mariana trench — the literal global minimum.

Workshop on Bayesian Decision-making and Uncertainty, 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1. deploy their own existing models to drive BO loops, and
2. build BO pipelines that harness the ease and computational efficiency provided by TENSORFLOW's automatic differentiation and support for modern compute resources like GPUs.

Our library is oriented towards real-world use and contains a wide range of advanced BO functionalities, with a focus on modularity to allow ease of extension with custom models and acquisition functions. Trieste's funcionality is matched only by TORCH-based BOTORCH of [2] (see Section 4) which, although widely regarded as the *state-of-the-art* BO Python library, does not support the large number researchers and engineers with TENSORFLOW models and pipelines.

2 Related Work

Many open-source libraries have been built to support the recent increase in the use and development of BO methodology. For example, Python users with models written in TORCH or NUMPY can easily find compatible libraries such as BOTORCH [2], GPYOPT [1], ROBO [22], EMUKIT [33] or DRAGONFLY [21]. Similarly, those running R, C++ or JAVA can use DICEOPTIM [41], BAYESOPT [26] or SMAC [17]. The library GPFLOWOPT [24] was built on TENSORFLOW 1 with an intent similar to TRIESTE, but is not actively maintained anymore and does not support the fundamentally different TENSORFLOW 2.

3 Key features and Design

We now present the modular structure of TRIESTE which contains four key building blocks, a choice of high-level interface (either `AskTellOptimizer` or `BayesianOptimizer`), a choice of `ProbabilisticModel`, and a pairing of `AcquisitionRule` and `AcquisitionFunction`. Although TRIESTE's structure allows a high level of customization, sensible defaults are provided throughout the library in order to give new users good starting points.

3.1 Interfaces for different levels of control over function evaluation

A key design choice of TRIESTE is its `AskTellOptimizer` interface, which need not have direct access to the objective function. In many libraries, the objective must be a query-able function to be passed into the loop and called for each BO step, an assumption that is rarely suitable when performing BO in the real world. In contrast, an `AskTellOptimizer` outputs recommended query points (the *ask*) and then waits for the user to return new evaluations (the *tell*) (see Figure 3). This interface allows TRIESTE users to apply BO across a range of non-standard real-world settings, e.g., when evaluating the objective function requires laboratory [19] or distributed computing resources, such that users have only partial control over the environment [37] or batches of evaluations arrive asynchronously [20]. For settings where it is appropriate for the objective function to be passed into the BO, e.g., for experiments with synthetic problems, we also provide a more standard `BayesianOptimizer` interface that will run multiple BO steps.

3.2 Versatile Model Support

The models in the BO loop can be any model written in TENSORFLOW, and TRIESTE is designed to make it easy to add a new model, through a set of general model interfaces. We provide direct interfaces to import models from well-established TENSORFLOW modeling libraries, e.g., Gaussian processes from GPFLOW [27] and GPFLUX [10], as well as neural networks from KERAS. TRIESTE users have a range of popular probabilistic models from these libraries available out of the box. They cover both regression and classification tasks and range from standard GPs [40] to alternatives like sparse variational GPs [14], Deep GPs [43] or Deep Ensembles [25], that scale much better with the number of function evaluations. Finally, we provide user-friendly model builders with sensible default setups, allowing users to get more quickly to good results and facilitate usage for those with less experience with probabilistic models. Hence, TRIESTE users can benefit from a large choice of models with a wide range of complexity, unlocking novel applications for BO.

Importantly, TRIESTE's BO loops allow the modeling of multiple quantities, using either multiple separate models or a single multi-output model. This framework naturally supports common BO

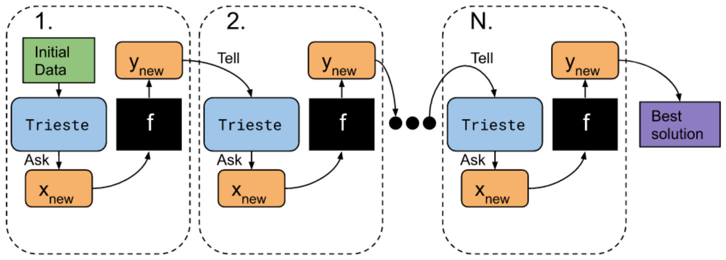

Figure 1: `AskTellOptimizer`

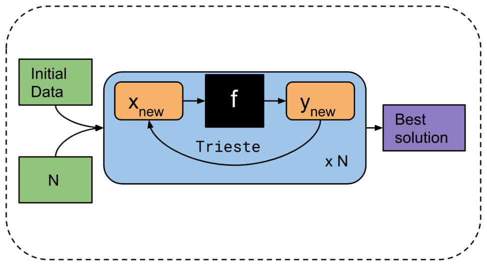

Figure 2: `BayesianOptimizer`

Figure 3: TRIESTE's interfaces over N optimization steps. The `AskTellOptimizer` requires users to make manual evaluations, which is useful for real-world settings, e.g. evaluating the objective function requires laboratory or distributed computing resources. In contrast, the `BayesianOptimizer` queries the black box directly, performing all N BO iterations without user interaction.

extensions like multi-objective optimization [23], multi-fidelity optimization [46], optimization with constraints [44], and combinations thereof.

3.3 Acquisition Rules and Functions

Regardless of the interface and model choice, the recommendation of query points is controlled by an `AcquisitionRule`. While the vanilla BO rule is to query the point that optimizes a particular `AcquisitionFunction`, the `AcquisitionRule` is a useful abstraction to handle complex cases for which a high level of flexibility is needed. For example, Trieste includes variable optimization spaces for `AcquisitionFunction` [e.g., using trust regions, 9], a multi-step procedure for selecting query points [4], and a greedy approach to build batches of query points [29].

A wide range of acquisition functions are already provided in TRIESTE to tackle most of the usual BO cases, with many based on the gold-standard Expected Improvement [EI; 18], including variants for batch [6, 12, 2], noisy [16], multi-objective [7] and constrained [11] optimization. We also include implementations of recent information-theoretic approaches for multi-fidelity optimization [30], a scalable extension of batched Thompson sampling [47], as well as popular active learning methods for improved classification [15] or contour line estimation [35].

TRIESTE is designed to make it straightforward for a user to specify a new acquisition function or rule. Automatic differentiation directly provides the function gradients, which are leveraged by TRIESTE's supported acquisition function optimizers, including an effective parallelized multi-start L-BFGS-B optimizer. Special care is taken to allow `AcquisitionFunctions` to be updated without expensive retracing of the computational graphs for each BO step.

4 Feature Benchmark

Table 1 compares TRIESTE with two other popular BO frameworks, demonstrating that it is a general-purpose BO toolbox. TRIESTE has comparable features to the Torch-based BOTORCH and provides substantially more functionality than the NUMPY-based EMUKIT. The other Python BO libraries introduced in Section 2 are not included in this comparison as they either have even less functionality

	Auto-diff	GPU	Ask-tell	Batch	Multi-fidelity	Multi-objective	Const-raints	Back-end
TRIESTE	✓	✓	✓	✓	✓	✓	✓	TENSORFLOW
BOTORCH	✓	✓	✓[2]	✓	✓	✓	✓	TORCH
EMUKIT	×	×	×	✓	✓	×	✓	NUMPY

Table 1: A comparison of feature support (according to their documentation) of relevant Python BO libraries still under active maintenance.

and are no longer maintained (GPYOPT, ROBO and GPFLOWOPT), or target only hyper-parameter optimization (DRAGONFLY and SMAC).

5 Conclusions and Future Plans

TRIESTE is an open-source project that allows TENSORFLOW practitioners to easily use BO in their systems. It is a highly flexible library designed with modularity in mind, easy to extend with custom models and acquisition functions. Backed by continuous integration and comprehensive unit tests (97% coverage), TRIESTE is a reliable and robust framework used for both real-world deployment and research has recently been taken up by researchers to develop new BO methodology [47, 36, 5, 13, 29, 31, 32, 37, 38, 39] whilst also being used across a range of applications including designing heat exchangers [34] and improving adhesive bonding [19].

TRIESTE relies on features added and improved by the community and so we gladly welcome feature requests and code contributions. We plan to continue to add new functionality orientated to supporting the application of BO in the real world. In the near term, this will include high-dimensional objective functions [3] and non-Euclidean search spaces [28, 42, 8].

6 Acknowledgements

This research receives funding from the Flemish Government under the "Onderzoeksprogramma Artifciele Intelligentie (AI) Vlaanderen" programme.

References

[1] The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[2] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: a framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 2020.

[3] Mickael Binois and Nathan Wycoff. A survey on high-dimensional gaussian process modeling with application to bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2022.

[4] Mickael Binois, Nicholson Collier, and Jonathan Ozik. A portfolio approach to massively parallel bayesian optimization. *arXiv preprint arXiv:2110.09334*, 2021.

[5] Paul E Chang, Prakhar Verma, ST John, Victor Picheny, Henry Moss, and Arno Solin. Fantasizing with dual gps in bayesian optimization and active learning. *arXiv preprint arXiv:2211.01053*, 2022.

[6] Clément Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, 2013.

[7] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 2020.

[2]The AX library (`https://ax.dev/`) is a high-level interface for BOTORCH somewhat similar to an ask-tell.

[8] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Mercer features for efficient combinatorial bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[9] Youssef Diouane, Victor Picheny, Rodolophe Le Riche, and Alexandre Scotto Di Perrotolo. Trego: a trust-region framework for efficient global optimization. *Journal of Global Optimization*, 2022.

[10] Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P Deisenroth, and ST John. Gpflux: A library for deep gaussian processes. *arXiv preprint arXiv:2104.05674*, 2021.

[11] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, 2014.

[12] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, 2016.

[13] Arash Heidari, Jixiang Qing, Sebastian Rojas Gonzalez, Jürgen Branke, Tom Dhaene, and Ivo Couckuyt. Finding knees in bayesian multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, 2022.

[14] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Proceedings of AISTATS*, 2015.

[15] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[16] Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 2006.

[17] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, 2011.

[18] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 1998.

[19] Jeroen Jordens, Bart Van Doninck, Nasrulloh RB Satrio, Alejandro Morales Hernández, Ivo Couckuyt, Inneke Van Nieuwenhuyse, and Maarten Witters. Optimization of plasma-assisted surface treatment for adhesive bonding via artificial intelligence. In *2nd International Conference on Industrial Applications of Adhesives 2022*, 2023.

[20] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[21] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *The Journal of Machine Learning Research*, 2020.

[22] Aaron Klein, Stefan Falkner, Numair Mansur, and Frank Hutter. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian optimization workshop*, 2017.

[23] Joshua Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 2006.

[24] Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. Gpflowopt: A bayesian optimization library using tensorflow. *arXiv preprint arXiv:1711.03845*, 2017.

[25] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[26] Ruben Martinez-Cantin. Bayesopt: a bayesian optimization library for nonlinear optimization, experimental design and bandits. *J. Mach. Learn. Res.*, 2014.

[27] Alexander G de G Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *J. Mach. Learn. Res.*, 2017.

[28] Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. Boss: Bayesian optimization over string spaces. *Advances in neural information processing systems*, 2020.

[29] Henry B Moss, David S Leslie, Javier Gonzalez, and Paul Rayson. Gibbon: General-purpose information-based bayesian optimisation. *Journal of Machine Learning Research*, 2021.

[30] Henry B Moss, David S Leslie, and Paul Rayson. Mumbo: Multi-task max-value bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases*, 2021.

[31] Henry B Moss, Sebastian W Ober, and Victor Picheny. Information-theoretic inducing point placement for high-throughput bayesian optimisation. *arXiv preprint arXiv:2206.02437*, 2022.

[32] Henry B Moss, Sebastian W Ober, and Victor Picheny. Inducing point allocation for sparse gaussian processes in high-throughput bayesian optimisation. *Artificial intelligence and statistics*, 2023.

[33] Andrei Paleyes, Mark Pullin, Maren Mahsereci, Cliff McCollum, Neil D Lawrence, and Javier Gonzalez. Emulation of physical processes with emukit. *arXiv preprint arXiv:2110.13293*, 2021.

[34] Andrei Paleyes, Henry B Moss, Victor Picheny, Piotr Zulawski, and Felix Newman. A penalisation method for batch multi-objective bayesian optimisation with application in heat exchanger design. *arXiv preprint arXiv:2206.13326*, 2022.

[35] Victor Picheny, David Ginsbourger, Olivier Roustant, Raphael T Haftka, and Nam-Ho Kim. Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design*, 2010.

[36] Victor Picheny, Henry Moss, Leonard Torossian, and Nicolas Durrande. Bayesian quantile and expectile optimisation. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[37] Jixiang Qing, Ivo Couckuyt, and Tom Dhaene. A robust multi-objective bayesian optimization framework considering input uncertainty. *Journal of Global Optimization*, 2022.

[38] Jixiang Qing, Tom Dhaene, and Ivo Couckuyt. Spectral representation of robustness measures for optimization under input uncertainty. In *ICML2022, the 39th International Conference on Machine Learning*, 2022.

[39] Jixiang Qing, Henry B Moss, Tom Dhaene, and Ivo Couckuyt. $\{\mathrm{PF}\}^2$es: Parallel feasible pareto frontier entropy search for multi-objective bayesian optimization. *Artificial intelligence and statistics*, 2023.

[40] CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. 2006.

[41] Olivier Roustant, David Ginsbourger, and Yves Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of statistical software*, 2012.

[42] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, 2020.

[43] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

[44] Matthias Schonlau, William J Welch, and Donald R Jones. Global versus local search in constrained optimization of computer models. *Lecture notes-monograph series*, 1998.

[45] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 2016.

[46] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 2013.

[47] Sattar Vakili, Henry Moss, Artem Artemev, Vincent Dutordoir, and Victor Picheny. Scalable thompson sampling using sparse gaussian process models. *Advances in Neural Information Processing Systems*, 2021.