H2020-INFRAEDI-2018-2020

Benchmarking & support report

Table of Contents

List of Figures

List of Tables

Executive Summary

The benchmarking and support activities in the European Center of Excellence in Exascale Computing “Research on AI- and Simulation-Based Engineering at Exascale” (CoE RAISE) are in part directed towards working on latest disruptive technologies such as on Quantum Computing (QC) systems. This Deliverable reports the initial developments on this front which have been made possible through the commissioning of the D-Wave Advantage™ Quantum Annealer (QA) System JUPSI at Forschungszentrum Jülich (FZJ) at the beginning of 2022. This document highlights the QA-related work jointly performed in a collaboration between the CoE RAISE partners FZJ, the European Center for Nuclear Research (CERN), and the University of Iceland (UOI). The QA is used to solve classification and regression problems in a Quantum Support Vector Machine/Regression (Q-SVM/R) framework in a hybrid computing workflow. The applications cover Hyperparameter Optimization (HPO) in Machine Learning (ML) workflows and multiclass classification for a Remote Sensing (RS) case. The QA-based performance is benchmarked against classical computations. It is seen that the Quantum version of the SVM provides comparable (and at present slightly worse) performance in relation to its classical counterpart. It is important to note that the QA predictions are noisy and more research is required to find ways to provide consistent predictions. Furthermore, strategies to improve the performance of the QA have been proposed. This Deliverable shows the potential of employing QA in a hybrid computing setup and provides a viable technique of inclusion of a QA-based routine in ML workflows. QC technologies are still at an early stage and this work identifies many opportunities for further research in this direction in the context of CoE RAISE and other projects.

1 Introduction

The European Center of Excellence in Exascale Computing "Research on AI- and Simulation-Based Engineering at Exascale" (CoE RAISE) intends to exploit various Modular Supercomputing Architectures (MSAs) for the different use-case applications. This has already been demonstrated in previous Deliverables with developments in the compute-driven and the data-driven use cases on both prototype and production systems. For instance, Deliverable D2.7 from M18 reports on both Artificial Intelligence (AI)-based and domain codes that have been tested on different High-Performance Computing (HPC) systems with repect to their performance. These systems consist of classical (digital) computing devices such as Central Processing Units (CPUs) and Graphical Processing Units (GPUs). However, CoE RAISE is also interested in testing other disruptive and state-of-the-art computing systems such as Quantum Computing (QC)-based systems. In this regard, Task 2.3 *"Benchmarking on Disruptive Technologies"*, led by Forschungszentrum Jülich (FZJ) within Work Package (WP) 2 *"AI- and HPC-Cross Methods at Exacale"*, is working on a Quantum Annealer (QA). The developments in Task 2.3, which commenced in M19, are reported in this Deliverable.

Quantum computers are devices that aim to exploit the Quantum mechanical properties of matter such as superposition, interference, and entanglement, to perform computation. In many cases, Quantum computers can solve problems significantly faster than their classical counterparts. Currently, there are two main types of QC systems: the gate-based and the Quantum Annealing-based. On the one hand, gate-based Quantum computers have Quantum gates that, in a similar fashion to boolean gates on regular computers, can solve any type of optimization problem with the correct mapping. QA systems, on the other hand, are only suitable for tasks that can be reformulated as energy-minimization problems. By cooling the QA close to absolute zero temperature, the machine automatically minimizes the energy and therefore finds solutions to the original problem.

Figure 1: Jülich Unified Infrastructure for Quantum Computing (JUNIQ)[1].

[1] JUNIQ https://www.fz-juelich.de/ias/jsc/EN/Expertise/JUNIQ/_node.html

In the context of CoE RAISE, the partners can request access to resources on the first QA system in Europe, which is a D-Wave Advantage™ system named Jülich Pioneer for Spin Interference (JUPSI)[2]. The system is operational at FZJ since the beginning of 2022 and features more than 5,000 qubits, the largest number of qubits currently available in any Quantum machine. It is part of the Jülich UNified Infrastructure for Quantum computing (JUNIQ) (shown in Figure 1), which provides German and European researchers access to and support for various Quantum systems[3]. JUNIQ is a manufacturer-independent, comprehensive QC user facility integrated in the Jülich Supercomputing Centre (JSC). JUNIQ will offer access to various computing systems that emulate or directly exploit Quantum effects. The existing QC emulator JUQCS can emulate ideal "pen-&-paper" Quantum computers with up to 43 qubits on the Jülich Wizard for European Leadership Science (JUWELS)[4] system, while the 30-qubit Atos Quantum Learning Machine simulates gate-based Quantum computers with up to 31 qubits. Furthermore, the Pasqal[5] Quantum simulator JURY, to be implemented in JUNIQ in mid 2023, will enable the simulation of Quantum spin systems and the Heisenberg model. Besides, three variants of gate-based Quantum systems are candidates in JUNIQ and access will also be provided to early-stage experimental systems. At the moment, the JUNIQ platform provides access to the Jülich JUPSI machine.

The project partners from FZJ and the European Organization for Nuclear Research (CERN) filed a proposal to the Jülich Aachen Research Alliance (JARA)[6] computing project call[7] that opened in May 2022. The proposal was scientifically and technically peer reviewed and the project was granted 4.8 hours on the JUPSI machine. The compute time project is directed towards working on a hybrid computing-based workflow, where the QA is used to perform Hyperparameter Optimization (HPO) together with a classical CPU/GPU-based HPC system. Furthermore, the partners at University of Iceland (UOI) have been exploiting the QA for a Remote Sensing (RS) use case in a classification as well as in a regression problem. This Deliverable presents the initial results of these developments in terms of benchmarking efforts, and highlights the collaborations among the project partners with respect to Task 2.3. The Deliverable is intended towards researchers and industrial partners who are new to QC in general and provides ideas to include QC in their AI-workflows. Furthermore, this document also highlights the opportunities for other partners to access the JUNIQ platform at FZJ.

The Deliverable is structured as follows. Section 2 presents the work at FZJ in collaboration with UOI on an RS dataset and a cifar-10[8] dataset using a Quantum Support Vector Regression (Q-SVR) framework. In Sec. 3, the QA activities at CERN, where HPO is applied to a High-Energy Physics (HEP) use case, is discussed. A Quantum Multiclass Support Vector Machine (QMSVM), which is based on the work at UOI, is presented in Sec. 4. Finally, Sec. 5 draws the main conclusions of this Deliverable along with summarizing the current developments.

[2] JUPSI https://www.fz-juelich.de/en/ias/jsc/systems/Quantum-computing/juniq-facility/juniq/d-wave-advantagetm-system-jupsi
[3] JUNIQ projects https://www.fz-juelich.de/en/ias/jsc/systems/Quantum-computing/juniq-facility/projects-and-partners
[4] JUWELS https://www.fz-juelich.de/en/ias/jsc/systems/supercomputers/juwels
[5] PASQAL https://pasqal.io/
[6] JARA https://www.jara.org/en/
[7] JARA compute project call https://www.jara.org/en/research/center-for-simulation-and-data-sciences/jara-partition/applying-for-computing-time
[8] CIFAR https://www.cs.toronto.edu/~kriz/cifar.html

2 Benchmarking and support activities at FZJ

HPO is a vital approach in ML, which enables to efficiently retrieve the optimal set of hyperparameters for the training of an ML-model with the least error in reconstruction and the fastest convergence. This process is computationally intensive and hence the use of hybrid computing approaches can be promising to make this process efficient. In CoE RAISE, the use of QA for HPO acceleration is investigated across various use cases. To involve the QA in the computation of larger problems, it has to be combined with a classical supercomputer in a hybrid approach. This concept is benchmarked in Task 2.3. More specifically, this task investigates possible advantages of the Q-SVR method to perform HPO.

In the following, the general concept of QA is introduced in Sec. 2.1, while the modifications necessary to fit a Support Vector Machine (SVM) method onto a QA are explained in Sec. 2.2. Finally, the preliminary results of the hybrid HPO approach are presented in Sec. 2.3.

2.1 Quantum annealing

As introduced in Sec. 1, the JUNIQ platform provides access to the QA JUPSI from D-Wave (5,000 qubits), which feature an order of magnitude more qubits than current gate-based Quantum machines (433 qubits, IBM). Hence, they are more suitable for tackling real-world applications. The Quantum Processing Unit (QPU) from D-Wave is a lattice of tiny metal loops. When temperatures are sufficiently low, they become superconductors and exhibit Quantum-mechanical effects.

On a QA, well-suited optimization problems can be computed much faster than on a classical computer. However, the size of problems that can be solved on a QA is still small. To solve a problem on the QA, it has to be written in the form of a Quadratic Unconstrained Binary Optimization (QUBO) problem which is defined as the minimization of the energy function

$$E = \sum_{i \leq j} a_i Q_{ij} a_j,$$

where $a_i \in \{0,1\}$ are the (binary) variables of the optimization problem and Q_{ij} is the QUBO weight matrix. This formulation is equivalent to the Ising problem on which the QA problem is defined [1].

The parameters of the QUBO model are then mapped onto the qubits and the edges (couplers) connecting the qubits of the QPU. This process is called minor embedding. Often, it is necessary to represent a single logical qubit with several physical qubits which are referred to as a chain of qubits. To be able to map the values of the qubits back to the original problem domain, the values of all qubits in a chain should be the same after the annealing cycle. This consistency is enforced by a chain strength parameter. It must be chosen with care as a too-low value would result in many broken chains, whereas a too-high value degrades the problem representation of the QUBO.

2.2 Quantum Support Vector Regression

As the QA can only solve optimization problems expressed as QUBOs, advanced ML methods such as Deep Learning (DL) algorithms are not feasible for use on the QA. Among the promising techniques are SVM methods that can be employed for classification or regression tasks.

A method for training SVMs using the D-Wave 2000Q QA has already been introduced and compared with the classical SVM version [2]. Results indicate that the Q-SVM method can perform well if the training data is limited. Ensembles of Q-SVMs have been employed to classify images from the RS domain [3]. A comparative study of gate-based QC systems and QAs suggests that QAs can solve larger problems than classical QC machines, and are therefore more interesting for applications from the RS domain [4].

While all of these referred works focus on classification, there are also approaches to use Quantum algorithms for regression tasks, e.g., for linear and ridge regression [5]. SVMs can also be adapted to perform regression in the form of Support Vector Regression (SVR) [6].

The algorithm takes as input a set of samples formed by a pair of one or more independent variables and one dependent variable. The objective of the training phase is to learn the relationship between these variables. The training phase amounts to an optimization problem in which the objective is the determination of the coefficients that define the estimated regression function.

For a given training set (x_n, y_n), where x_n is the input vector and y_n is the target vector, the formulation of the SVR optimization problem is given by

$$y(x) = \sum_{n=1}^{N} (w_n - \hat{w}_n)\, k(x_n, x) + b,$$

where $w_n, \hat{w}_n$, and b are the parameters that are calculated during the optimization process, k is the Radial Basis Function (RBF) kernel, which is chosen in this case to introduce non-linearity, N is the number of samples, and n is the running index. To reformulate the problem in the QUBO form, it is written in the Lagrangian form with the Lagrangian multipliers $\alpha, \hat{\alpha}$ as

$$L(\alpha, \hat{\alpha}) = \frac{1}{2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} (\alpha_n - \hat{\alpha}_n)(\alpha_m - \hat{\alpha}_m)\, k(x_n, x_m) - \epsilon \sum_{n=0}^{N-1} (\alpha_n + \hat{\alpha}_n) + \sum_{n=0}^{N-1} (\alpha_n - \hat{\alpha}_n)\, t_n,$$

with the constraints

$$\sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) = 0,$$

$$0 \leq \alpha_n \leq C, \text{ and}$$

$$0 \leq \hat{\alpha}_n \leq C,$$

where C is the regularization parameter to control overfitting, ϵ is the error sensitivity, and t is the target data. As $\alpha_n, \hat{\alpha}_n$ are the optimized parameters, they are encoded into variables of the QUBO problem with

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} a_{Kn+k}$$

$$\hat{\alpha}_n = \sum_{k=0}^{K-1} B^{k-P} a_{K(N+n)+k}.$$

Here K represents the number of actual qubits to use per variable, B is the base of the encoding and P is a parameter to enable negative exponents to the base. As a_n (the parameters on the QA) can only take binary values, the upper bound of α is defined as

$$C = \sum_{i=0}^{K-1} B^i.$$

To enforce the constraints to the Lagrangian optimization problem, two penalty terms with the Lagrangian parameters ξ and β are added to the formulation

$$\xi \left(\sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n)\right)^2,$$

$$\beta \left(\sum_{n=1}^{N} \alpha_n \hat{\alpha}_n\right).$$

Following some algebraic reformulations, the QUBO matrix Q can be derived (not shown here for brevity) and the problem is in a suitable form for the QA.

Run	SVR	Q-SVR 1	Q-SVR 2	Q-SVR 3	Q-SVR 4	Q-SVR 5	Q-SVR 6
1	7.961	10.9786	11.0244	11.0016	11.0811	10.2775	11.0849
2	12.1908	12.7688	12.7452	12.7919	12.9086	12.1013	12.9172
3	6.6338	10.5555	9.1293	11.2271	13.6113	10.9359	13.9183
4	9.6695	10.2681	10.2782	10.2851	10.3545	9.9055	10.3585
5	7.0943	11.7815	11.7834	11.7825	11.7844	10.9887	11.7845
6	12.4384	9.6309	9.6935	9.6967	9.7999	10.9737	9.8057
7	5.7134	9.7858	9.7859	9.7859	9.7859	9.1881	9.7859
8	6.8256	11.3143	11.4065	11.353	11.4646	11.1698	11.4684
9	5.9804	9.5261	9.5289	9.5272	9.5315	8.9554	9.5317
10	9.4644	13.3755	13.3756	13.3764	13.3795	12.1185	13.3798
Average	***8.3972***	***10.9985***	***10.8751***	***11.0827***	***11.3701***	***10.6614***	***11.4035***

Table 1: Mean squared error achieved by classical and Q-SVR models on ten different train/test splits on a small RS dataset. For the Q-SVR, six different solutions, each one with a different solution combination technique are reported.

Running the QUBO problem on the QA returns a set of solutions whose number is selected by the user as output. To obtain the final solution, the individual solutions are combined. In the literature, different methods for combining the solutions are proposed and benchmarked on a RS dataset [7]. Here, the Q-SVR method is used to estimate the chlorophyll concentration in water by considering some measurements carried out at different wavelengths as a feature vector. Each method is based on performing a weighted average of the candidate solutions and the difference between them is used to compute the coefficients of the weighted average. The results in Table 1 show that while the Q-SVR can outperform the classical SVR in some instances, on average the classical SVR still achieves lower errors. The different solutions are averaged by weights that are computed on the training set. To compute these weights, the Q-SVR-1 and -2 solution combination methods use a Mean Squared Error (MSE) loss function (with and without an additional SOFTMAX[9] loss), Q-SVR 3 and 4 use a log-hyperbolic cosine

[9] SOFTMAX https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.softmax.html

function (with and without an additional SOFTMAX loss), Q-SVR 5 picks just the single best solution and Q-SVR 6 performs a simple averaging.

2.3 Hyperparameter Optimization with Q-SVR

For ML models like Neural Networks (NNs), the performance is susceptible to the choice of hyperparameters that are set manually before the training process. These decisions involve not only the network's general architecture and the optimizer-related parameters but also include the selection of pre-processing methods.

As the hyperparameter search space is usually large, finding the best combinations of hyperparameters is computationally expensive. To evaluate the results of a combination, complete model training has to be performed. To save resources and energy, it is therefore of great interest to reduce the computing time required for HPO.

One way to solve this problem is to train multiple ML models with different hyperparameter combinations in parallel, to cut short the training process for models that show sub-par performance, and only train the most promising configurations until completion. In literature, it has been shown that regression methods can accurately predict the final performance of a training run by extrapolating the learning curves of NNs [8]. The regression methods use the hyperparameter configurations and a fraction of the learning curve as input and output the final loss or accuracy of the network, exemplarily shown in Figure 2 for the training of a ResNet18 model [9] on cifar-10.

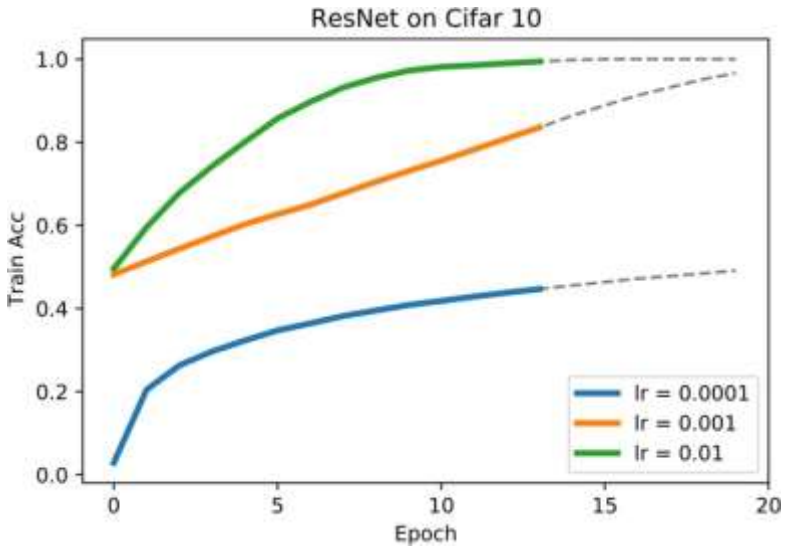


Figure 2: Partial learning curves of a ResNet18 [9] model trained on the cifar-10 dataset with different learning rates.

Q-SVR is one of the possible methods to perform this learning curve extrapolation. Others include classical SVR or linear regression methods. Q-SVR has the potential to outperform the classical SVR due to two reasons:

- The algorithmic complexity of classical SVR is between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, i.e., long runtimes can be expected for problems with a large number of samples. The QA only takes a few milliseconds to solve even larger optimization problem.
- The QA reports several thousand low-energy solutions for a problem instead of just a single one with each one of the multiple solutions fitted for certain features of the

training data. By combining these solutions, the final solution can be expected to generalize better to unseen data.

If the training of the NNs is performed on the GPUs of a classical supercomputer and the regression is performed on the QA, the hyperparameters of the NNs can be optimized with the following **hybrid Quantum-classical workflow,** which has been adopted in the ongoing activities in Task 2.3:

- Train different NNs with different hyperparameters partially on the GPUs of classical supercomputers (here the Jülich Research on Exascale Cluster Architectures (JURECA)-DC-GPU[10] partition at JSC has been used);
- Transfer the learning curves to JUPSI and perform extrapolation with Q-SVR;
- Fully train only the most promising models until completion on JURECA-DC-GPU.

To assess the performance of the workflow, benchmarks are performed on the cifar-10 dataset (consisting of 50,000 training images split into 10 classes and 10,000 validation images) using a simple ResNet18 model with varying learning rates[11]. A total of 300 different learning curves are generated by randomly sampling different learning rates in a log-uniform fashion from the interval $[10^{-4}, 1]$. All NNs are trained for 60 epochs. The learning rate and the first 25% of the learning curve are used as the input feature vector for the regression methods, while the output is the validation accuracy at epoch 60.

Although the JUPSI machine features more than 5,000 qubits, the number of samples that can be used in a single Q-SVR problem on the D-Wave QPU is still limited, as multiple qubits are required to represent each parameter of the problem. In these experiments, 20 samples per Q-SVR model are used for training, 20 samples for validation and finding the best hyperparameters of the SVR models (C and ε), and 20 samples for testing. The QUBO hyperparameters for the subsequent experiments are fixed at $K = 3$ (qubits per variable), $B = 2$ (encoding base), $P = 0.005$ (enabling negative exponents), and $\xi = 0.1$ (penalty term), while the chain strength is set to 4. The performance of the regression methods is measured in terms of the R^2 score[12] on the test dataset (a metric to calculate how well the regression predictions approximate the actual data points). A hybrid solver that has been developed by D-Wave can also be employed. This solver internally runs computations to pick the best solutions from several runs on the QPU and therefore can handle a larger number of samples.

A comparison of the classical SVR and Q-SVR method for 20 randomly chosen samples from the 300 learning curves is presented in Figure 3. In this case, both methods are able to accurately predict the final validation scores of the NN training with an R^2 score of 0.80 for the Q-SVR and 0.83 for the SVR. The plots in Figure 4 confirm that across a large number of evaluations (40 different train/validation/test splits), the classical SVR still achieves better results at average with a mean R^2 score of 0.73. The Q-SVR computed directly on the QPU can only achieve a mean R^2 score of 0.65, which can be increased to 0.68 with a post-processing method (pQSVR in Figure 4) that adaptively chooses the solutions that should be used from the several thousand reported by the QPU by scoring their performance on the training set. A linear regression model, i.e., an Ordinary Least Squares (OLS) model, is fitted as a baseline, achieving a mean R^2 value of 0.69. The hybrid D-Wave solver matches the performance of the classical SVR with a mean R^2 score of 0.73. It is interesting to observe that

[10] JURECA https://www.fz-juelich.de/en/ias/jsc/systems/supercomputers/jureca

[11] Results presented are part of a summer project in cooperation with Þorsteinn Elí, Tómas Kristinn, and Þorsteinn Ingólfsson from the UOI

[12] R2 score https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

for the maximum R^2 score achieved, regular Q-SVR and hybrid Q-SVR slightly outperform the classical SVR. It can be seen that there is a large difference between the mean and maximum R^2 score from the QA-based solutions, which suggest that the results are noisy. Hence, more research should be directed towards developing consistent predictions.

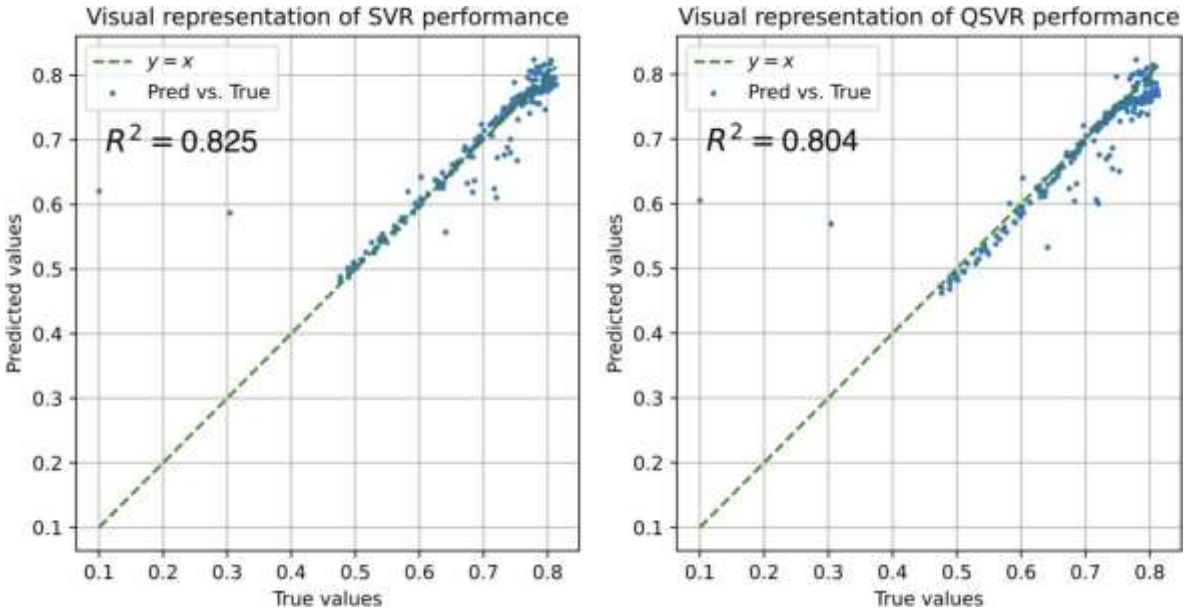


Figure 3: Comparison of average prediction performance of the classical SVR and Q-SVR methods for 20 random samples in terms of R^2 score.

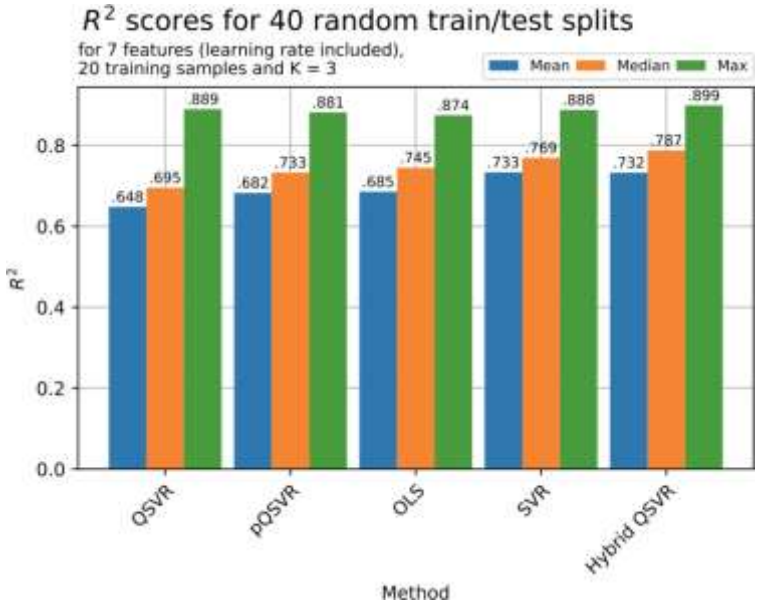


Figure 4: Comparison of R^2 score for Q-SVR, post-processed Q-SVR (pQSVR), linear regression (OLS), classical SVR, and Hybrid Q-SVR.

3 Benchmarking and support activities at CERN

The work presented in this section is part of Task 4.1 *"Event reconstruction and classification at the CERN HL-LHC"* within WP4 *"Data-Driven Use-Cases towards Exascale"*. To address the significant increase in data production that CERN's experiments will face in the coming decades, some traditional algorithms that run on CPUs may be replaced by NN-based algorithms that can easily be accelerated by GPUs. An example of one of these new algorithms is the Graph Neural Network (GNN)-based Machine-Learned Particle-Flow (MLPF) algorithm [10] that aims to replace the traditional particle-flow algorithm using a data-driven approach. This is a complex model for which it is crucial to achieve the best possible performance. Similar to FZJ's work described in Sec. 2, the potential of using performance prediction to speed up the HPO process is studied here, for the case of the MLPF model. The use of a QA to train the performance predictor is also studied and a method to overcome some of the problems is proposed that is derived from the current problem size limitations in Quantum systems while increasing the stability of solutions. With the proposed method, results on a Quantum machine comparable to those obtained by a classical regressor are achieved, showing how Quantum computers could be integrated within the classical ML tuning pipeline. The work presented here was done in close collaboration with FZJ for studying, developing, and using QA for Q-SVR.

In the following, the performance of Q-SVR for particle-flow reconstruction with an HPO framework is analyzed Sec. 3.1, while an ensemble method to increase the effective training size is developed in Sec. 3.2.

3.1 Predicting the performance of a GNN for particle-flow reconstruction using Q-SVR

The work presented in this section consists of a study of the potential of using a simple classical SVR model as a performance predictor for MLPF together with a simple proposal on how to use a trained SVR to improve an HPO process. In addition, the use of a QA to train a Q-SVR model as the performance predictor is studied. The current limitations of the QA technology that are encountered in this study are elucidated and strategies that have been adopted to manage them are discussed.

The goal is to train a performance predictor, e.g., an SVR, that takes the hyperparameter configuration of MLPF as input, as well as a partial learning curve generated by training MLPF with this hyperparameter configuration for a few epochs, and the output of the regressor is prediction of the final performance of MLPF. The validation loss is chosen as the measure of performance, which means that the performance predictor will predict the final validation loss at the end of MLPF training.

To train and evaluate the chosen predictor, a dataset with corresponding pairs of hyperparameter configurations and learning curves is generated. Approximately 300 hyperparameter configurations are randomly sampled from a search space consisting of 7 different hyperparameters and the MLPF is trained for 100 epochs for each one of these pairs. The publicly available Delphes dataset[13], which was first introduced in Pata et al. [10], is used

[13] Delphes dataset https://zenodo.org/record/4559324#.Y25MHS8w1zU

for the MLPF training. Following this procedure, a dataset with 296 learning curves is generated that is also publicly available[14].

Once the SVR training is represented as a QUBO problem (details in Sec. 2), it can be solved using the D-wave QA and the solutions provided by the QA can be used to reconstruct a trained Q-SVR (in dual formulation), that is able to make predictions. As mentioned in Sec. 2, Pasetto et al. [7] proposed several ways of reconstructing the trained Q-SVR from the D-Wave system solutions. For this work, all those methods as well as another one consisting of using only the solution corresponding to the lowest energy state (or one of them in case there are several that have the same energy) returned by the D-Wave solver are employed.

Due to the limited number of qubits available, only 20 samples can be read on the QA and hence the feature vector is reduced to have only 13 features obtained from the down-sampled learning curve for 25 epochs without any hyperparameter information. In other words, the size of the input is reduced to only 13 elements, derived from the down-sampled learning curve, leaving out the hyperparameter values.

The hyperparameters of the Q-SVR implementation is divided into three categories: SVR, QUBO, and Quantum parameters, as shown in Table 2 below.

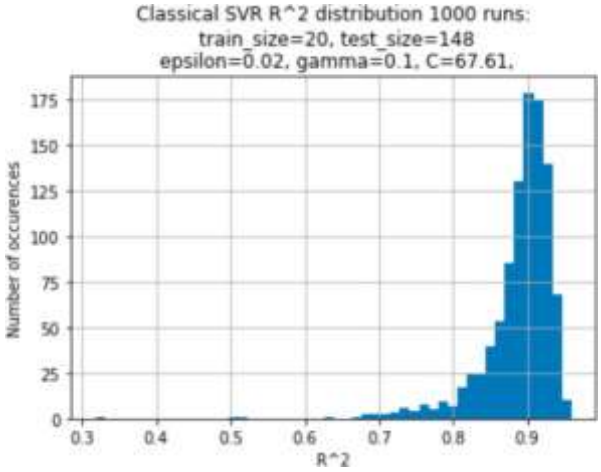
	Hyperparameter	Description
	`percentage_used`	Fraction of the multiple Quantum solutions used to reconstruct the SVR
SVR	ϵ	Distance from the actual value with no penalty
	γ	For RBF kernel
	C	Penalty term for misclassifying
QUBO	K	Number of binary variables used to represent a continuous one
	B	Base for the encoding
	k_0	Negative exponent for encoding
	ξ	Penalty term
Quantu	`anneal_time`	Duration of each annealing cycle
	`num_reads`	Number of annealing cycles (number of Quantum solutions)
	`chain_mult`	Strength for pairing physical qbits that represent the same variable

Table 2: Q-SVR hyperparameters.

After several runs of the Q-SVR, despite the instability of the results, a set of SVR hyperparameters that seemed to perform better than others are identified. However, it is observed that certain runs still result in low and even negative R^2 values. Hence, it needs to be determined which set of hyperparameters (SVR, QUBO, or Quantum) are responsible for the worse-performing trials. One way of doing that is to study and compare results from a classical SVR (giving information regarding the classical SVR parameters), a QSVR trained using simulated QA (giving information regarding the QUBO parameters) and a QSVR trained using actual QA (giving information regarding the Quantum parameters). In order to determine this, 1,000 classical SVRs were run with the same configuration that worked best for the Q-SVR using a random train/test split, each time consisting of 20 training points and 150 test points. The results of these 1,000 runs, which can be seen in Figure 5, show that the classical

[14] DELPHES processed https://github.com/JP-Amboage/performance-pred/blob/main/data/mlpf/delphes_trainings_processed.csv

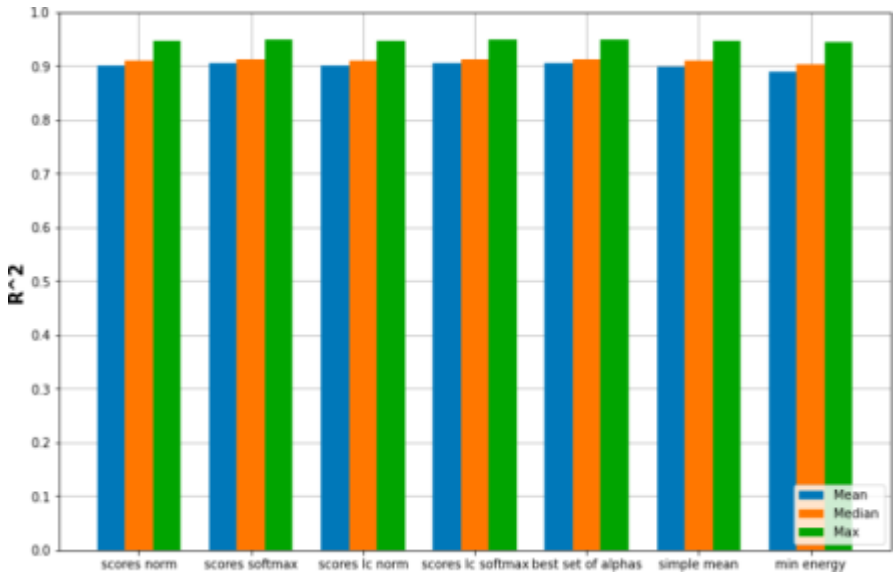
SVR can achieve a good performance with the previously described configuration despite the low number of training points. Thus, the performance issues of the Q-SVR when this configuration is used are more likely arising from the QUBO and Quantum hyperparameters.



R^2 Summary	
Maximum	0.957
Minimum	0.437
Mean	0.890
Median	0.902
Standard Deviation	0.048

Figure 5: Resulting R^2 scores after training and testing an SVR on 1,000 different splits.

While retaining the previous set of SVR hyperparameters, the influence of the QUBO hyperparameters is now studied using Q-SVR, but setting the D-Wave sampler (the function that receives the QUBO problem and returns the solutions) to use simulated annealing instead of QA. As the QUBO problem is still the input for the sampler, the translation from SVR to QUBO can effectively be tested without having to worry about the time constraints imposed while using the Quantum system, such as the Quantum hyperparameters or possible interference effects in the Quantum annealing process. The parameters $K = 3$, $B = 0.5$, $k_0 = 0.005$, and $\xi = 0.01$ are used and an experiment using simulated annealing on 100 different splits is conducted, which gives the results shown in Figure 6. It can be seen that the obtained results are comparable to those from the classical SVR, hence the hyperparameters that still need to be optimized are those related to the Quantum machine.



R^2 Summary	
Maximum	0.949
Minimum	0.383
Mean	0.901
Median	0.910
Standard Deviation	0.045

Figure 6: Resulting R^2 after training and testing the Simulated Annealing Q-SVR on 100 different splits with `train_size`=20, $\epsilon = 0.02$, $\gamma = 0.01$, $C = 67.61$, $K = 3$, $B = 0.5$, $k_0 = 0.005$, and $\xi = 0.01$. The different groups of bars on the plot correspond to the different reconstruction techniques.

Several experiments are conducted to find the optimal Quantum hyperparameters. Following these trials, it is concluded that a `chain_mult` value of 10 works well for this application. Thereafter, 10 Q-SVR on different train/test splits are run using this value. The results are shown in Figure 7 and it can be seen that a configuration for the Q-SVR that is able to provide results comparable to those from the classical SVR is obtained.

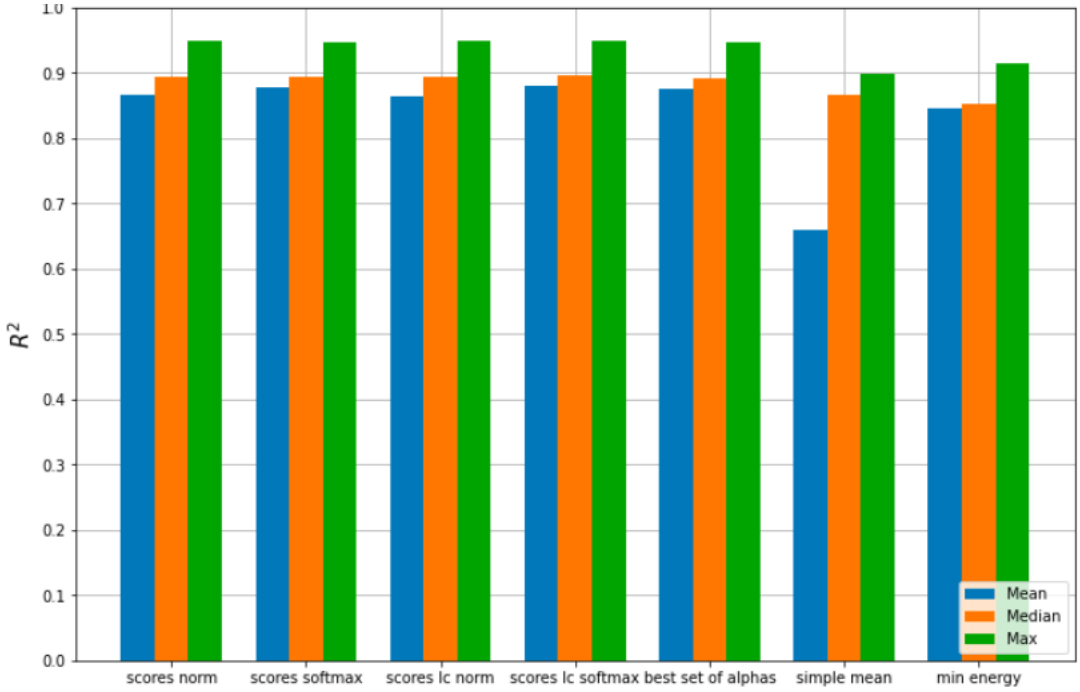


Figure 7: Resulting R^2 values after training and testing the Q-SVR on an actual QPU on 10 different splits with `train_size`=20, $\epsilon = 0.02$, $\gamma = 0.01$, $C = 67.61$, $K = 3$, $B = 0.5$, $k_0 = 0.005$, $\xi = 0.01$, `num_reads`=1000, `chain_mult`=10, `anneal_time`=20, and `percentage_used`=0.004. The different groups of bars on the plot correspond to the different reconstruction techniques.

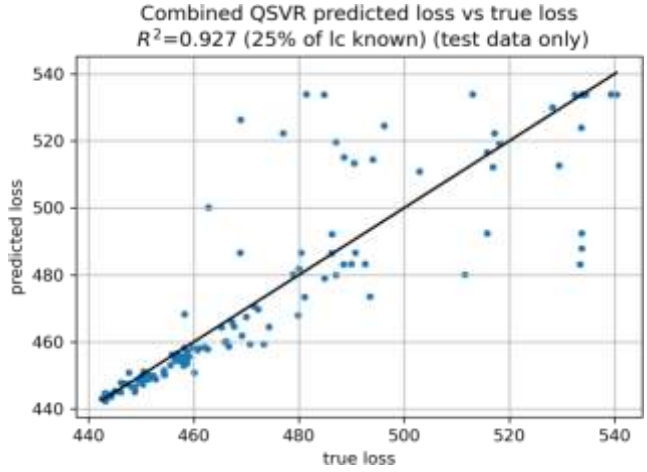
3.2 Q-SVR ensemble to increase effective training set size

As described in previous sections, one of the disadvantages of training an SVR in a Quantum device is the strong limitation on the training set size. In an attempt to overcome this limitation, different techniques to use a larger training set by combining several Q-SVRs trained on 20 separate samples each is studied. It is observed that the most successful approach was to split the training set into disjoint subsets of 20 points and train a Q-SVR for each subset. Then, to make a prediction, all the Q-SVRs are used separately and the final prediction is calculated as the average of all the individual predictions.

The design of the algorithm benefits from the fact that for each Q-SVR, only a fraction of the total training set is used. Therefore, the remaining points of the training set are used to evaluate the R^2 score of each Q-SVR for each reconstruction method. When making predictions, only the prediction made by the reconstruction method that reached the highest R^2 value during training of each Q-SVR is used. Finally, the predictions of all Q-SVRs are combined by means of a weighted average using their R^2 values as weights.

To evaluate this ensemble-like Q-SVR combination technique, 80 points for training and 150 points for testing are used and the experiment for 10 random train/test splits is repeated. Note that as 80 training points are used, each prediction is made by combining 4 Q-SVRs trained with 20 points each. The results of this experiment can be seen in Figure 8 and they show that the predictions under the combination technique are more stable than those achieved using a single Q-SVR trained with 20 points. This is interesting as one of the problems faced while

using the Q-SVR is the instability of the results. Bigger training sets have been tested but the results did not improve. A point of further improvement might be to modify the calculation of the weights in the weighted average that combines the Q-SVRs as they are likely to have similar weights in the current method.



R^2 Summary	
Maximum	0.927
Minimum	0.857
Mean	0.899
Median	0.897
Standard Deviation	0.019

Figure 8: Results after running the Q-SVR combination technique 10 times for 10 different train/test splits using 80 training samples.

4 Benchmarking and support activities at UOI

The work presented in this section is part of Task 4.2 *"Seismic imaging with remote sensing for energy applications"* within WP4. The task of multispectral image classification in the field of RS involves assigning each pixel of a satellite image to a certain class, e.g., the type of land cover or type of building. As the number of pixels even in a single image is large, ML models, which perform the assignment to classes in an automated way, are employed. One option for such a method is the SVM algorithm for classification. SVM has historically performed well in this context. Research has been conducted on the possibility of enhancing SVM by leveraging the computing power of Quantum annealing, considering RS as a benchmark application.

In the following, the performance of QMSVM for image classification and the achieved results are summarized in Sec. 4.1. Some final remarks are given in Sec. 4.2.

4.1 Quantum Multiclass SVM for Multispectral Image Classification

As SVM is originally defined for binary classification tasks, an adaptation for RS datasets is required, which often feature more than two classes. To achieve this, a common approach is the One-Versus-One (OVO) method, where each pair of classes defines an SVM binary classifier and the outcome is decided by max-voting, i.e., the prediction output obtained by the highest number of SVM classifiers. This approach is, however, computationally expensive for a large number of pixels, as its algorithmic complexity is $\mathcal{O}(n^3)$. Therefore, the QMSVM algorithm has been developed [11]. It consists of a single optimization step performed by a QA. It differs from traditional multiclass approaches, which consist of multiple models and multiple training steps. The training is based on a small training subset, due to the memory limitation of the JUPSI machine, and requires a fixed, limited amount of time. The maximum size of the training subset can be determined empirically and, in this case, it is around 60 samples. The whole training set is then used in post-processing, where the solutions obtained by JUPSI are refined. This has shown to have a low impact on the overall execution time.

(a) (b) (c)

Figure 9: Land cover classification maps for a SemCity Toulouse dataset tile, classes "water" (blue), "pervious surface" (green), "building" (orange) and 20,000 training samples: ground-truth (a), OVO (b), and QMSVM (c).

Figure 9 shows the labeled map (ground-truth) of an urban area of the city of Toulouse and an example classification map predicted by OVO and QMSVM in a multispectral image classification task. Figure 10 shows the results of benchmarking the QMSVM and OVO approach in terms of accuracy achieved and execution time over the number of training set size N. The benchmark is performed on the SemCity Toulouse dataset [12], where each pixel belongs to one of seven classes. The results show that the classical OVO method in general

achieves only slightly better classification accuracy on the test set than the QMSVM method. In this case, the execution time for the QMSVM remains constant at approximately 100 seconds while it increases for the OVO method with an increasing number of training samples.

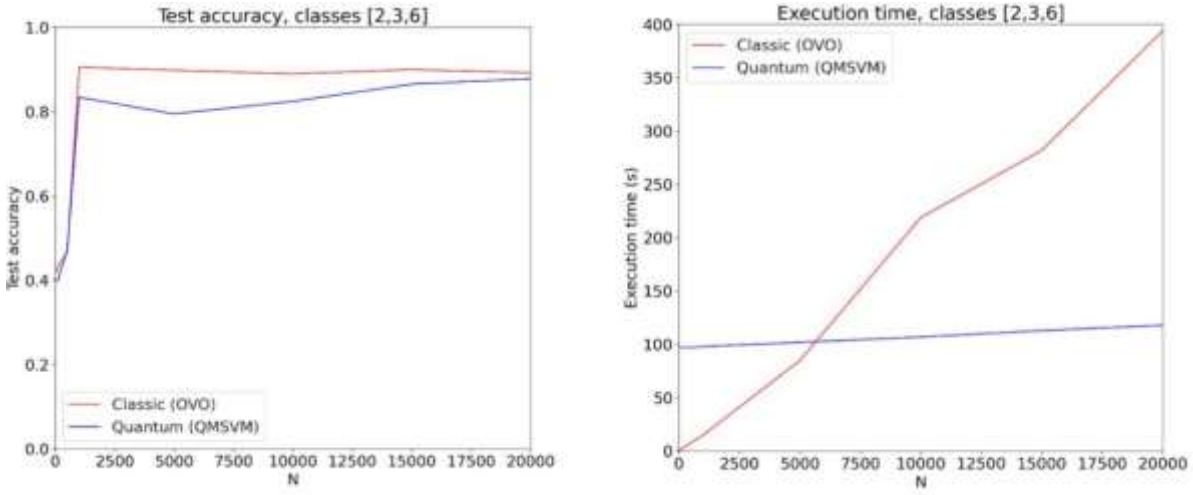


Figure 10: Test accuracy and execution time of OVO and QMSVM method for classes "water" (2), "pervious surface" (3), "building" (6) with respect to the training set size N.

4.2 Remarks

It has been shown that the SVM can be adapted to be run on the JUPSI machine, both for binary and multiclass classification. The memory limitation of the machine is still a challenge when addressing real applications. This constraint is even more evident in the multiclass case. However, this problem can be addressed, for example in earlier investigations, ensembles of binary Q-SVM classifiers have proven to be effective. In this work, it has been shown that applying a post-processing step to the output of the QA can also improve the overall accuracy of the classifier with little impact on the overall time.

5 Summary and conclusions

This Deliverable is the first in the context of Task 2.3, which commenced in M19 of the project. It highlighted the developments that have taken place in the first six months of this task. The use cases considered in this time period came from Task 4.1 *"Event reconstruction and classification at the CERN HL-LHC"* and Task 4.2 *"Seismic imaging with remote sensing for energy applications"*, both part of WP4. Furthermore, within WP2, AI-based benchmark applications were selected to test the QA.

With the hybrid Quantum-classical workflow for HPO, as proposed in Sec. 2, the JUPSI machine can aid in the computation of large-size problems, as they are present across many different tasks in CoE RAISE. The current limits in terms of the number of qubits can be circumvented by offloading only the smaller learning curve prediction problem to the QA and keeping the computation of the larger NN training on the powerful GPUs of a classical supercomputer. While the first results are still noisy and classical regression methods outperform the Quantum methods, this is a promising direction of research. The next steps include working on stabilizing the results of the QA to reduce the noise, and also to explore other Quantum methods that can help in the HPO process.

Furthermore, the strong potential of using performance prediction techniques for MLPF was shown, leaving the door open for the use of this technique in future HPO cycles of MLPF. It was demonstrated that, despite the current limitations of Quantum computers, it is possible to train an SVR on a QA achieving prediction performance comparable to those obtained on a classical SVR. Even though QC technologies are still at an early stage and QAs may be more suitable for other types of problems (like graph or combinatorics problems), it has been shown that Quantum technologies can potentially have a place in the classical ML design and hyperparameter tuning pipeline.

Finally, RS image classification was considered for benchmarking Quantum ML algorithms. QA has been proven effective for optimizing SVM-based models, which have shown good accuracy. Current work has laid the foundation for overcoming the memory limitation of current QA systems and addressing large-scale applications in RS.

Besides the developments presented in this Deliverable, trainings were also organized by WP2 on Q-SVM[15], which will be beneficial for communities within and outside RAISE. Such trainings will be continued as new developments occur. In the next phase, other methods for improving the accuracy of the QA predictions will be investigated. The hybrid computing setup can also be useful for applications to other use cases within CoE RAISE, which will also be explored.

[15] Q-SVM training https://www.youtube.com/watch?v=WBRfpBRSepg

References

[1] Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse Ising model. Phys. Rev. E, 58, 5355–5363. https://doi.org/10.1103/PhysRevE.58.5355

[2] Willsch, D., Willsch, M., De Raedt, H., & Michielsen, K. (2020). Support vector machines on the D-Wave Quantum annealer. Computer Physics Communications, 248, 107006. https://doi.org/10.1016/j.cpc.2019.107006

[3] Cavallaro, G., Willsch, D., Willsch, M., Michielsen, K., & Riedel, M. (2020). Approaching Remote Sensing Image Classification with Ensembles of Support Vector Machines on the D-Wave Quantum Annealer. In IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, 1973-1976. https://doi.org/10.1109/IGARSS39084.2020.9323544

[4] Delilbasic, A., Cavallaro, G., Willsch, M., Melgani, F., Riedel, M., & Michielsen, K. (2021). Quantum Support Vector Machine Algorithms for Remote Sensing Data Classification. In 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS (pp. 2608-2611). https://doi.org/10.1109/IGARSS47720.2021.9554802

[5] Wang, G. (2017). Quantum algorithm for linear regression. Phys. Rev. A, 96, 012335. https://doi.org/10.1103/PhysRevA.96.012335

[6] Pasetto, E., Riedel, M., Melgani, F., Michielsen, K., & Cavallaro, G. (2022). Quantum SVR for Chlorophyll Concentration Estimation in Water With Remote Sensing. IEEE Geoscience and Remote Sensing Letters, 19, 1-5. https://doi.org/10.1109/LGRS.2022.3200325

[7] Pasetto, E., Riedel, M., Melgani, F., Michielsen, K., & Cavallaro, G. (2022). Quantum SVR for Chlorophyll Concentration Estimation in Water With Remote Sensing. IEEE Geoscience and Remote Sensing Letters, 19, 1-5. https://doi.org/10.1109/LGRS.2022.3200325

[8] Baker, B., Gupta, O., Raskar, R., & Naik, N.. (2017). Accelerating Neural Architecture Search using Performance Prediction. Retrieved from https://arxiv.org/abs/1705.10823

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90

[10] Pata, J., Duarte, J., Vlimant, JR., Pierini, M., & Spiropulu, M. (2021). MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. The European Physical Journal C 81 (381). https://doi.org/10.1140/epjc/s10052-021-09158-w

[11] Delilbasic, A., Cavallaro, G., Le Saux, B. Riedel, M., & Michielsen, K. (2022). Multiclass SVM with Quantum Annealing. (preprint)

[12] Roscher, R., Volpi, M., Mallet, C., Drees, L., & Wegner, J. (2020). SemCity Toulouse: A Benchmark for Building Instance Segmentation in Satellite Images. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, V-5-2020, 109–116. https://doi.org/10.5194/isprs-annals-V-5-2020-109-2020

List of Acronyms and Abbreviations

AI	Artificial Intelligence
BSC	Barcelona Supercomputing Centre, Spain
CERN	European Center for Nuclear Research
CoE RAISE	European Center of Excellence in Exascale Computing “Research on AI- and Simulation-Based Engineering at Exascale”
CPU	Central Processing Unit
DL	Deep Learning
FZJ	Forschungszentrum Jülich
GNN	Graph Neural Network
GPU	Graphical Processing Unit
HEP	High-Energy Physics
HPC	High Performance Computing
HPO	Hyperparameter Optimization
JARA	Jülich Aachen Research Alliance
JUNIQ	Jülich UNified Infrastructure for Quantum computing
JUPSI	Jülich Pioneer for Spin Interference
JURECA	Jülich Research on Exascale Cluster Architectures
JUWELS	Jülich Wizard for European Leadership Science
ML	Machine Learning
MLPF	Machine-Learned Particle-Flow
MSA	Modular Supercomputing Architecture
MSE	Mean Squared Error
NN	Neural Network
OLS	Ordinary Least Squares
OVO	One-Versus-One
ParTec	ParTec AG, Munich, Germany. Linked third Party of FZJ in CoE RAISE
QA	Quantum Annealer
QC	Quantum Computing
QPU	Quantum Processing Unit
Q-SVM	Quantum Support Vector Machine
Q-SVR	Quantum Support Vector Regression
QMSVM	Quantum Multiclass Support Vector Machine
QUBO	Quadratic Unconstrained Binary Optimization
RAISE	see CoE RAISE
RBF	Radial Basis Function
RS	Remote Sensing
SVM	Support Vector Machine
SVR	Support Vector Regression
UOI	University of Iceland
WP	Work Package