Model Performance Prediction for Hyperparameter Optimization of Deep Learning Models Using High Performance Computing and Quantum Annealing

Juan Pablo García Amboage[1,2,*], *Eric* Wulff[1,**], *Maria* Girone[1], and *Tomás* F. Pena[2]

[1]The European Organization for Nuclear Research, CERN
[2]CiTIUS, Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain

Abstract. Hyperparameter Optimization (HPO) of Deep Learning (DL)-based models tends to be a compute resource intensive process as it usually requires to train the target model with many different hyperparameter configurations. We show that integrating model performance prediction with early stopping methods holds great potential to speed up the HPO process of deep learning models. Moreover, we propose a novel algorithm called Swift-Hyperband that can use either classical or quantum Support Vector Regression (SVR) for performance prediction and benefit from distributed High Performance Computing (HPC) environments. This algorithm is tested not only for the Machine-Learned Particle Flow (MLPF), model used in High-Energy Physics (HEP), but also for a wider range of target models from domains such as computer vision and natural language processing. Swift-Hyperband is shown to find comparable (or better) hyperparameters as well as using less computational resources in all test cases.

1 Introduction

Training and Hyperparameter Optimization (HPO) of Deep Learning (DL) models is often compute resource intensive and calls for the use of large-scale High Performance Computing (HPC) resources as well as scalable and resource efficient Hyperparameter (HP) search and evaluation algorithms [1]. Current state-of-the-art HPO algorithms such as Hyperband [2], ASHA [3], and BOHB [4], rely on a method of early termination. Badly performing trials are automatically terminated to allocate compute resources to more promising ones. Such methods have been successfully applied to optimize Machine-Learned Particle Flow (MLPF), a particle flow reconstruction Neural Network (NN) [5]. Using this technique led to o a reduction of ~44% in the validation loss of MLPF [1].

In this context, performance prediction emerges as a potential approach to accelerate the HPO process. This involves using meta-models, referred to as a performance predictors, that can estimate the performance of a given configuration at a particular epoch by leveraging information from its partial learning curve. By employing performance prediction, it is possible to prioritize the training of the most promising configurations based on their predicted performance, while avoiding the need to fully train configurations with poorer

*e-mail: jugarcia@student.ethz.ch
**e-mail: eric.wulff@cern.ch

predicted performance. Consequently, this approach holds great potential for reducing the time and computational resources required for the HPO process.

This work explores novel techniques based on performance prediction to accelerate the HPO process of MLPF and other NN architectures that leverage the use of HPC resources for training the target model and quantum computing for training the performance predictors. Moreover, a new HPO algorithm is proposed, Swift-Hyperband, that integrates Hyperband with the use of model performance predictors.

2 Related work

Baker et al. [6] demonstrated that Support Vector Regression (SVR) models can effectively serve as performance predictors for various NN architectures. In addition to showing a good predictive capability for performance prediction tasks, SVR models offer the advantage of having negligible training and inference times, even when using a consumer-grade laptop CPU. Hence, using SVRs prevents the training of performance predictors from becoming a bottleneck for the potential resource savings expected from this technique.

In the European Center of Excellence in Exascale Computing "Research on AI- and Simulation-based Engineering at Exascale" (CoE RAISE) the capability of SVRs to predict the loss of MLPF after 100 training epochs in the Delphes dataset [7] was successfully shown [8], achieving R^2 values of around 0.9 when using 25% of the target learning curve as input. Here, R^2 is the so-called coefficient of determination, defined as

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\sum_i e_i^2}{\sum_i (y_i - \bar{y})^2} \tag{1}$$

where y_i is the ground truth for data point i, f_i is prediction i, $\bar{y}$ is the mean of all y_i and e_i is the error of prediction i. Furthermore, the Quantum Annealer at the Jülich Supercomputer Centre, was used to train Quantum Support Vector Regression (QSVR) [9] models on MLPF learning curves. While no significant performance benefit was expected from the use of quantum resources, and the idea of employing quantum computers for the task of performance prediction was primarily a proof of concept of integrating this technology into the HPO workflow, the QSVRs achieved comparable performance to that obtained with classical SVRs [8]. Note that the number of training samples in this case had to be reduced to 20 due to limitations in problem size arising from the current state of quantum technologies.

There are several strategies that can be considered for integrating performance predictors with the HPO process [6]. A straightforward approach is to generate a certain amount of random HP configurations, fully train M configurations and partially train $N > M$ configurations. The final loss of the fully trained configurations and part of their learning curves are used to train a performance predictor. Then, this performance predictor is used to predict the final loss of the partially trained configurations and only those whose predicted loss is below a certain threshold are selected to complete training. This approach was tested using different types of performance predictors, including QSVRs, in [10].

Another, more sophisticated example of the use of performance predictors for HPO is the algorithm Fast-Hyperband [6]. This algorithm is a modified version of the well-known Hyperband algorithm that uses performance predictors which are trained on the fly during the

execution of the algorithm to save training epochs of the target model with respect to Hyperband. More precisely, Fast-Hyperband adds an extra decision point [1] based on performance prediction for every epoch in each Hyperband round. Decisions in these new intermediate points use a probability threshold, computed from an estimate of the standard deviation of every predictor used. The proposed method of computing this estimate in Fast-Hyperband is leave one out cross validation. The main drawback of this approach is that it requires to train many performance predictors, which makes it impractical to use QSVRs. This is partly due to runtime limitations on the quantum machine and partly due to the time needed to formulate the regression problem in a suitable way for the quantum annealer. In addition, the time spent to connect to the quantum machine needs to be taken into account. Furthermore, Fast-Hyperband, as it is defined in [6], is a sequential algorithm which makes it unable to benefit from running in a distributed manner on multiple nodes in an HPC environment.

3 Swift-Hyperband

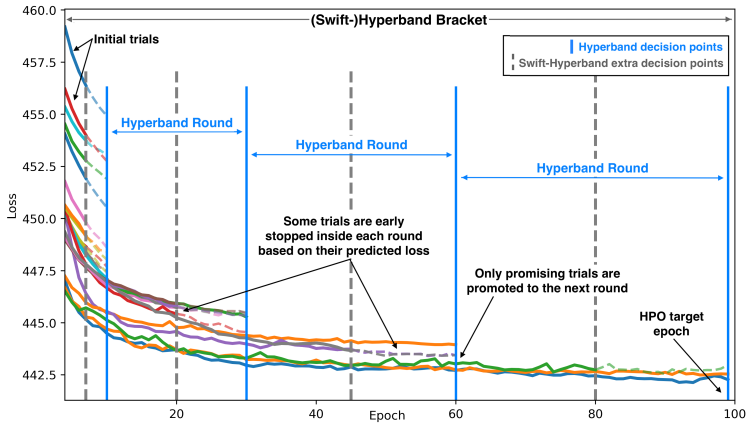


Figure 1: Graphic representation of a Swift-Hyperband bracket. This is an illustrative example, not the result of an actual execution the algorithm.

To overcome the limitations of Fast-Hyperband mentioned in the previous section we propose a different way to integrate performance predictors with Hyperband by a new algorithm called Swift-Hyperband. Similarly to Fast-Hyperband, Swift-Hyperband adds extra decision points to the classical Hyperband algorithm that are based on performance prediction. However, Swift-Hyperband adds only one extra decision point per Hyperband round. When a round starts, Swift-Hyperband trains a certain number of trials until the end of the round and then uses their loss at the end of the round to define a threshold. The other trials in the round are partially trained to the extra decision point, where the performance predictors are used to predict their losses at the end of the round. These predictions are compared to the previously defined threshold to decide if each trial should be terminated or not.

Figure 1 illustrates what happens inside each bracket of Swift-Hyperband. The vertical dashed lines represent the new decision points in which some configurations or trials are

[1] We understand a decision point as a certain epoch in which an early stopping HPO algorithm discards some configurations to keep training only the most promising ones according to a certain criterion.

discarded based on their predicted performance. The trials that did not complete their round are represented using dashed learning curves. That is, the dashed learning curves represent training epochs of the target model that Swift-Hyperband saved with respect to the classical Hyperband algorithm. The trials that complete their training until the end of the round are the ones that are always represented by continuous lines. The decision of which of these trials are promoted to the next round is made as it is made in the classical Hyperband algorithm.

Swift-Hyperband not only needs to train less performance predictors than Fast-Hyperband but can also be easily parallelized, as all the initial full and partial trainings inside a round can be executed in parallel. For these two reasons, our algorithm can potentially use QSVRs and benefit from HPC environments. A schematic comparison between Fast-Hyperband and Swift-Hyperband is shown in Figure 2.

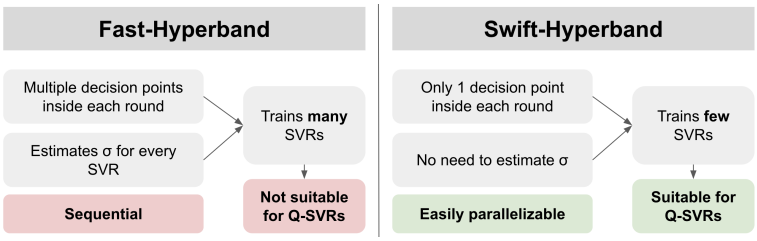


Figure 2: Key differences between Fast-Hyperband and Swift-Hyperband.

4 Results

To compare Hyperband, Fast-hyperband, Swift-Hyperband and Quantum-Swift-Hyperband (Swift-Hyperband using QSVRs) for different NN architectures we simulate 10 runs of each algorithm using the datasets of learning curves derived from the following model-dataset combinations:

- MLPF [8] trained on the Delphes dataset [7].
- An image recognition Convolutional Neural Network (CNN) modified from [3] trained on the Cifar10 dataset.
- An image recognition CNN trained on the SVHN dataset used in [6].
- A natural language processing Long Shot-Term Memory (LSTM) NN trained in the PTB dataset [11].

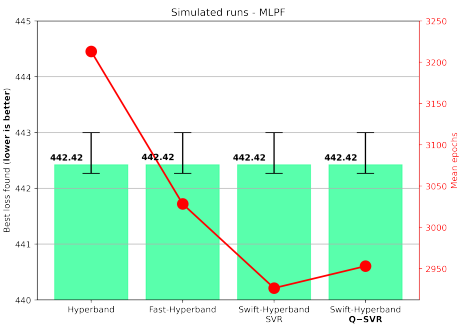
The result of these simulated runs can be seen in Figure 3 and a summary of the learning curve datasets used for the simulation is available in Table 1.

Beyond the simulated runs, we test the speedup provided by the parallelization of Swift-Hyperband along with the achieved accuracies by running Hyperband, Fast-Hyperband, Swift-Hyperband, and a parallel version of Swift-Hyperband that uses MPI to coordinate one CPU node and two GPU worker nodes. For these runs, the HPO target was a simple 6-layer CNN (different to the CNN used in the simulated runs) trained on Cifar10 using a 3-dimensional search space consisting of learning rate, weight decay, and dropout. This network was chosen because it was relatively fast to train. The results can be seen in Figure 4.

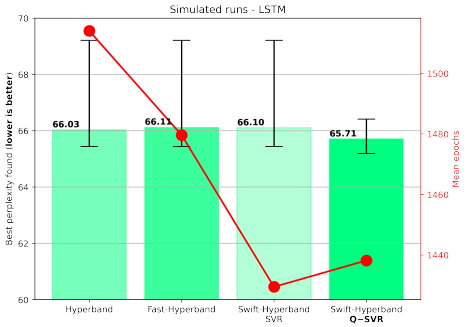
The results in Figure 3 and Figure 4 show that both Swift-Hyperband and its version using QSVRs achieve accuracies comparable to classical Hyperband while needing considerably

Neural Network	Evluation metric	HPs search space dimension	Target epoch for HPO
MLPF for Delphes	Loss	7	100
LSTM for PTB	Perplexity	2	60
CNN for Cifar10	Accuracy	5	100
CNN for SVHN	Accuracy	9	100

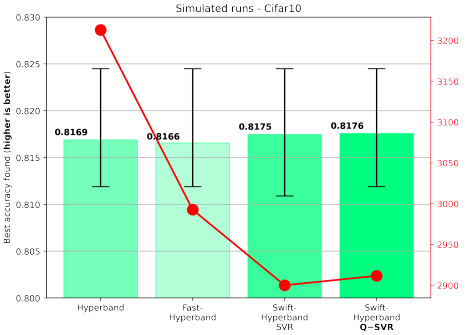
Table 1: Summary of learning curve datasets.



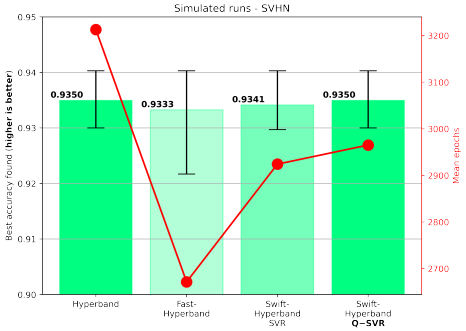
(a) MLPF trained on Delphes



(b) LSTM trained on PTB



(c) CNN trained on Cifar10



(d) CNN trained on SVHN

Figure 3: Average, best, and worst performance of the best configuration found as well as the total number of epochs consumed by the different HPO algorithms for different NNs and datasets.

fewer epochs in all cases. In comparison to Fast-Hyperband, Swift-Hyperband (SVR and Q-SVR) is faster in all cases except on the SVHN problem. When it comes to the non-simulated runs we observe that all algorithms achieve accuracies around 87%, with both Swift-Hyperband and Parallel-Swift-Hyperband slightly beating Fast-Hyperband. Note that in several cases the version of Swift-Hyperband that uses QSVRs finds better performing configurations than the original Hyperband algorithm, something that would not happen if the QSVRs made perfect predictions. This may indicate that using performance predictors,

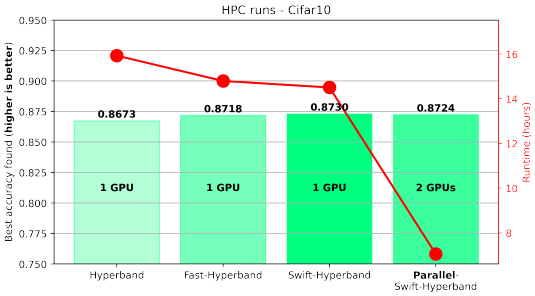


Figure 4: Performance of the best configuration found and total runtime needed for different HPO algorithms.

aside from saving compute resources, has some type of regularizing effect that prevents some of the errors that Hyperband makes when terminating configurations at the end of each round.

5 Conclusions

We proposed a new promising parallelizable HPO algorithm integrating Hyperband and performance predictors that can be used in combination with SVRs or QSVRs. This leaves the door open for the use of Swift-Hyperband in later HPO cycles of MLPF. Furthermore, it was shown that, despite the current limitations of quantum computers, it is possible to execute hybrid Quantum/HPC workflows for HPO, achieving comparable performance to fully classical workflows. We consider that there is a need for further studies on the speedup achieved by the parallelization of Swift-Hyperband when using a greater number of nodes as Hyperband is known to suffer from straggler issues [3]. Hence, developing a version of the HPO algorithm ASHA that integrates performance predictors, potentially to be named Swift-ASHA, is one of the identified lines to continue this work. In addition to this, conducting more empirical tests of Swift-Hyperband on a wider range of target models would provide valuable insights on the behavior of the algorithm. Finally, conducting theoretical studies on SVRs and QSVRs could provide a deeper understanding of why Swift-Hyperband occasionally outperforms the original Hyperband algorithm and shed light on the differences observed when employing quantum or classical SVRs.

Acknowledgements

Eric Wulff and Juan Pablo García Amboage was supported by CoE RAISE. The CoE RAISE project has received funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

The authors gratefully acknowledge the computing time granted through JARA on the supercomputer JURECA [12] at Forschungszentrum Jülich. The authors gratefully acknowledge the Jülich Supercomputing Centre for funding this project by providing computing time through the Jülich UNified Infrastructure for Quantum computing (JUNIQ) on the D-Wave Advantage™ System JUPSI.

References

[1] E. Wulff, M. Girone, J. Pata, Journal of Physics: Conference Series **2438**, 012092 (2023)

[2] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, The Journal of Machine Learning Research **18**, 6765 (2017)

[3] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, A. Talwalkar, CoRR **abs/1810.05934** (2018), 1810.05934

[4] S. Falkner, A. Klein, F. Hutter, *BOHB: Robust and efficient hyperparameter optimization at scale*, in *International Conference on Machine Learning* (PMLR, 2018), pp. 1437–1446

[5] J. Pata, J. Duarte, J. Vlimant, M. Pierini, M. Spiropulu, The European Physical Journal C **81** (2021)

[6] B. Baker, O. Gupta, R. Raskar, N. Naik, *Accelerating neural architecture search using performance prediction* (2017), https://arxiv.org/abs/1705.10823

[7] J. Pata et al., *Simulated particle-level events of $t\bar{t}$ and QCD with PU200 using PYTHIA8+DELPHES3 for machine learned particle flow (MLPF)* (2021), https://zenodo.org/record/4559324

[8] E. Wulff, M. Girone, D. Southwick, J.P.G. Amboage, E. Cuba, *Hyperparameter optimization, quantum-assisted model performance prediction, and benchmarking of ai-based high energy physics workloads using hpc* (2023), 2303.15053

[9] E. Pasetto, M. Riedel, F. Melgani, K. Michielsen, G. Cavallaro, IEEE Geoscience and Remote Sensing Letters **19**, 1 (2022)

[10] M. Aach, E. Wulff, E. Pasetto, A. Delilbasic, R. Sarma, E. Inanc, M. Girone, M. Riedel, A. Lintermann, *A hybrid quantum-classical workflow for hyperparameter optimization of neural networks* (2023)

[11] M.P. Marcus, M.A. Marcinkiewicz, B. Santorini, Comput. Linguist. **19**, 313–330 (1993)

[12] Jülich Supercomputing Centre, Journal of large-scale research facilities **7** (2021)