Hyperparameter Optimization as a Service on INFN Cloud

Matteo Barbetti[1,2] **and Lucio Anderlini**[2]

[1] Department of Information Engineering, University of Firenze,
via Santa Marta 3, Firenze (FI), Italy

[2] Istituto Nazionale di Fisica Nucleare, Sezione di Firenze,
via G. Sansonse 1, Sesto Fiorentino (FI), Italy

E-mail: Matteo.Barbetti@fi.infn.it

Abstract. The simplest and often most effective way of parallelizing the training of complex machine learning models is to execute several training instances on multiple machines, scanning the hyperparameter space to optimize the underlying statistical model and the learning procedure. Often, such a meta learning procedure is limited by the ability of accessing securely a common database organizing the knowledge of the previous and ongoing trials. Exploiting opportunistic GPUs provided in different environments represents a further challenge when designing such optimization campaigns. In this contribution we discuss how a set of REST APIs can be used to access a dedicated service based on INFN Cloud to monitor and coordinate multiple training instances, with gradient-less optimization techniques, via simple HTTP requests. The service, called Hopaas (*Hyperparameter OPtimization As A Service*), is made of web interface and sets of APIs implemented with a FastAPI backend running through Uvicorn and NGINX in a virtual instance of INFN Cloud. The optimization algorithms are currently based on Bayesian techniques as provided by Optuna. A Python frontend is also made available for quick prototyping. We present applications to hyperparameter optimization campaigns performed by combining private, INFN Cloud and CINECA resources. Such multi-node multi-site optimization studies have given a significant boost to the develop of a set of parameterization for the ultra-fast simulation of the LHCb experiment.

1. Introduction

In the last decade, machine learning (ML) has become an incredibly valuable tool in practically every field of application, from scientific research to industry. Increasingly complex models exhibit performance unthinkable until few years ago in a wide range of applications, such as image generation [1], language modelling [2] or medical diagnosis [3]. Most of the ML techniques rely on the optimization of an *objective function* with respect to some internal parameters, describing the performance of the algorithm. Usually, when the optimum of the objective function is a minimum, the names *cost* or *loss function* are adopted. The fastest iterative optimization techniques rely on the (Stochastic) Gradient Descent technique [4]. Unfortunately, for a wide

class of optimization problems the gradient of the loss function with respect to the model parameter is extremely expensive to compute or cannot be defined at all. For example, optimization problems involving noisy loss functions in contexts where analytical derivatives cannot be computed cannot rely on gradient-descent techniques, requiring the adoption of slower, often heuristic, methods. A widely adopted option is to define a *surrogate model* describing the variations of the loss function across the parameter space together with its uncertainty. Such model is then employed to drive the optimization algorithm to explore those regions where improvements were not statistically excluded from previous evaluations. Techniques adopting this approach are referred to as Bayesian optimization (BO) methods and have been an active area of research in ML for the last decade [5, 6, 7, 8, 9, 10, 11].

Tuning the performance of ML models may benefit from *hyperparameter* optimization (HPO). Hyperparameters are defined as all those parameters that are not learned during the model training procedure, but rather encode some arbitrariness in the architecture of the model itself or in the procedure to train it [5]. In practice, HPO studies require training the model multiple times to explore the hyperparameter space. Since training ML models is computationally expensive, HPO campaigns should focus as much as possible on those regions of the hyperparameter space where the model performs better. This allows to reduce the time needed for finding the best configuration of the model under investigation. Typically, multiple training procedures may result in different model performance because of the intrinsic randomness of the stochastic gradient-descent techniques. Namely, the loss is often a noisy function of the hyperparameters.

Exploring the hyperparameter space requires many independent trainings, or *trials*, that can run in parallel on different computing resources. In general, the greater the number of resources tackling trainings, the larger the hyperparameter space is explored. This makes it possible to obtain better models. Opportunistic access to computing resources may provide valuable contribution to HPO campaigns. Unfortunately, coordinating studies on resources from different providers, restrictions, and regulations challenges the adoption of existing HPO services.

In this document, we propose Hopaas (*Hyperparameter OPtimization As A Service*). Hopaas allows to orchestrate HPO studies across multiple computing instances by using a minimal set of REST APIs. Computing nodes from multiple HPC centers can concur *dynamically* to the same optimization study, requesting to the Hopaas server a set of hyperparameters to test and then sending back the outcome of the training procedure. Several trials of one or more studies can be tracked and monitored through the web interface provided by the Hopaas service. A reference implementation, with a server instance‡ deployed on INFN Cloud resources and a simple client package [12] wrapping the REST APIs to Python functions, is also discussed.

‡ Visit `https://hopaas.cloud.infn.it` for additional details.

API	Description	HTTP method	Request path
`version`	Provides the version of the HOPAAS backend.	GET	`/api/version`
`ask`	Creates a new trial, contributing to a new/existing study. The POST body request should include the set of settings to refer unambiguously to a study. The API response contains the hyperparameters to test.	POST	`/api/ask/token`
`tell`	Provides the final score of a trial to the backend optimizer chosen for the study.	POST	`/api/tell/token`
`should_prune`	Provides an intermediate score to the backend optimizer. If the study includes a *pruner* strategy, the API response is a boolean value saying whether or not to continue the current trial.	POST	`/api/should_prune/token`

Table 1. Minimal description of the REST APIs provided by the HOPAAS service.

2. HOPAAS API specification

We refer to a *trial* as a single training attempt with a specific set of hyperparameters to test. A *study* represents an optimization session and includes a collection of trials. In practice, a study is unambiguously defined by the set of hyperparameters to optimize, the range of values where searching the optimum, and the modality in which this search is carried out (e.g., grid search, Bayesian methods [5], or evolutionary algorithms [13]).

The core activity of the HOPAAS service is to manage distributed optimization studies by providing sets of hyperparameters to requesting computing nodes, the so-called HOPAAS clients. The creation, intermediate updates, and finalization of a trial is controlled from the client-side by using a set of REST APIs. Such APIs, named `ask`, `tell`, and `should_prune`, implement these actions upon POST HTTP requests with user authentication based on an *API token* in the request path. A minimal description of the HOPAAS REST APIs is depicted in Table 1 and further detailed in the rest of this section.

A computing node ready to test a set of hyperparameters, whether it comes from on-premises, Cloud, or HPC resources, will simply need a network connection with the HOPAAS server to take part to an optimization campaign. In particular, it will query the HOPAAS server via the `ask` API, including in the request body all the information needed to define a study unambiguously. The HOPAAS server will define a new trial,

possibly assigning it to an existing study, or creating a new one. Once created the trial, the HOPAAS server provides it with a unique identifier that is included in the HTTP response together with the set of hyperparameters to be evaluated for the study.

Usually, the evaluation of a set of hyperparameters consists of training a model defined by those hyperparameters aiming at the resulting value of the objective function. The evaluated performance metric may correspond to the loss function computed during the training procedure but, in general, it can be any numerical score obtained processing a given set of hyperparameters. Once the evaluation is completed, the computing node will finalize the trial using the `tell` API, whose body will include the unique identifier of the trial and the final evaluation of the objective function.

The HOPAAS server may serve multiple `ask` requests from different sources, assigning them to one or different studies, while updating the surrogate model each time a new evaluation is made available by querying the `tell` API.

Depending on the specific ML algorithm, intermediate evaluations of the objective function can be accessed during the training procedure and used to abort non-promising trials (*pruning*) without wasting computing power to take the training procedure to an end. Optionally, the computing node may update the HOPAAS server with intermediate evaluations of the objective function by querying the `should_prune` API for monitoring and pruning purposes. The body of a `should_prune` request will contain the unique identifier of the trial, the intermediate value of the loss function, and an integer number encoding the progress of the training procedure, the so-called *step*. The HTTP response will indicate whether the study should be early terminated, or it is sufficiently likely to result in an improvement over the previous tests.

A reference Python frontend was developed aiming at a facilitated access to the HOPAAS service from Python applications [12]. While Python is a primary choice for many scientific applications, it should be noticed that the client simply wraps the REST APIs into classes and functions, as the HOPAAS protocol is designed to be language-agnostic, relying on widely adopted web communication standards. In addition, the HOPAAS client is also framework-agnostic since the evaluation of the objective function for a given set of hyperparameters can be implemented with any framework and environment.

3. Implementation

The reference implementation for the HOPAAS service running on INFN Cloud relies on containerized applications orchestrated with `docker-compose` [14]. The web server implementing the REST APIs is a scalable set of Uvicorn instances [15] running an application based on the FastAPI framework [16]. The BO algorithms are provided by integrating the backend with OPTUNA [17], while future extensions to additional frameworks are planned. The access to the Uvicorn instances from the Internet is mediated by an NGINX reverse proxy [18] accessed via the encrypted HTTPS protocol. A PostgreSQL instance [19] is part of the `docker-compose` configuration to provide

shared persistency to the multiple instances of the web application backend. The workflow of the interaction between the HOPAAS server and computing nodes is depicted in Figure 1.

The same HOPAAS server is designed to serve web-based user access. A web application, developed in HTML, CSS and JavaScript, is shipped to the client browser as defined by a set of web-specific APIs in Uvicorn. The web pages of the frontend provide dynamic visualizations by fetching data from specialized APIs at regular intervals. Plots showing the evolution of the loss reported by different studies and trials are obtained with the CHARTIST library [20].

The user authentication and authorization procedure of the web application is managed relying on *access tokens* as defined by the OAuth2 standard, using the INFN GitLab instance as identity provider. Support for INDIGO IAM is also planned for the future [21]. Once authenticated, users can generate multiple API tokens through the web application. Each API token has a validity period defined at generation and can be revoked at any time. Tokens with shorter validity are more appropriate for usage in public or untrusted contexts.

4. Tuning the LHCb ultra-fast simulation models with HOPAAS and MARCONI 100

Machine Learning is an important research area in High Energy Physics (HEP), with first applications dating back to the 1990s. Recent years have witnessed an explosion

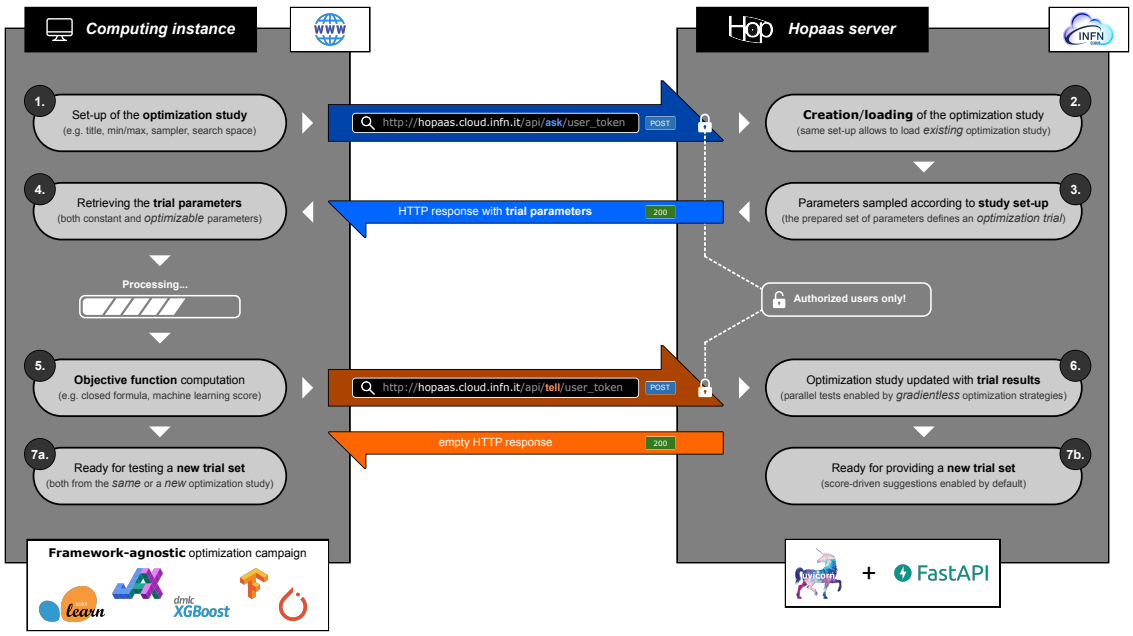

Figure 1. Workflow of an optimization study with a client-server approach based on REST APIs.

in the use of ML techniques in HEP to face the computational challenges raised by the upcoming and future runs of the Large Hadron Collider (LHC) [22]. With an increasing number, complexity and range of applications of the ML models, HPO is becoming popular in HEP [23], and specialized frameworks targeting distributed computing are being developed [8].

The reference implementation of the Hopaas service presented here has been successfully used for HEP applications and, in particular, to optimize the parameterizations for Lamarr [24, 25], a novel LHCb *ultra-fast simulation* framework. Most of the parameterizations of Lamarr rely on Generative Adversarial Networks (GANs) [26], advanced algorithms taken from Computer Vision that were demonstrated to be able to well reproduce the distributions obtained from standard simulation techniques [27, 28]. Adversarial models are particularly sensitive to the choice of the hyperparameter configuration and require intensive optimization campaigns to model accurately the target distributions.

Several optimization studies have been orchestrated by the Hopaas service using *diverse* computing instances, from scientific providers (like INFN, CERN and CINECA) and from commercial cloud providers (like GCP or AWS). Most of the resources have been provided by the CINECA supercomputer Marconi 100, with a custom network configuration to enable the communication with the Hopaas server [29]. Hopaas was able to coordinate dozens of optimization studies with hundreds of trials on each study from more than twenty concurrent and diverse computing nodes. This complex setup has allowed to outperform the previous results and obtain a set of GAN models that succeeds in parameterizing the high-level response of the LHCb experiment [24, 25].

5. Conclusion and future work

Hyperparameter tuning and Bayesian methods for gradient-less optimization provide an effective and simple mean of exploiting opportunistic compute resources to improve ML models. Unfortunately, environment variability and constraints set by different resource providers make the application of existing HPO services challenging. With Hopaas, we propose a solution designed to require the addition of the thinnest possible layer in the model training application, querying a central service via HTTPS and minimal REST APIs. A reference implementation with a server instance running on INFN Cloud and a Python client was presented and tested in a real-world application to coordinate hyperparameter optimization campaigns on multiple resource providers including INFN, CERN, and CINECA. In the future we will improve the quality of the Web User Interface, for example enabling custom model documentation and sharing among multiple users, and introduce support to multi-objective optimizations.

Acknowledgments

We would like to thank Doina Cristina Duma and the rest of the INFN Cloud group for the technical support in the deployment and test of HOPAAS. We acknowledge enlightening and motivating discussions with Diego Ciangottini, Stefano Dal Pra, Piergiulio Lenzi and Daniele Spiga, especially on future applications and developments.

This work is partially supported by ICSC – *Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing*, funded by European Union – NextGenerationEU.

References

[1] Ramesh A *et al.* 2021 Zero-Shot Text-to-Image Generation *Proceedings of the 38th International Conference on Machine Learning (PMLR)* vol 139 pp 8821–8831
[2] Brown T *et al.* 2020 Language Models are Few-Shot Learners *Advances in Neural Information Processing Systems (NeurIPS)* vol 33
[3] Richens J G, Lee C M and Johri S 2020 *Nat. Comm.* **11** 3923
[4] Orr G and Müller K 2003 *Neural Networks: Tricks of the Trade* (Springer Berlin Heidelberg)
[5] Bergstra J *et al.* 2011 Algorithms for Hyper-Parameter Optimization *Advances in Neural Information Processing Systems (NeurIPS)* vol 24
[6] Bergstra J, Yamins D and Cox D 2013 Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures *Proceedings of the 30th International Conference on Machine Learning (PMLR)* vol 28 pp 115–123
[7] Golovin D *et al.* 2017 Google Vizier: A Service for Black-Box Optimization *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp 1487–1495
[8] Liaw R *et al.* 2018 Tune: A Research Platform for Distributed Model Selection and Training (*Preprint* `arXiv:1807.05118`)
[9] Akiba T *et al.* 2019 Optuna: A Next-generation Hyperparameter Optimization Framework *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* pp 2623–2631 (*Preprint* `arXiv:1907.10902`)
[10] Head T *et al.* 2021 scikit-optimize (v0.9.0) `https://doi.org/10.5281/zenodo.5565057`
[11] Song X *et al.* 2022 Open Source Vizier: Distributed Infrastructure and API for Reliable and Flexible Black-box Optimization *First Conference on Automated Machine Learning (AutoML)* (*Preprint* `arXiv:2207.13676`)
[12] Barbetti M and Anderlini L 2023 Reference implementation for Hopaas REST APIs Client in Python `https://doi.org/10.5281/zenodo.7528502`
[13] Tani L *et al.* 2021 *EPJ C* **81** 170
[14] Docker Compose `https://docs.docker.com/compose`
[15] Uvicorn `https://www.uvicorn.org`
[16] FastAPI `https://fastapi.tiangolo.com`
[17] Optuna `https://optuna.org`
[18] NGINX `https://www.nginx.com`
[19] PostgreSQL `https://www.postgresql.org`
[20] Chartist `https://gionkunz.github.io/chartist-js`
[21] Spiga D *et al.* 2020 *EPJ Web Conf.* **245** 07020
[22] Albertsson K *et al.* 2018 *J. Phys. Conf. Ser.* **1085** 022008
[23] Wulff E, Girone M and Pata J 2023 *J. Phys. Conf. Ser.* **2438** 012092 (*Preprint* `arXiv:2203.01112`)
[24] Anderlini L *et al.* 2023 *PoS* **ICHEP2022** 233

[25] Barbetti M 2023 Lamarr: LHCb ultra-fast simulation based on machine learning models deployed within Gauss *21st International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT)* (*Preprint* arXiv:2303.11428)
[26] Goodfellow I *et al.* 2014 Generative Adversarial Nets *Advances in Neural Information Processing Systems (NeurIPS)* vol 27 (*Preprint* arXiv:1406.2661)
[27] Anderlini L *et al.* (LHCb) 2023 *J. Phys. Conf. Ser.* **2438** 012130 (*Preprint* arXiv:2204.09947)
[28] Ratnikov F *et al.* 2023 *Nucl. Instrum. Meth. A* **1046** 167591
[29] Mariotti M, Spiga D and Boccali T 2021 *PoS* **ISGC2021** 002