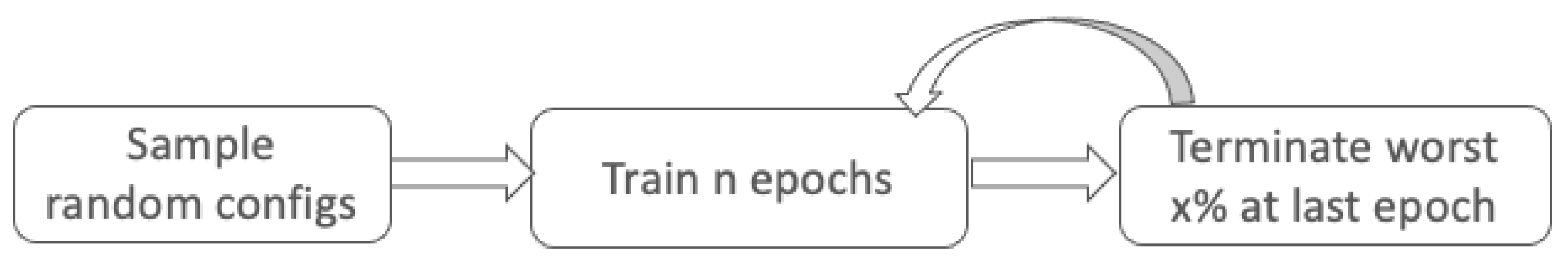
Hyperparameter optimization, multi-node distributed training and benchmarking of AI-based HEP workloads using HPC

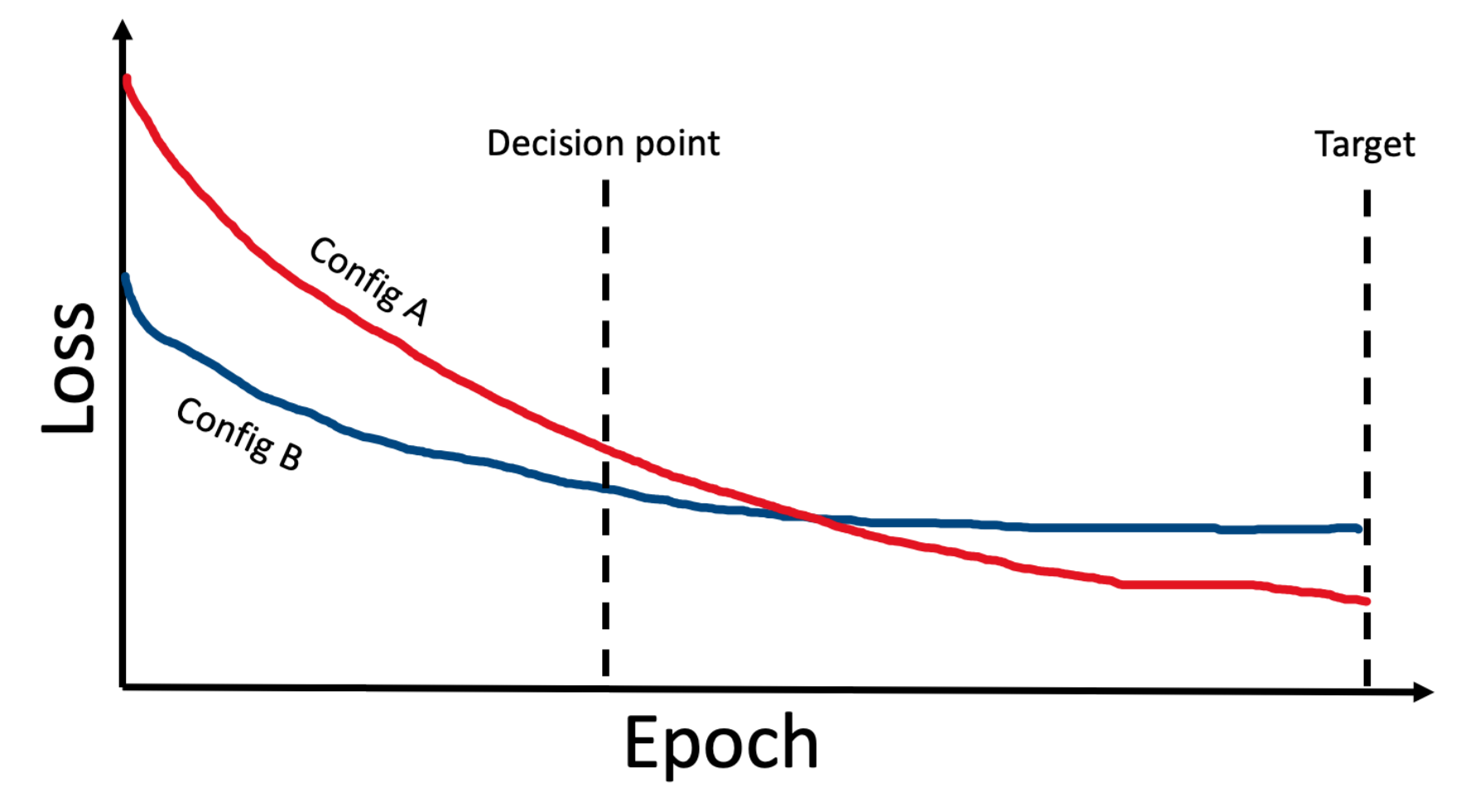
E. Wulff*, M. Girone, D. Southwick, E. Cuba, J.P. García Amboage
CERN, 1211 Geneva 23, Switzerland,
CoE RAISE, Wilhelm-Johnen-Straße, 52428 Jülich, Germany
*eric.wulff@cern.ch

1. Hyperparameter optimization

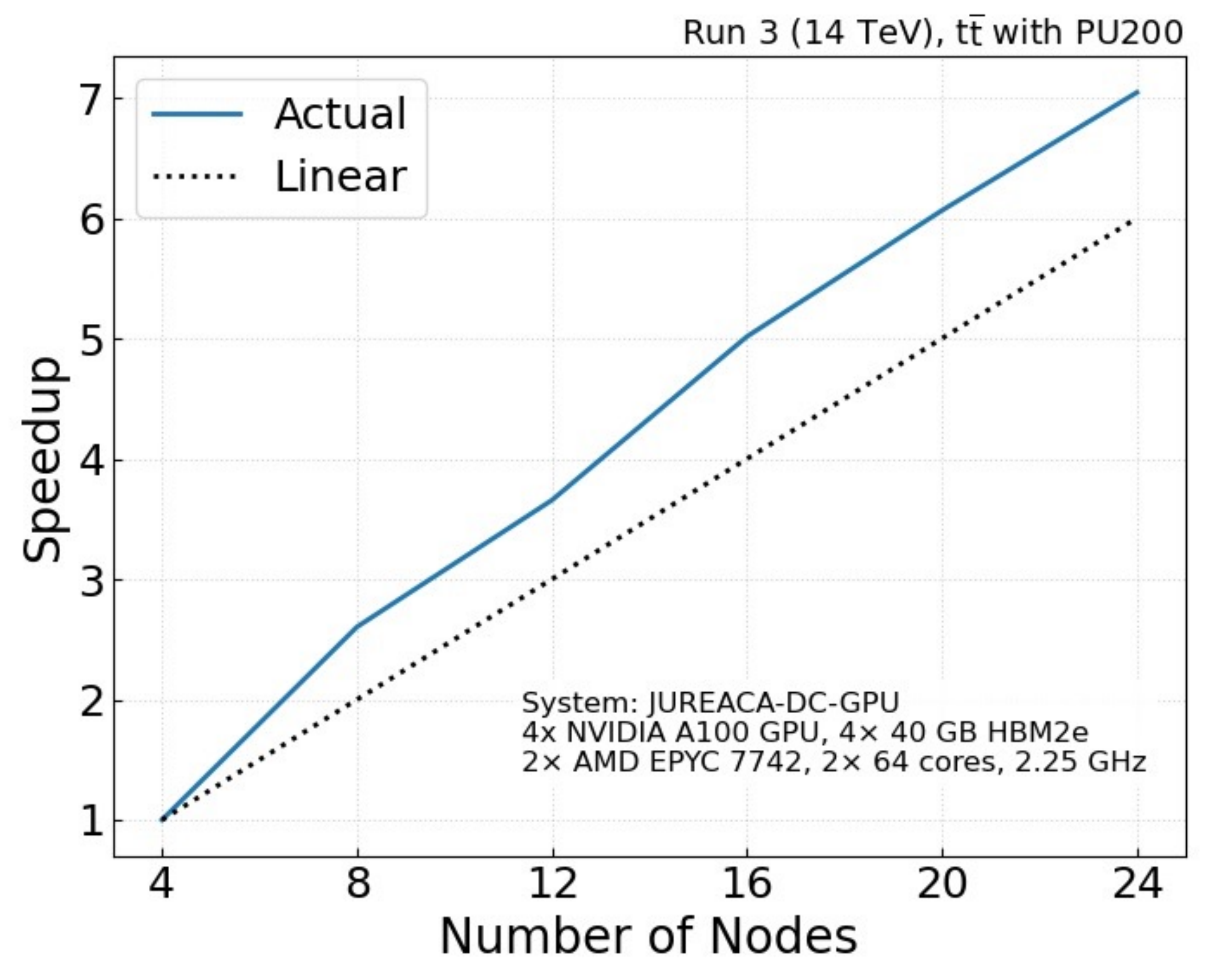
Training and hyperparameter optimization (HPO) of DL-based AI models is often compute resource intensive and calls for the use of large-scale distributed resources as well as scalable and resource efficient HPO algorithms [1]. Current state-of-the-art HP search algorithms such as Hyperband [2], ASHA [3] and BOHB [4] rely on a method of early termination, where badly performing trials are automatically terminated to free up compute resources for new trials to be started.



The stopping criterion, based on the ranking of trials according to a chosen metric, usually accuracy or loss, can be problematic. Since the training process is non-linear, the ranking of trials at the decision point does not necessarily hold at the target point. A potential solution to this problem is to use a non-linear stopping criterion, e.g. using Support Vector Regression (SVR) to predict the final model performance from a partially trained model [5].



HPO using algorithms such as ASHA is especially suited for large-scale HPC since greater than linear scaling can be achieved, as seen in tests at the Jülich Supercomputer Centre (JSC).



2. Quantum-SVR for model performance prediction

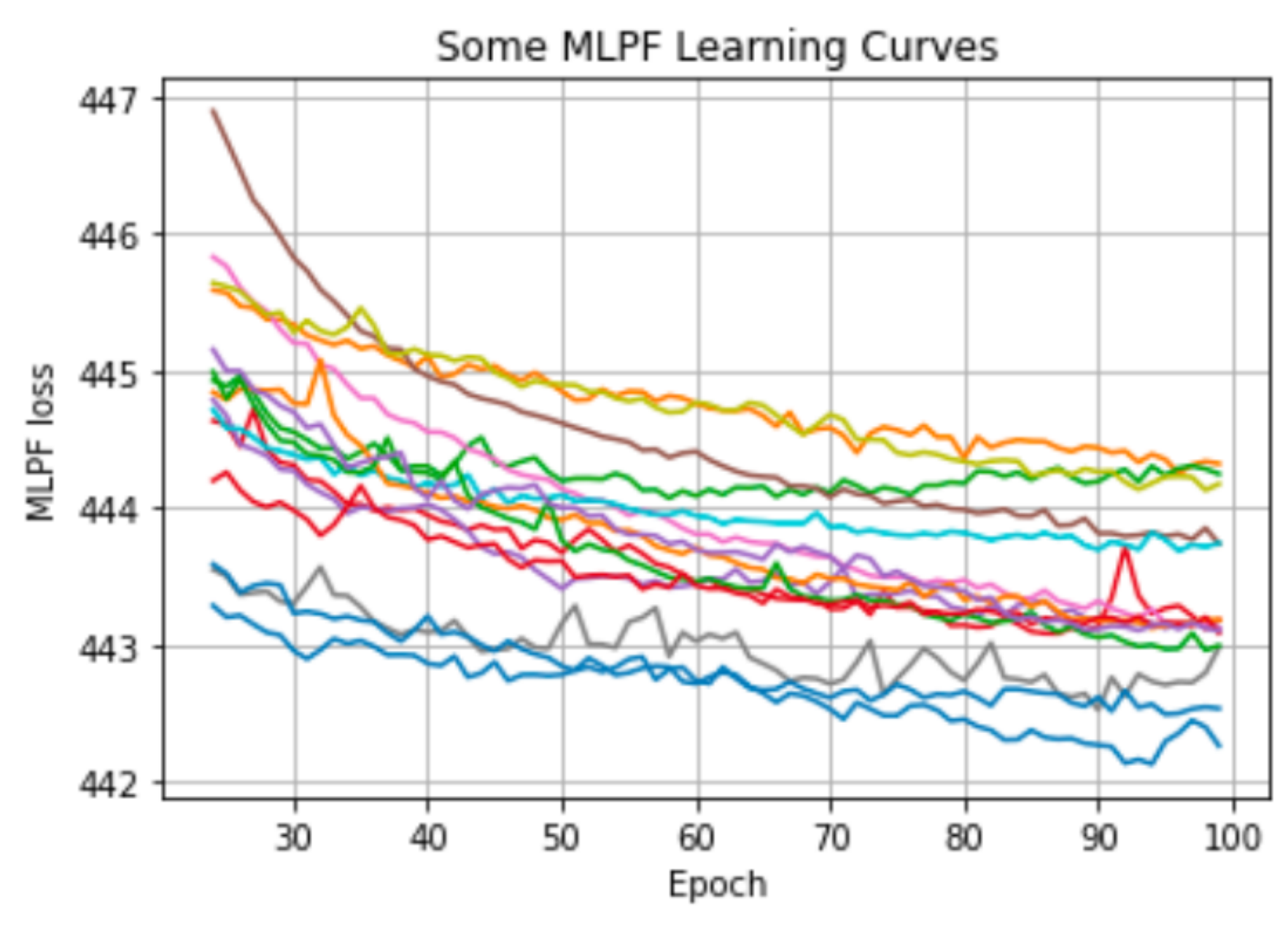
The potential to speed up the HPO process via performance prediction as well as the use of a quantum annealer (QA) to train the performance predictor is investigated. QSVR performance comparable to a classical SVR is obtained.

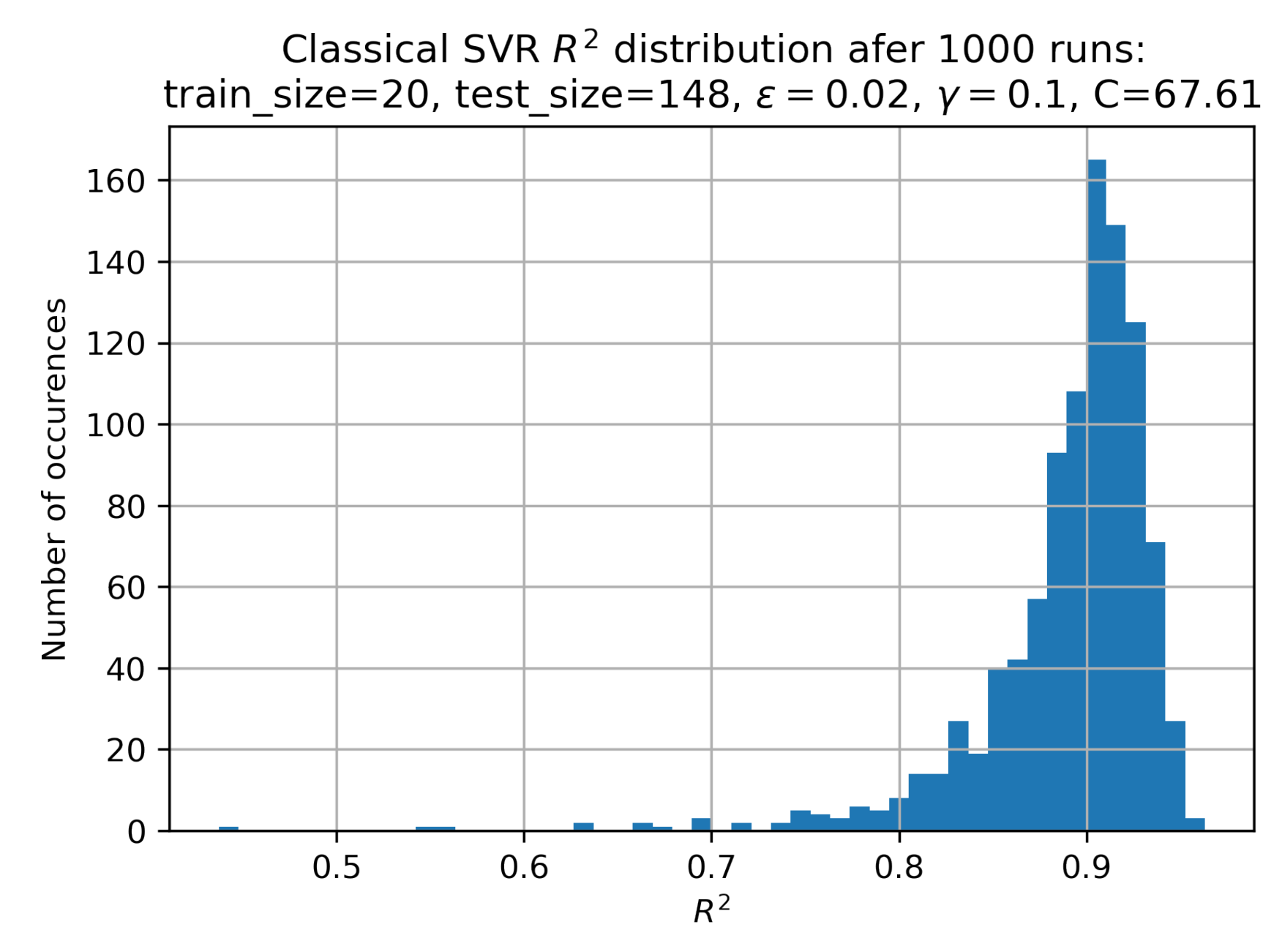
A GNN-based algorithm, developed for the task of machine learned particle flow reconstruction in HEP [6, 7], acts as the base model for which studies are performed. A dataset consisting of learning curves and HP configurations was generated by training 296 different configurations of MLPF on the publicly available Delphes dataset [8].

Hyperparameter space for dataset generation

	learning rate	num graph layers id	num graph layers reg	dropout	bin size	output dim	weight decay
type	log uniform	quantized uniform	quantized uniform	uniform	choice	choice	log uniform
range	(1e-6, 3e-2)	[0, 4]	[0, 4]	(0.0, 0.5)	[8, 16, 32, 64, 128]	[8, 16, 32, 64, 128, 256]	(1e-6, 1e-1)

To establish a baseline, classical SVR was studied by fitting 1000 models on different train/test splits with the same constraint on dataset size as required by the QSVR training of 20 samples. Results vary depending on the training split and statistics are shown in the figure and table below.

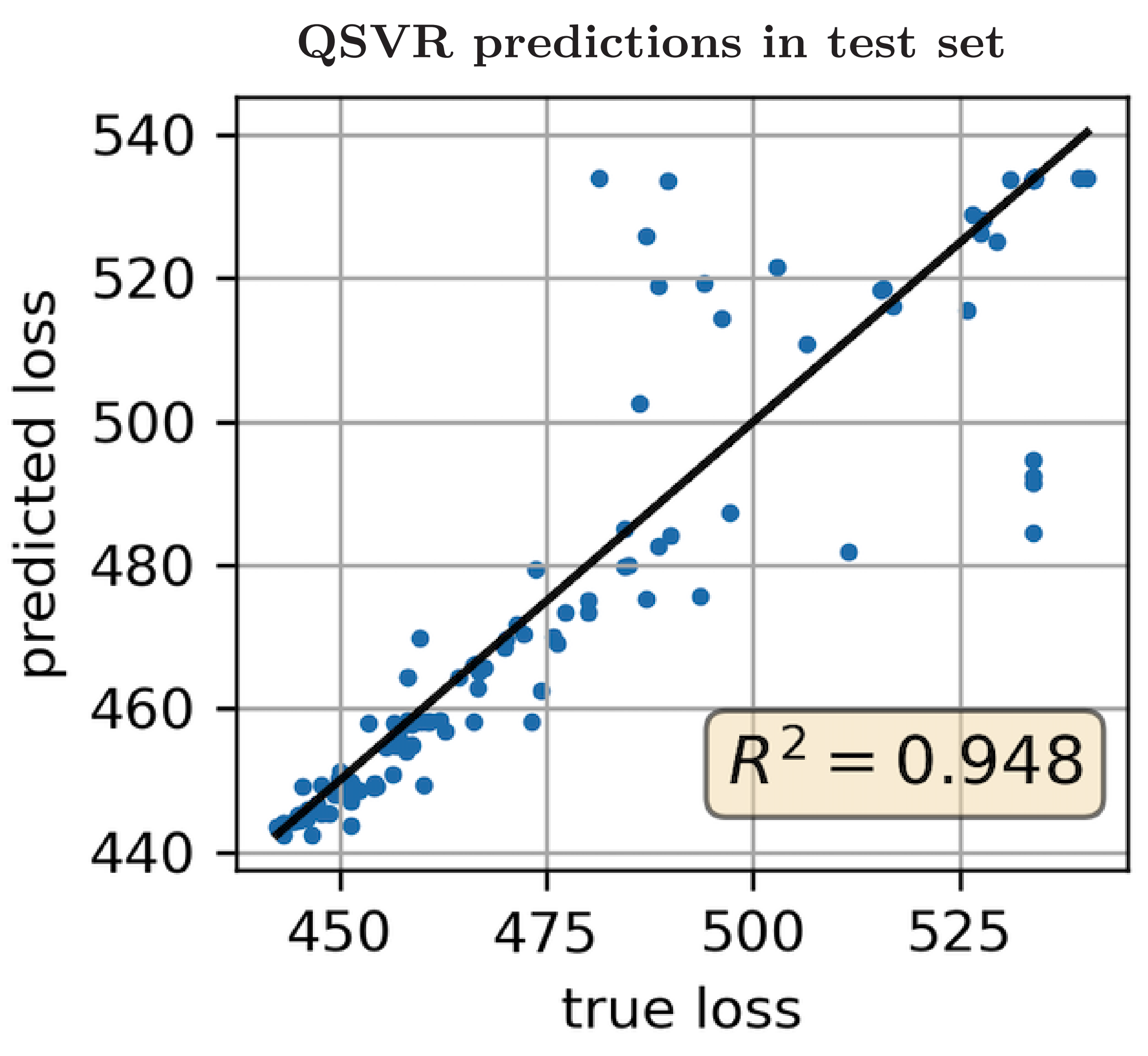




Via CoE RAISE, access to JUPSI, a QA at the Jülich Supercomputer Centre, was leveraged to train a QSVR model for performance prediction. Due to the probabilistic nature of quantum processes the annealing is run multiple times and returns multiple solutions which can then be combined in several different ways to crate the final QSVR model.

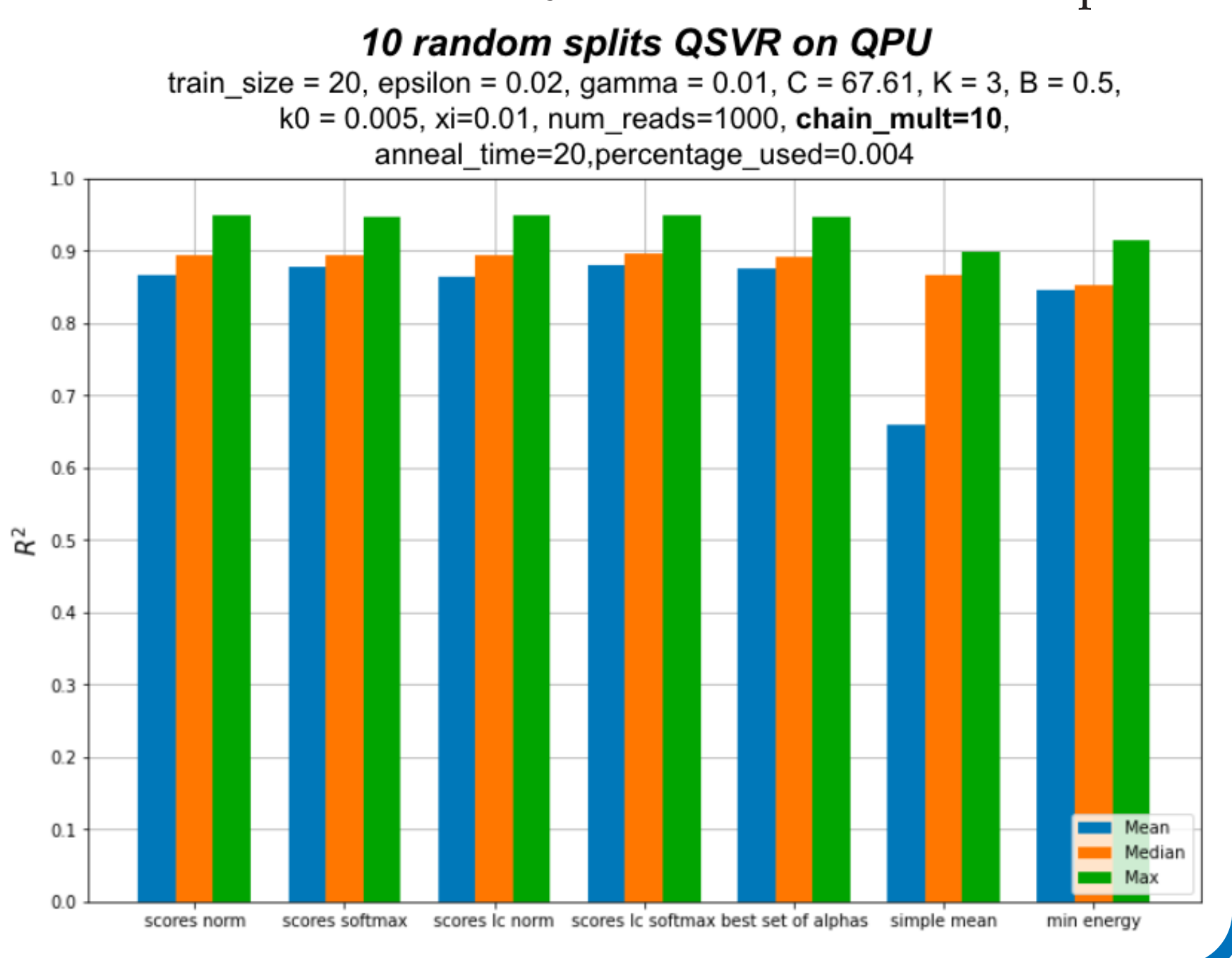
R^2 scores

	Best	Worst	Mean	Std	Number of trainings
SVR	0.959	0.318	0.889	0.050	1000
Sim-QSVR	0.949	0.383	0.901	0.045	100
QSVR	0.948	0.742	0.880	0.056	10
QSVR Ensemble	0.927	0.857	0.899	0.019	10



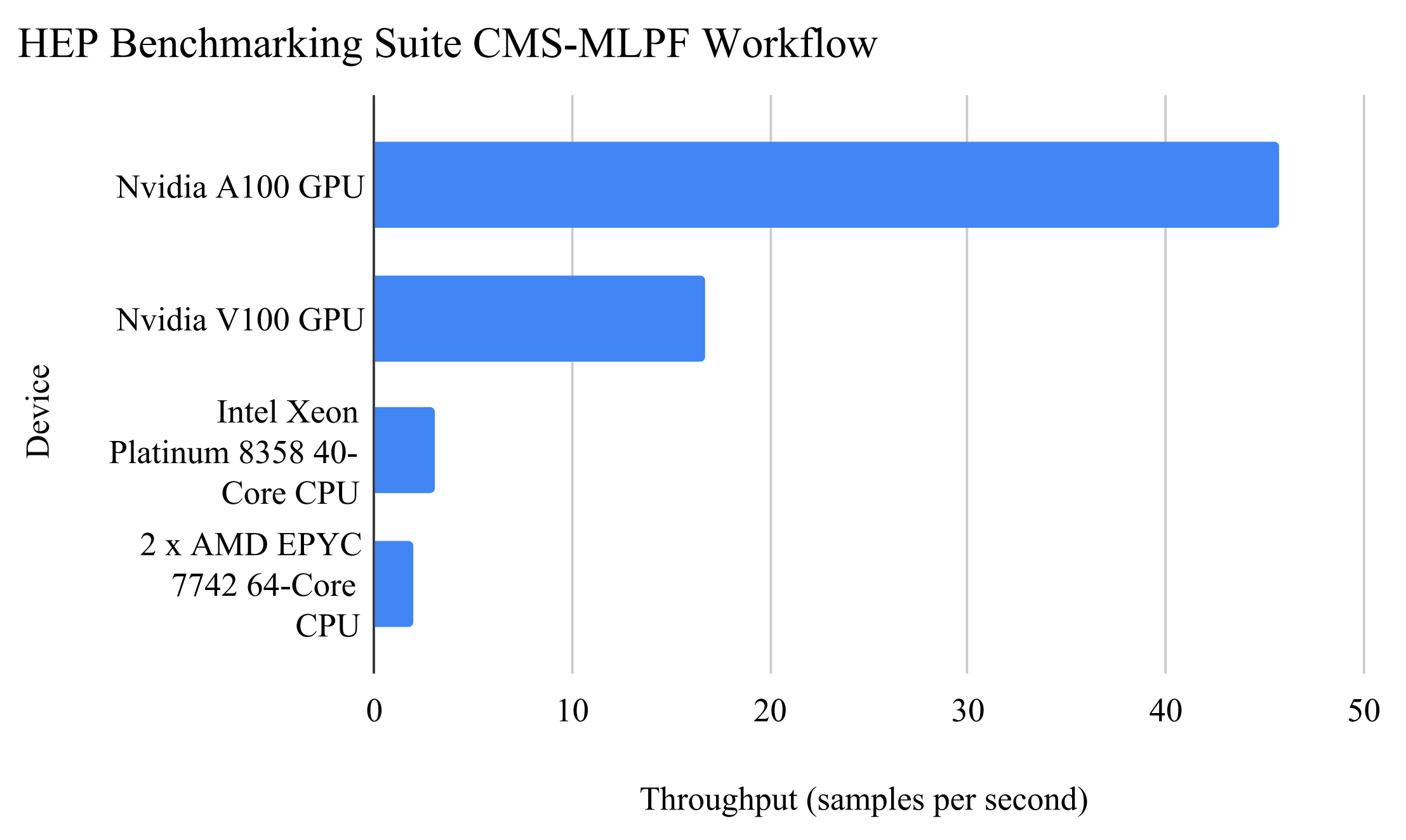
The predictions of the best performing QSVR are plotted against the true values in the figure above. A summary of the performance of different QSVR combination methods is show in the figure to the right. The methods used are described in [9]. Results are comparable to classical SVR models when trained on datasets of the same size.

With the aim of stabilizing QSVR performance, as well as to increase the effective training set size, four separate QSVR models were trained on disjoint training sets of 20 samples each. Although this approach did not improve the maximum R^2 score, it did produce more stable results by significantly improving the worst performing split and reducing the standard deviation of R^2 scores between splits.



3. Containerized AI benchmark

The promise of better accuracy and reconstruction scalability at inference time for AI-based algorithms is preceded by resource-intensive training. We propose a self-contained, reproducible benchmark based on the training of DL models to explore the feasibility of deploying AI-driven HEP applications to different HPC environments. The metric shown is the training throughput on a subset of the public Delphes dataset [8] in HEPscore [10] format.



4. Conclusions

HPC systems are essential for large-scale hypertuning and distributed training and can significantly increase model performance as well as speed up the iteration of model development and training. We have seen that, in this case of hypertuning using Ray Tune, greater than linear scaling in speed-up to number of nodes can be achieved, indicating the benefits of HPC for such use-cases. The current work is limited by resource constraints imposed by supercomputer centers on the maximum number of compute nodes that individual users are allowed to access at any one time.

The strong potential of using performance prediction techniques for HPO was demonstrated, leaving the door open for the use of this technique in later HPO studies. It was also shown that, despite the current limitations of quantum computers, it is possible to train SVR models on a quantum annealer while achieving prediction performance comparable to those obtained on a classical SVR. This encourages further studies in utilizing hybrid Quantum/HPC workflows for HPO as well as in other use-cases.

Finally, the development of a containerized benchmark application with an AI use-case from HEP allows for quick and easy benchmarking of new hardware accelerators in the HEPScore format.

5. References

[1] E. Wulff, M. Girone, and J. Pata. Hyperparameter optimization of data-driven ai models on HPC systems, 2022.
[2] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. 2016.
[3] Liam Li, Kevin G. Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. *CoRR*, abs/1810.05934, 2018.
[4] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. *CoRR*, abs/1807.01774, 2018.
[5] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction, 2017.
[6] J. Pata, J. Duarte, JR. Vlimant, M. Pierini, and M. Spiropulu. MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *The European Physical Journal C*, 81(5), may 2021.
[7] J. Pata, J. Duarte, F. Mokhtar, E. Wulff, J. Yoo, JR. Vlimant, M. Pierini, and M. Girone. Machine learning for particle flow reconstruction at CMS, 2022.
[8] 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.
[9] Edoardo Pasetto, Morris Riedel, Farid Melgani, Kristel Michielsen, and Gabriele Cavallaro. Quantum svr for chlorophyll concentration estimation in water with remote sensing. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.
[10] Domenico Giordano et al. HEPiX Benchmarking Solution for WLCG Computing Resources. *Comput. Softw. Big Sci.*, 5(1):28, 2021.